9.4

IBM MQ - Riferimento per lo sviluppo delle applicazioni

Nota

Prima di utilizzare queste informazioni e il prodotto che supportano, leggere le informazioni in “Informazioni particolari” a pagina 2255.

Questa edizione si applica alla versione 9 release 4 di IBM® MQ e a tutte le successive release e modifiche se non diversamente indicato nelle nuove edizioni.

Quando si inviano informazioni a IBM, si concede a IBM un diritto non esclusivo di utilizzare o distribuire le informazioni in qualsiasi modo ritenga appropriato senza incorrere in alcun obbligo verso l'utente.

Indice

Riferimento per lo sviluppo di applicazioni

Utilizzare i link forniti in questa sezione per sviluppare le applicazioni IBM MQ .

- “Riferimento applicazioni MQI” a pagina 7
- IBM i “ILE/RPG (IBM i Application Programming Reference)” a pagina 1022
- IBM i “Conversione dati su IBM i” a pagina 1477
- “Riferimenti alle uscite utente, alle uscite API e ai servizi installabili” a pagina 1497
- “Classi e interfacce di IBM MQ .NET” a pagina 1769
- “Classi C++ IBM MQ” a pagina 1833
- “Proprietà degli oggetti IBM MQ classes for JMS” a pagina 1944
- “Riferimento REST API di messaggistica” a pagina 2221

Attività correlate

Sviluppo di applicazioni

Riferimenti correlati

Classi IBM MQ per librerie Java

Classi IBM MQ per JMS

Riferimento applicazioni MQI

Utilizzare i collegamenti forniti in questa sezione per sviluppare le applicazioni MQI (Message Queue Interface).

- “Esempi di codice” a pagina 8
- “Costanti” a pagina 61
- “Tipi di dati utilizzati in MQI” a pagina 236
- “Chiamate della funzione” a pagina 642
- “Attributi degli oggetti” a pagina 818
- “Codici di ritorno” a pagina 898
- “Regole per la convalida delle opzioni MQI” a pagina 899
- “Codifiche macchina” a pagina 925
- “Opzioni di report e indicatori di messaggi” a pagina 928
- “Uscita conversione dati” a pagina 932
- “Proprietà specificate come elementi MQRFH2” a pagina 956
- “Conversione code page” a pagina 965

Concetti correlati

“Riferimenti alle uscite utente, alle uscite API e ai servizi installabili” a pagina 1497
Utilizzare le linformation in questa sezione per sviluppare le uscite utente, le uscite API e le applicazioni dei servizi installabili:

Attività correlate

Sviluppo di applicazioni

Riferimenti correlati

“Classi e interfacce di IBM MQ .NET” a pagina 1769
Le classi e interfacce IBM MQ .NET sono elencate in ordine alfabetico. Vengono descritti le proprietà, i metodi e i costruttori.

“Classi C++ IBM MQ” a pagina 1833
Le classi IBM MQ C+ + incapsulano l'interfaccia MQI (Message Queue Interface) IBM MQ . Esiste un singolo file di intestazione C + +, **imqi.hpp**, che copre tutte queste classi.

Le classi IBM MQ per le librerie Java

IBM MQ Classi per JMS

Esempi di codice

Utilizzare le informazioni di riferimento fornite in questa sezione per svolgere le attività utili alle proprie esigenze aziendali.

Esempi di linguaggio C

Questa raccolta di argomenti è per lo più tratta dalle applicazioni di esempio IBM MQ for z/OS . Sono applicabili a tutte le piattaforme, tranne dove indicato.

Connessione a un gestore code

Questo esempio illustra come utilizzare la chiamata MQCONN per connettere un programma a un gestore code in batch z/OS .

Questo estratto è estratto dall'applicazione di esempio Sfoglia (programma CSQ4BCA1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
#include <cmqc.h>
.
.
.
static char Parm1[MQ_Q_MGR_NAME_LENGTH] ;

int  main(int argc, char *argv[] )
   {
   /*                                                  */
   /*     Variables for MQ calls                       */
   /*                                                  */
   MQHCONN Hconn;      /* Connection handle            */
   MQLONG  CompCode;   /* Completion code              */
   MQLONG  Reason;     /* Qualifying reason            */

   /* Copy the queue manager name, passed in the       */
   /* parm field, to Parm1                             */
   strncpy(Parm1,argv[1],MQ_Q_MGR_NAME_LENGTH);

   /*                                                  */
   /* Connect to the specified queue manager.          */
   /*   Test the output of the connect call.  If the   */
   /*   call fails, print an error message showing the */
   /*   completion code and reason code, then leave the*/
   /*   program.                                       */
   /*                                                  */
   MQCONN(Parm1,
          &Hconn,
          &CompCode,
          &Reason);
   if ((CompCode != MQCC_OK) | (Reason != MQRC_NONE))
      {
      sprintf(pBuff, MESSAGE_4_E,
              ERROR_IN_MQCONN, CompCode, Reason);
      PrintLine(pBuff);
      RetCode = CSQ4_ERROR;
      goto AbnormalExit2;
      }
      .
      .
      .
}
```

Disconnessione da un gestore code

Questo esempio illustra come utilizzare la chiamata MQDISC per disconnettere un programma da un gestore code nel batch z/OS .

Le variabili utilizzate in questo estratto di codice sono quelle impostate in “Connessione a un gestore code” a pagina 8. Questo estratto è estratto dall'applicazione di esempio Sfoglia (programma CSQ4BCA1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
⋮
   /*                                                  */
   /*   Disconnect from the queue manager.  Test the   */
   /*   output of the disconnect call.  If the call    */
   /*   fails, print an error message showing the      */
   /*   completion code and reason code.               */
   /*                                                  */
   MQDISC(&Hconn,
          &CompCode,
          &Reason);
   if ((CompCode != MQCC_OK) || (Reason != MQRC_NONE))
      {
      sprintf(pBuff, MESSAGE_4_E,
              ERROR_IN_MQDISC, CompCode, Reason);
      PrintLine(pBuff);
      RetCode = CSQ4_ERROR;
      }
⋮
```

Creazione di una coda dinamica

Questo esempio illustra come utilizzare la chiamata MQOPEN per creare una coda dinamica.

Questo estratto è estratto dall'applicazione di esempio Mail Manager (programma CSQ4TCD1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
⋮
MQLONG  HCONN = 0;   /* Connection handle      */
MQHOBJ  HOBJ;        /* MailQ Object handle    */
MQHOBJ  HobjTempQ;   /* TempQ Object Handle    */
MQLONG  CompCode;    /* Completion code        */
MQLONG  Reason;      /* Qualifying reason      */
MQOD    ObjDesc = {MQOD_DEFAULT};
                     /* Object descriptor      */
MQLONG  OpenOptions; /* Options control MQOPEN */

   /*---------------------------------------- */
   /* Initialize the Object Descriptor (MQOD)  */
   /* control block.  (The remaining fields    */
   /* are already initialized.)                */
   /*-----------------------------------------*/
   strncpy( ObjDesc.ObjectName,
            SYSTEM_REPLY_MODEL,
            MQ_Q_NAME_LENGTH );
   strncpy( ObjDesc.DynamicQName,
            SYSTEM_REPLY_INITIAL,
            MQ_Q_NAME_LENGTH );
   OpenOptions = MQOO_INPUT_AS_Q_DEF;
   /*-----------------------------------------*/
   /* Open the model queue and, therefore,     */
   /* create and open a temporary dynamic      */
   /* queue                                    */
   /*-----------------------------------------*/
   MQOPEN( HCONN,
           &ObjDesc,
           OpenOptions,
           &HobjTempQ,
           &CompCode,
           &Reason );
   if ( CompCode == MQCC_OK ) {

   }
   else {
      /*-------------------------------------*/
      /* Build an error message to report the  */
      /* failure of the opening of the model   */
      /* queue                                 */
      /*-------------------------------------*/
      MQMErrorHandling( "OPEN TEMPQ", CompCode,
```

```
                          Reason );
      ErrorFound = TRUE;
   }
   return ErrorFound;
}
.
```

Apertura di una coda esistente

Questo esempio mostra come utilizzare la chiamata MQOPEN per aprire una coda che è già stata definita.

Questo estratto è estratto dall'applicazione di esempio Sfoglia (programma CSQ4BCA1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
#include <cmqc.h>
.
static char Parm1[MQ_Q_MGR_NAME_LENGTH];
.
int  main(int argc, char *argv[] )
   {
   /*
   /*     Variables for MQ calls                          */
   /*
   MQHCONN Hconn ;            /* Connection handle       */
   MQLONG  CompCode;          /* Completion code         */
   MQLONG  Reason;            /* Qualifying reason       */
   MQOD    ObjDesc = { MQOD_DEFAULT };
                              /* Object descriptor       */
   MQLONG  OpenOptions;       /* Options that control    */
                              /*  the MQOPEN call        */
   MQHOBJ  Hobj;              /* Object handle           */
   .
   /*  Copy the queue name, passed in the parm field,    */
   /*  to Parm2 strncpy(Parm2,argv[2],         */
   /*  MQ_Q_NAME_LENGTH);                                */
   .
   /*                                                    */
   /*  Initialize the object descriptor (MQOD) control   */
   /*  block.  (The initialization default sets StrucId, */
   /*  Version, ObjectType, ObjectQMgrName,              */
   /*  DynamicQName, and AlternateUserid fields)         */
   /*                                                    */
   strncpy(ObjDesc.ObjectName,Parm2,MQ_Q_NAME_LENGTH);
   .
   /*  Initialize the other fields required for the open */
   /*  call (Hobj is set by the MQCONN call).            */
   /*                                                    */
   OpenOptions  = MQOO_BROWSE;
   .
   /*                                                    */
   /* Open the queue.                                    */
   /*   Test the output of the open call.  If the call   */
   /*   fails, print an error message showing the        */
   /*   completion code and reason code, then bypass     */
   /*   processing, disconnect and leave the program.    */
   /*                                                    */
   MQOPEN(Hconn,
          &ObjDesc,
          OpenOptions,
          &Hobj,
          &CompCode,
          &Reason);
```

```
   if ((CompCode != MQCC_OK) || (Reason != MQRC_NONE))
      {
      sprintf(pBuff, MESSAGE_4_E,
              ERROR_IN_MQOPEN, CompCode, Reason);
      PrintLine(pBuff);
      RetCode = CSQ4_ERROR;
      goto AbnormalExit1;       /* disconnect processing */
      }
   .
   } /* end of main */
```

Chiusura di una coda

Questo esempio illustra come utilizzare la chiamata MQCLOSE per chiudere una coda.

Questo estratto è estratto dall'applicazione di esempio Sfoglia (programma CSQ4BCA1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
.
   /*                                                  */
   /*   Close the queue.                               */
   /*   Test the output of the close call.  If the call  */
   /*   fails, print an error message showing the        */
   /*   completion code and reason code.               */
   /*                                                  */
   MQCLOSE(Hconn,
           &Hobj,
           MQCO_NONE,
           &CompCode,
           &Reason);
   if ((CompCode != MQCC_OK) || (Reason != MQRC_NONE))
      {
      sprintf(pBuff, MESSAGE_4_E,
              ERROR_IN_MQCLOSE, CompCode, Reason);
      PrintLine(pBuff);
      RetCode = CSQ4_ERROR;
      }
.
```

Inserimento di un messaggio utilizzando MQPUT

Questo esempio illustra come utilizzare la chiamata MQPUT per inserire un messaggio su una coda.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ. Per i nomi e i percorsi delle applicazioni di esempio, vedere Programmi procedurali di esempio (piattaforme eccetto z/OS) z/OS e Programmi di esempio per IBM MQ for z/OS .

```
.
qput()
{
     MQMD    MsgDesc;
     MQPMO   PutMsgOpts;
     MQLONG  CompCode;
     MQLONG  Reason;
     MQHCONN Hconn;
     MQHOBJ  Hobj;
     char message_buffer[] = "MY MESSAGE";
    /*-------------------------------*/
    /*  Set up PMO structure.        */
    /*-------------------------------*/
     memset(&PutMsgOpts, '\0', sizeof(PutMsgOpts));
     memcpy(PutMsgOpts.StrucId, MQPMO_STRUC_ID,
            sizeof(PutMsgOpts.StrucId));
     PutMsgOpts.Version = MQPMO_VERSION_1;
     PutMsgOpts.Options = MQPMO_SYNCPOINT;

    /*-------------------------------*/
    /*  Set up MD structure.         */
    /*-------------------------------*/
    memset(&MsgDesc, '\0', sizeof(MsgDesc));
    memcpy(MsgDesc.StrucId, MQMD_STRUC_ID,
           sizeof(MsgDesc.StrucId));
    MsgDesc.Version     =  MQMD_VERSION_1;
    MsgDesc.Expiry      =  MQEI_UNLIMITED;
    MsgDesc.Report      =  MQRO_NONE;
    MsgDesc.MsgType     =  MQMT_DATAGRAM;
    MsgDesc.Priority    = 1;
    MsgDesc.Persistence = MQPER_PERSISTENT;
    memset(MsgDesc.ReplyToQ,
           '\0',
           sizeof(MsgDesc.ReplyToQ));
    /*--------------------------------------------------*/
    /*  Put the message.                                */
    /*--------------------------------------------------*/
    MQPUT(Hconn, Hobj, &MsgDesc, &PutMsgOpts,
```

```
          sizeof(message_buffer), message_buffer,
          &CompCode, &Reason);
```

```
    /*-------------------------------------*/
    /*  Check completion and reason codes. */
    /*-------------------------------------*/
    switch (CompCode)
        {
         case MQCC_OK:
             break;
         case MQCC_FAILED:
             switch (Reason)
                 {
                  case MQRC_Q_FULL:
                  case MQRC_MSG_TOO_BIG_FOR_Q:
                      break;
                  default:
                      break; /* Perform error processing */
                  }
             break;
         default:
             break;          /* Perform error processing */
        }
}
```

Inserimento di un messaggio utilizzando MQPUT1

Questo esempio illustra come utilizzare la chiamata MQPUT1 per aprire una coda, inserire un singolo messaggio nella coda e chiudere la coda.

Questo estratto viene estratto dall'applicazione di esempio Credit Check (programma CSQ4CCB5) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
⋮
MQLONG   Hconn;              /* Connection handle        */
MQHOBJ   Hobj_CheckQ;        /* Object handle            */
MQLONG   CompCode;           /* Completion code          */
MQLONG   Reason;             /* Qualifying reason        */
MQOD     ObjDesc    = {MQOD_DEFAULT};
                             /* Object descriptor        */
MQMD     MsgDesc    = {MQMD_DEFAULT};
                             /* Message descriptor       */
MQLONG   OpenOptions;        /* Control the MQOPEN call  */

MQGMO    GetMsgOpts = {MQGMO_DEFAULT};
                             /* Get Message Options      */
MQLONG   MsgBuffLen;         /* Length of message buffer */
CSQ4BCAQ MsgBuffer;          /* Message structure        */
MQLONG   DataLen;            /* Length of message        */

MQPMO    PutMsgOpts = {MQPMO_DEFAULT};
                             /* Put Message Options      */
CSQ4BQRM PutBuffer;          /* Message structure        */
MQLONG   PutBuffLen = sizeof(PutBuffer);
                             /* Length of message buffer */
⋮
```

```
void Process_Query(void)
   {
   /*                                                    */
   /* Build the reply message                            */
   /*                                                    */
   ⋮
   /*                                                    */
   /* Set the object descriptor, message descriptor and  */
   /* put message options to the values required to      */
   /* create the reply message.                          */
   /*                                                    */
   strncpy(ObjDesc.ObjectName, MsgDesc.ReplyToQ,
           MQ_Q_NAME_LENGTH);
   strncpy(ObjDesc.ObjectQMgrName, MsgDesc.ReplyToQMgr,
           MQ_Q_MGR_NAME_LENGTH);
   MsgDesc.MsgType = MQMT_REPLY;
```

```
   MsgDesc.Report  = MQRO_NONE;
   memset(MsgDesc.ReplyToQ, ' ', MQ_Q_NAME_LENGTH);
   memset(MsgDesc.ReplyToQMgr, ' ', MQ_Q_MGR_NAME_LENGTH);
   memcpy(MsgDesc.MsgId, MQMI_NONE, sizeof(MsgDesc.MsgId));
   PutMsgOpts.Options = MQPMO_SYNCPOINT +
                        MQPMO_PASS_IDENTITY_CONTEXT;
   PutMsgOpts.Context = Hobj_CheckQ;
   PutBuffLen = sizeof(PutBuffer);
   MQPUT1(Hconn,
          &ObjDesc,
          &MsgDesc,
          &PutMsgOpts,
          PutBuffLen,
          &PutBuffer,
          &CompCode,
          &Reason);

   if (CompCode != MQCC_OK)
      {
      strncpy(TS_Operation, "MQPUT1",
              sizeof(TS_Operation));
      strncpy(TS_ObjName, ObjDesc.ObjectName,
              MQ_Q_NAME_LENGTH);
      Record_Call_Error();
      Forward_Msg_To_DLQ();
      }
   return;
   }
   .
   .
   .
```

acquisizione di un messaggio

Questo esempio mostra come utilizzare la chiamata MQGET per rimuovere un messaggio da una coda.

Questo estratto è estratto dall'applicazione di esempio Sfoglia (programma CSQ4BCA1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
#include "cmqc.h"
.
.
.
#define BUFFERLENGTH 80
.
.
.
int main(int argc, char *argv[] )
   {
   /*                                                  */
   /*     Variables for MQ calls                       */
   /*                                                  */
   MQHCONN Hconn ;            /* Connection handle     */
   MQLONG  CompCode;          /* Completion code       */
   MQLONG  Reason;            /* Qualifying reason     */
   MQHOBJ  Hobj;              /* Object handle         */
   MQMD    MsgDesc  = { MQMD_DEFAULT };
                              /* Message descriptor    */
   MQLONG  DataLength ;       /* Length of the message */
   MQCHAR  Buffer[BUFFERLENGTH+1];
                              /* Area for message data */
   MQGMO   GetMsgOpts = { MQGMO_DEFAULT };
                              /* Options which control */
                              /*  the MQGET call       */
   MQLONG  BufferLength = BUFFERLENGTH ;
                              /* Length of buffer      */
   .
   .
   .
   /*     No need to change the message descriptor     */
   /*     (MQMD) control block because initialization  */
   /*     default sets all the fields.                 */
   /*                                                  */
   /*     Initialize the get message options (MQGMO)   */
   /*     control block (the copy file initializes all */
   /*     the other fields).                           */
   /*                                                  */
   GetMsgOpts.Options = MQGMO_NO_WAIT      +
                        MQGMO_BROWSE_FIRST +
                        MQGMO_ACCEPT_TRUNCATED_MSG;
   /*                                                  */
   /* Get the first message.                           */
   /*   Test for the output of the call is carried out */
   /*   in the 'for' loop.                             */
   /*                                                  */
```

```
   MQGET(Hconn,
         Hobj,
         &MsgDesc,
         &GetMsgOpts,
         BufferLength,
         Buffer,
         &DataLength,
         &CompCode,
         &Reason);
```

```
   /*                                                          */
   /* Process the message and get the next message,            */
   /* until no messages remaining.                             */
   .
   /*     If the call fails for any other reason,              */
   /*     print an error message showing the completion        */
   /*     code and reason code.                                */
   /*                                                          */
   if ( (CompCode == MQCC_FAILED) &&
        (Reason   == MQRC_NO_MSG_AVAILABLE) )
      {
      .
      }
   else
      {
      sprintf(pBuff, MESSAGE_4_E,
              ERROR_IN_MQGET, CompCode, Reason);
      PrintLine(pBuff);
      RetCode = CSQ4_ERROR;
      }
      .
   } /* end of main */
```

Ricezione di un messaggio utilizzando l'opzione di attesa

Questo esempio dimostra come utilizzare l'opzione di attesa della chiamata MQGET.

Questo codice accetta messaggi troncati. Questo estratto viene estratto dall'applicazione di esempio Credit Check (programma CSQ4CCB5) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
.
MQLONG   Hconn;              /* Connection handle        */
MQHOBJ   Hobj_CheckQ;        /* Object handle            */
MQLONG   CompCode;           /* Completion code          */
MQLONG   Reason;             /* Qualifying reason        */
MQOD     ObjDesc    = {MQOD_DEFAULT};
                             /* Object descriptor        */
MQMD     MsgDesc    = {MQMD_DEFAULT};
                             /* Message descriptor       */
MQLONG   OpenOptions;
                             /* Control the MQOPEN call  */
MQGMO    GetMsgOpts = {MQGMO_DEFAULT};
                             /* Get Message Options      */
MQLONG   MsgBuffLen;         /* Length of message buffer */
CSQ4BCAQ MsgBuffer;          /* Message structure        */
MQLONG   DataLen;            /* Length of message        */
```

```
.
void main(void)
   {
   .
   /*                                                          */
   /* Initialize options and open the queue for input          */
   /*                                                          */
   .
      /*                                                       */
      /* Get and process messages                              */
      /*                                                       */
      GetMsgOpts.Options = MQGMO_WAIT +
                           MQGMO_ACCEPT_TRUNCATED_MSG +
                           MQGMO_SYNCPOINT;
      GetMsgOpts.WaitInterval = WAIT_INTERVAL;
```

```
      MsgBuffLen = sizeof(MsgBuffer);
      memcpy(MsgDesc.MsgId, MQMI_NONE,
             sizeof(MsgDesc.MsgId));
      memcpy(MsgDesc.CorrelId, MQCI_NONE,
             sizeof(MsgDesc.CorrelId));
      /*                                                  */
      /* Make the first MQGET call outside the loop       */
      /*                                                  */
      MQGET(Hconn,
            Hobj_CheckQ,
            &MsgDesc,
            &GetMsgOpts,
            MsgBuffLen,
            &MsgBuffer,
            &DataLen,
            &CompCode,
            &Reason);
      .
      .
      .
      /*                                                  */
      /* Test the output of the MQGET call.  If the call */
      /* failed, send an error message showing the       */
      /* completion code and reason code, unless the     */
      /* reason code is NO_MSG AVAILABLE.                */
      /*                                                  */
      if (Reason != MQRC_NO_MSG_AVAILABLE)
         {
         strncpy(TS_Operation, "MQGET", sizeof(TS_Operation));
         strncpy(TS_ObjName, ObjDesc.ObjectName,
                 MQ_Q_NAME_LENGTH);
         Record_Call_Error();
         }
         .
```

Ricezione di un messaggio utilizzando la segnalazione

La segnalazione è disponibile solo con IBM MQ for z/OS .

Questo esempio mostra come utilizzare la chiamata MQGET per impostare un segnale in modo da ricevere una notifica quando un messaggio adatto arriva su una coda. Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
.
.
.
get_set_signal()
{
     MQMD    MsgDesc;
     MQGMO   GetMsgOpts;
     MQLONG  CompCode;
     MQLONG  Reason;
     MQHCONN Hconn;
     MQHOBJ  Hobj;
     MQLONG  BufferLength;
     MQLONG  DataLength;
     char message_buffer[100];
     long  int q_ecb, work_ecb;
     short int signal_sw, endloop;
     long  int mask = 255;

     /*--------------------------*/
     /*  Set up GMO structure.    */
     /*--------------------------*/
     memset(&GetMsgOpts,'\0',sizeof(GetMsgOpts));
     memcpy(GetMsgOpts.StrucId, MQGMO_STRUC_ID,
            sizeof(GetMsgOpts.StrucId);
     GetMsgOpts.Version      = MQGMO_VERSION_1;
     GetMsgOpts.WaitInterval = 1000;
     GetMsgOpts.Options      = MQGMO_SET_SIGNAL +
                               MQGMO_BROWSE_FIRST;
     q_ecb              = 0;
     GetMsgOpts.Signal1      = &q_ecb;
     /*--------------------------*/
     /*  Set up MD structure.     */
     /*--------------------------*/
     memset(&MsgDesc,'\0',sizeof(MsgDesc));
     memcpy(MsgDesc.StrucId, MQMD_STRUC_ID,
            sizeof(MsgDesc.StrucId);
     MsgDesc.Version  = MQMD_VERSION_1;
     MsgDesc.Report   = MQRO_NONE;
     memcpy(MsgDesc.MsgId,MQMI_NONE,
```

```
        sizeof(MsgDesc.MsgId));
 memcpy(MsgDesc.CorrelId,MQCI_NONE,
        sizeof(MsgDesc.CorrelId));
```

```
/*-----------------------------------------------------*/
/*  Issue the MQGET call.                              */
/*-----------------------------------------------------*/
 BufferLength = sizeof(message_buffer);
 signal_sw   = 0;

 MQGET(Hconn, Hobj, &MsgDesc, &GetMsgOpts,
       BufferLength, message_buffer, &DataLength,
       &CompCode, &Reason);
/*-------------------------------------*/
/*  Check completion and reason codes. */
/*-------------------------------------*/
 switch (CompCode)
     {
      case (MQCC_OK):          /* Message retrieved  */
          break;
      case (MQCC_WARNING):
          switch (Reason)
          {
           case (MQRC_SIGNAL_REQUEST_ACCEPTED):
               signal_sw = 1;
               break;
           default:
               break;   /* Perform error processing  */
          }
          break;
      case (MQCC_FAILED):
          switch (Reason)
          {
           case (MQRC_Q_MGR_NOT_AVAILABLE):
           case (MQRC_CONNECTION_BROKEN):
           case (MQRC_Q_MGR_STOPPING):
               break;
           default:
               break;  /* Perform error processing.  */
          }
          break;
      default:
          break;       /* Perform error processing.  */
     }
/*-----------------------------------------------------*/
/* If the SET_SIGNAL was accepted, set up a loop to  */
/* check whether a message has arrived at one second */
/* intervals. The loop ends if a message arrives or  */
/* the wait interval specified in the MQGMO          */
/* structure has expired.                            */
/*                                                   */
/* If a message arrives on the queue, another MQGET  */
/* must be issued to retrieve the message. If other  */
/* MQM calls have been made in the intervening       */
/* period, this may necessitate reinitializing the   */
/* MQMD and MQGMO structures.                        */
/* In this code, no intervening calls                */
/* have been made, so the only change required to    */
/* the structures is to specify MQGMO_NO_WAIT,       */
/* since we now know the message is there.           */
/*                                                   */
/* This code uses the EXEC CICS DELAY command to     */
/* suspend the program for a second. A batch program */
/* may achieve the same effect by calling an         */
/* assembler language subroutine which issues a      */
/* z/OS STIMER macro.                                */
/*-----------------------------------------------------*/
```

```
if (signal_sw == 1)
   {
    endloop = 0;
    do
      {
       EXEC CICS DELAY FOR HOURS(0) MINUTES(0) SECONDS(1);
       work_ecb = q_ecb & mask;
       switch (work_ecb)
           {
            case (MQEC_MSG_ARRIVED):
```

```
                    endloop = 1;
                    mqgmo_options = MQGMO_NO_WAIT;
                    MQGET(Hconn, Hobj, &MsgDesc, &GetMsgOpts,
                          BufferLength, message_buffer,
                          &DataLength, &CompCode, &Reason);
                    if (CompCode != MQCC_OK)
                        ;          /* Perform error processing. */
                    break;
                case (MQEC_WAIT_INTERVAL_EXPIRED):
                case (MQEC_WAIT_CANCELED):
                    endloop = 1;
                    break;
                default:
                    break;
               }
          } while (endloop == 0);
       }
     return;
}
```

Richiesta di informazioni sugli attributi di un oggetto

Questo esempio dimostra come utilizzare la chiamata MQINQ per richiedere informazioni sugli attributi di una coda.

Questo estratto viene estratto dall'applicazione di esempio Attributi coda (programma CSQ4CCC1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
#include <cmqc.h>      /* MQ API header file       */
.
#define NUMBEROFSELECTORS  2

const MQHCONN Hconn = MQHC_DEF_HCONN;
.
static void InquireGetAndPut(char   *Message,
                             PMQHOBJ pHobj,
                             char   *Object)
```

```
   {
   /*      Declare local variables                        */
   /*                                                     */
   MQLONG  SelectorCount = NUMBEROFSELECTORS;
                                /* Number of selectors  */
   MQLONG  IntAttrCount  = NUMBEROFSELECTORS;
                                /* Number of int attrs  */
   MQLONG  CharAttrLength = 0;
                    /* Length of char attribute buffer  */
   MQCHAR *CharAttrs ;
                    /* Character attribute buffer       */
   MQLONG  SelectorsTable[NUMBEROFSELECTORS];
                                /* attribute selectors  */
   MQLONG  IntAttrsTable[NUMBEROFSELECTORS];
                                /* integer attributes   */
   MQLONG  CompCode;            /* Completion code      */
   MQLONG  Reason;              /* Qualifying reason    */
   /*                                                     */
   /*     Open the queue.  If successful, do the inquire */
   /*     call.                                          */
   /*                                                     */
      /*                                                  */
      /*   Initialize the variables for the inquire      */
      /*   call:                                         */
      /*    - Set SelectorsTable to the attributes whose */
      /*      status is                                  */
      /*       required                                  */
      /*    - All other variables are already set        */
      /*                                                  */
      SelectorsTable[0] = MQIA_INHIBIT_GET;
      SelectorsTable[1] = MQIA_INHIBIT_PUT;
      /*                                                  */
      /*   Issue the inquire call                        */
      /*     Test the output of the inquire call. If the */
      /*     call failed, display an error message       */
      /*     showing the completion code and reason code,*/
      /*     otherwise display the status of the         */
```

```
      /*     INHIBIT-GET and INHIBIT-PUT attributes      */
      /*                                                 */
      MQINQ(Hconn,
            *pHobj,
            SelectorCount,
            SelectorsTable,
            IntAttrCount,
            IntAttrsTable,
            CharAttrLength,
            CharAttrs,
            &CompCode,
            &Reason);
      if (CompCode != MQCC_OK)
         {
         sprintf(Message, MESSAGE_4_E,
                 ERROR_IN_MQINQ, CompCode, Reason);
         SetMsg(Message);
         }
      else
         {
           /* Process the changes */
         } /* end if CompCode */
```

Impostazione degli attributi di una coda

Questo esempio illustra come utilizzare la chiamata MQSET per modificare gli attributi di una coda.

Questo estratto viene estratto dall'applicazione di esempio Attributi coda (programma CSQ4CCC1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
#include <cmqc.h>      /* MQ API header file      */
⋮
#define NUMBEROFSELECTORS  2

const MQHCONN Hconn = MQHC_DEF_HCONN;

static void InhibitGetAndPut(char   *Message,
                             PMQHOBJ pHobj,
                             char   *Object)
   {
   /*                                                 */
   /*      Declare local variables                    */
   /*                                                 */
   MQLONG  SelectorCount = NUMBEROFSELECTORS;
                              /* Number of selectors  */
   MQLONG  IntAttrCount  = NUMBEROFSELECTORS;
                              /* Number of int attrs  */
   MQLONG  CharAttrLength = 0;
                    /* Length of char attribute buffer  */
   MQCHAR *CharAttrs ;
                    /* Character attribute buffer       */
   MQLONG  SelectorsTable[NUMBEROFSELECTORS];
                              /* attribute selectors  */
   MQLONG  IntAttrsTable[NUMBEROFSELECTORS];
                              /* integer attributes   */
   MQLONG  CompCode;          /* Completion code      */
   MQLONG  Reason;            /* Qualifying reason    */
   ⋮
   /*                                                 */
   /*     Open the queue.  If successful, do the      */
   /*     inquire call.                               */
   /*                                                 */
   ⋮
      /*                                               */
      /*   Initialize the variables for the set call:  */
      /*    - Set SelectorsTable to the attributes to be */
      /*      set                                      */
      /*    - Set IntAttrsTable to the required status */
      /*    - All other variables are already set      */
      /*                                               */
      SelectorsTable[0] = MQIA_INHIBIT_GET;
      SelectorsTable[1] = MQIA_INHIBIT_PUT;
      IntAttrsTable[0]  = MQQA_GET_INHIBITED;
      IntAttrsTable[1]  = MQQA_PUT_INHIBITED;
      ⋮
```

```
     /*                                                  */
     /*   Issue the set call.                            */
     /*     Test the output of the set call.  If the     */
     /*     call fails, display an error message         */
     /*     showing the completion code and reason       */
     /*     code; otherwise move INHIBITED to the        */
     /*     relevant screen map fields                   */
     /*                                                  */
     MQSET(Hconn,
           *pHobj,
           SelectorCount,
           SelectorsTable,
           IntAttrCount,
           IntAttrsTable,
           CharAttrLength,
           CharAttrs,
           &CompCode,
           &Reason);
     if (CompCode != MQCC_OK)
        {
        sprintf(Message, MESSAGE_4_E,
                ERROR_IN_MQSET, CompCode, Reason);
        SetMsg(Message);
        }
     else
        {
          /* Process the changes */
        } /* end if CompCode */
```

Richiamo delle informazioni di stato con MQSTAT

Questo esempio mostra come emettere un MQPUT asincrono e richiamare le informazioni sullo stato con MQSTAT.

Questo estratto viene estratto dall'applicazione di esempio Calling MQSTAT (programma amqsapt0) forniti con i sistemi IBM MQ for Windows . Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
/*********************************************************************/
/*                                                                   */
/* Program name: AMQSAPT0                                            */
/*                                                                   */
/* Description: Sample C program that asynchronously puts messages   */
/*              to a message queue (example using MQPUT & MQSTAT).   */
/*                                                                   */
/* Licensed Materials - Property of IBM                              */
/*                                                                   */
/* 63H9336                                                           */
/* (c) Copyright IBM Corp. 2006, 2024. All Rights Reserved.          */
/*                                                                   */
/* US Government Users Restricted Rights - Use, duplication or       */
/* disclosure restricted by GSA ADP Schedule Contract with           */
/* IBM Corp.                                                         */
/*                                                                   */
/*********************************************************************/
/*                                                                   */
/* Function:                                                         */
/*                                                                   */
/*   AMQSAPT0 is a sample C program to put messages on a message     */
/*   queue with asynchronous response option, querying the success   */
/*   of the put operations with MQSTAT.                              */
/*                                                                   */
/*      -- messages are sent to the queue named by the parameter     */
/*                                                                   */
/*      -- gets lines from StdIn, and adds each to target           */
/*         queue, taking each line of text as the content            */
/*         of a datagram message; the sample stops when a null       */
/*         line (or EOF) is read.                                    */
/*         New-line characters are removed.                          */
/*         If a line is longer than 99 characters it is broken up    */
/*         into 99-character pieces. Each piece becomes the          */
/*         content of a datagram message.                            */
/*         If the length of a line is a multiple of 99 plus 1, for   */
/*         example, 199, the last piece will only contain a          */
/*         new-line character so will terminate the input.           */
/*                                                                   */
```

```
/*      -- writes a message for each MQI reason other than         */
/*         MQRC_NONE; stops if there is a MQI completion code      */
/*         of MQCC_FAILED                                          */
/*                                                                 */
/*      -- summarizes the overall success of the put operations    */
/*         through a call to MQSTAT to query MQSTAT_TYPE_ASYNC_ERROR*/
/*                                                                 */
/*    Program logic:                                               */
/*         MQOPEN target queue for OUTPUT                          */
/*         while end of input file not reached,                    */
/*         .  read next line of text                               */
/*         .  MQPUT datagram message with text line as data        */
/*         MQCLOSE target queue                                    */
/*         MQSTAT connection                                       */
/*                                                                 */
/*                                                                 */
/*******************************************************************/
/*                                                                 */
/*   AMQSAPT0 has the following parameters                         */
/*       required:                                                 */
/*                 (1) The name of the target queue                */
/*       optional:                                                 */
/*                 (2) Queue manager name                          */
/*                 (3) The open options                            */
/*                 (4) The close options                           */
/*                 (5) The name of the target queue manager        */
/*                 (6) The name of the dynamic queue               */
/*                                                                 */
/*******************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
   /* includes for MQI */
#include <cmqc.h>

int main(int argc, char **argv)
{
  /*  Declare file and character for sample input                  */
  FILE *fp;

  /*   Declare MQI structures needed                               */
  MQOD     od = {MQOD_DEFAULT};    /* Object Descriptor            */
  MQMD     md = {MQMD_DEFAULT};    /* Message Descriptor           */
  MQPMO   pmo = {MQPMO_DEFAULT};   /* put message options          */
  MQSTS   sts = {MQSTS_DEFAULT};   /* status information           */
     /** note, sample uses defaults where it can **/
  MQHCONN  Hcon;                   /* connection handle            */
  MQHOBJ   Hobj;                   /* object handle                */
  MQLONG   O_options;              /* MQOPEN options               */
  MQLONG   C_options;              /* MQCLOSE options              */
  MQLONG   CompCode;               /* completion code              */
  MQLONG   OpenCode;               /* MQOPEN completion code       */
  MQLONG   Reason;                 /* reason code                  */
  MQLONG   CReason;                /* reason code for MQCONN       */
  MQLONG   messlen;                /* message length               */
  char     buffer[100];            /* message buffer               */
  char     QMName[50];             /* queue manager name           */

  printf("Sample AMQSAPT0 start\n");
  if (argc < 2)
  {
    printf("Required parameter missing - queue name\n");
    exit(99);
  }

  /*******************************************************************/
  /*                                                                 */
  /*   Connect to queue manager                                      */
  /*                                                                 */
  /*******************************************************************/
  QMName[0] = 0;    /* default */
  if (argc > 2)
    strcpy(QMName, argv[2]);
  MQCONN(QMName,                 /* queue manager                  */
         &Hcon,                  /* connection handle              */
         &Compcode,              /* completion code                */
         &Reason);               /* reason code                    */
  /* report reason and stop if it failed     */
  if (CompCode == MQCC_FAILED)
  {
    printf("MQCONN ended with reason code %d\n", CReason);
    exit( (int)CReason );
```

```
}

/*****************************************************************/
/*                                                               */
/*   Use parameter as the name of the target queue               */
/*                                                               */
/*****************************************************************/
strncpy(od.ObjectName, argv[1], (size_t)MQ_Q_NAME_LENGTH);
printf("target queue is %s\n", od.ObjectName);

if (argc > 5)
{
  strncpy(od.ObjectQMgrName, argv[5], (size_t) MQ_Q_MGR_NAME_LENGTH);
  printf("target queue manager is %s\n", od.ObjectQMgrName);
}

if (argc > 6)
{
  strncpy(od.DynamicQName, argv[6], (size_t) MQ_Q_NAME_LENGTH);
  printf("dynamic queue name is %s\n", od.DynamicQName);
}

/*****************************************************************/
/*                                                               */
/*   Open the target message queue for output                    */
/*                                                               */
/*****************************************************************/
if (argc > 3)
{
  O_options = atoi( argv[3] );
  printf("open  options are %d\n", O_options);
}
else
{
  O_options = MQOO_OUTPUT            /* open queue for output     */
            | MQOO_FAIL_IF_QUIESCING /* but not if MQM stopping   */
            ;                        /* = 0x2010 = 8208 decimal   */
}

MQOPEN(Hcon,                      /* connection handle            */
       &od,                       /* object descriptor for queue  */
       O_options,                 /* open options                 */
       &Hobj,                     /* object handle                */
       &OpenCode,                 /* MQOPEN completion code       */
       &Reason);                  /* reason code                  */

/* report reason, if any; stop if failed      */
if (Reason != MQRC_NONE)
{
  printf("MQOPEN ended with reason code %d\n", Reason);
}

if (OpenCode == MQCC_FAILED)
{
  printf("unable to open queue for output\n");
}

/*****************************************************************/
/*                                                               */
/*   Read lines from the file and put them to the message queue  */
/*   Loop until null line or end of file, or there is a failure  */
/*                                                               */
/*****************************************************************/
CompCode = OpenCode;        /* use MQOPEN result for initial test */
fp = stdin;

memcpy(md.Format,           /* character string format           */
       MQFMT_STRING, (size_t)MQ_FORMAT_LENGTH);

/*****************************************************************/
/* These options specify that put operation should occur         */
/* asynchronously and the application will check the success     */
/* using MQSTAT at a later time.                                 */
/*****************************************************************/
md.Persistence = MQPER_NOT_PERSISTENT;
pmo.Options |= MQPMO_ASYNC_RESPONSE;

/*****************************************************************/
/* These options cause the MsgId and CorrelId to be replaced, so */
/* that there is no need to reset them before each MQPUT         */
/*****************************************************************/
pmo.Options |= MQPMO_NEW_MSG_ID;
```

```
pmo.Options |= MQPMO_NEW_CORREL_ID;

while (CompCode != MQCC_FAILED)
{
  if (fgets(buffer, sizeof(buffer), fp) != NULL)
  {
    messlen = (MQLONG)strlen(buffer); /* length without null      */
    if (buffer[messlen-1] == '\n')  /* last char is a new-line    */
    {
      buffer[messlen-1]  = '\0';    /* replace new-line with null */
      --messlen;                    /* reduce buffer length       */
    }
  }
  else messlen = 0;        /* treat EOF same as null line         */

  /*****************************************************************/
  /*                                                               */
  /*   Put each buffer to the message queue                        */
  /*                                                               */
  /*****************************************************************/
  if (messlen > 0)
  {
    MQPUT(Hcon,               /* connection handle               */
          Hobj,               /* object handle                   */
          &md,                /* message descriptor              */
          &pmo,               /* default options (datagram)      */
          messlen,            /* message length                  */
          buffer,             /* message buffer                  */
          &CompCode,          /* completion code                 */
          &Reason);           /* reason code                     */

    /* report reason, if any */
    if (Reason != MQRC_NONE)
    {
      printf("MQPUT ended with reason code %d\n", Reason);
    }
  }
  else   /* satisfy end condition when empty line is read */
    CompCode = MQCC_FAILED;
}

/*****************************************************************/
/*                                                               */
/*   Close the target queue (if it was opened)                   */
/*                                                               */
/*****************************************************************/
if (OpenCode != MQCC_FAILED)
{
  if (argc > 4)
  {
    C_options = atoi( argv[4] );
    printf("close options are %d\n", C_options);
  }
  else
  {
    C_options = MQCO_NONE;        /* no close options            */
  }

  MQCLOSE(Hcon,                   /* connection handle           */
          &Hobj,                  /* object handle               */
          C_options,
          &CompCode,              /* completion code             */
          &Reason);               /* reason code                 */

  /* report reason, if any     */
  if (Reason != MQRC_NONE)
  {
    printf("MQCLOSE ended with reason code %d\n", Reason);
  }
}

/*****************************************************************/
/*                                                               */
/*   Query how many asynchronous puts succeeded                  */
/*                                                               */
/*****************************************************************/
MQSTAT(&Hcon,                   /* connection handle           */
       MQSTAT_TYPE_ASYNC_ERROR, /* status type                 */
       &Sts,                    /* MQSTS structure             */
       &CompCode,               /* completion code             */
       &Reason);                /* reason code                 */
```

```
   /* report reason, if any     */
   if (Reason != MQRC_NONE)
   {
     printf("MQSTAT ended with reason code %d\n", Reason);
   }
   else
   {
     /* Display results */
     printf("Succeeded putting %d messages\n",
            sts.PutSuccessCount);
     printf("%d messages were put with a warning\n",
            sts.PutWarningCount);
     printf("Failed to put %d messages\n",
            sts.PutFailureCount);

     if(sts.CompCode == MQCC_WARNING)
     {
       printf("The first warning that occurred had reason code %d\n",
              sts.Reason);
     }
     else if(sts.CompCode == MQCC_FAILED)
     {
       printf("The first error that occurred had reason code %d\n",
              sts.Reason);
     }
   }

   /*****************************************************************/
   /*                                                               */
   /*   Disconnect from MQM if not already connected                */
   /*                                                               */
   /*****************************************************************/
   if (CReason != MQRC_ALREADY_CONNECTED)
   {
     MQDISC(&Hcon,                  /* connection handle            */
            &CompCode,              /* completion code              */
            &Reason);               /* reason code                  */

     /* report reason, if any     */
     if (Reason != MQRC_NONE)
     {
       printf("MQDISC ended with reason code %d\n", Reason);
     }
   }

   /*****************************************************************/
   /*                                                               */
   /* END OF AMQSAPT0                                               */
   /*                                                               */
   /*****************************************************************/
   printf("Sample AMQSAPT0 end\n");
   return(0);
 }
```

Esempi COBOL

Questa raccolta di argomenti è tratta dalle applicazioni di esempio IBM MQ for z/OS . Sono applicabili a tutte le piattaforme, tranne dove indicato.

Connessione a un gestore code

Questo esempio illustra come utilizzare la chiamata MQCONN per connettere un programma a un gestore code in batch z/OS .

Questo estratto viene estratto dall'applicazione di esempio Sfoglia (programma CSQ4BVA1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
* -------------------------------------------------------*
WORKING-STORAGE SECTION.
* -------------------------------------------------------*
*    W02 - Data fields derived from the PARM field
 01  W02-MQM                    PIC X(48) VALUE SPACES.
*    W03 - MQM API fields
 01  W03-HCONN                  PIC S9(9) BINARY.
 01  W03-COMPCODE               PIC S9(9) BINARY.
```

```
 01  W03-REASON                 PIC S9(9) BINARY.
*
*    MQV contains constants (for filling in the control
*    blocks)
*    and return codes (for testing the result of a call)
*
 01  W05-MQM-CONSTANTS.
 COPY CMQV SUPPRESS.
 ⋮
*    Separate into the relevant fields any data passed
*    in the PARM statement
*
     UNSTRING PARM-STRING DELIMITED BY ALL ','
                             INTO W02-MQM
                                  W02-OBJECT.
     ⋮
*    Connect to the specified queue manager.
*
     CALL 'MQCONN' USING W02-MQM
                         W03-HCONN
                         W03-COMPCODE
                         W03-REASON.
*
*    Test the output of the connect call.  If the call
*    fails, print an error message showing the
*    completion code and reason code.
*
     IF (W03-COMPCODE NOT = MQCC-OK) THEN
     ⋮
     END-IF.
     ⋮
```

Disconnessione da un gestore code

Questo esempio illustra come utilizzare la chiamata MQDISC per disconnettere un programma da un gestore code nel batch z/OS .

Le variabili utilizzate in questo estratto di codice sono quelle impostate in “Connessione a un gestore code” a pagina 23. Questo estratto viene estratto dall'applicazione di esempio Sfoglia (programma CSQ4BVA1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
⋮
*
* Disconnect from the queue manager
*
     CALL 'MQDISC' USING W03-HCONN
                         W03-COMPCODE
                         W03-REASON.
*
*    Test the output of the disconnect call.  If the
*    call fails, print an error message showing the
*    completion code and reason code.
*
     IF (W03-COMPCODE NOT = MQCC-OK) THEN
⋮
           END-IF.
⋮
```

Creazione di una coda dinamica

Questo esempio illustra come utilizzare la chiamata MQOPEN per creare una coda dinamica.

Questo estratto viene estratto dall'applicazione di esempio Credit Check (programma CSQ4CVB1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
⋮
* -------------------------------------------------------*
WORKING-STORAGE SECTION.
* -------------------------------------------------------*
*
*    W02 - Queues processed in this program
*
```

```
 01  W02-MODEL-QNAME        PIC X(48) VALUE
     'CSQ4SAMP.B1.MODEL                          '.
 01  W02-NAME-PREFIX        PIC X(48) VALUE
     'CSQ4SAMP.B1.*                              '.
 01  W02-TEMPORARY-Q         PIC X(48).
*
*    W03 - MQM API fields
*
 01  W03-HCONN       PIC S9(9) BINARY VALUE ZERO.
 01  W03-OPTIONS     PIC S9(9) BINARY.
 01  W03-HOBJ        PIC S9(9) BINARY.
 01  W03-COMPCODE    PIC S9(9) BINARY.
 01  W03-REASON      PIC S9(9) BINARY.
*
*    API control blocks
*
 01  MQM-OBJECT-DESCRIPTOR.
     COPY CMQODV.
*
*    CMQV contains constants (for setting or testing
*    field values) and return codes (for testing the
*    result of a call)
*
 01  MQM-CONSTANTS.
 COPY CMQV SUPPRESS.
* ------------------------------------------------------------*
 PROCEDURE DIVISION.
* ------------------------------------------------------------*
⋮
* ------------------------------------------------------------*
 OPEN-TEMP-RESPONSE-QUEUE SECTION.
* ------------------------------------------------------------*
```

```
*
*  This section creates a temporary dynamic queue
*  using a model queue
*
* ------------------------------------------------------------*
*
* Change three fields in the Object Descriptor (MQOD)
* control block.  (MQODV initializes the other fields)
*
     MOVE MQOT-Q           TO MQOD-OBJECTTYPE.
     MOVE W02-MODEL-QNAME  TO MQOD-OBJECTNAME.
     MOVE W02-NAME-PREFIX  TO MQOD-DYNAMICQNAME.
*
     COMPUTE W03-OPTIONS = MQOO-INPUT-EXCLUSIVE.
*
     CALL 'MQOPEN' USING W03-HCONN
                         MQOD
                         W03-OPTIONS
                         W03-HOBJ-MODEL
                         W03-COMPCODE
                         W03-REASON.
*
     IF W03-COMPCODE NOT = MQCC-OK
          MOVE 'MQOPEN'      TO M01-MSG4-OPERATION
          MOVE W03-COMPCODE  TO M01-MSG4-COMPCODE
          MOVE W03-REASON    TO M01-MSG4-REASON
          MOVE M01-MESSAGE-4 TO M00-MESSAGE
     ELSE
          MOVE MQOD-OBJECTNAME TO W02-TEMPORARY-Q
     END-IF.
*
 OPEN-TEMP-RESPONSE-QUEUE-EXIT.
*
*    Return to performing section.
*
     EXIT.
     EJECT
*
```

Apertura di una coda esistente

Questo esempio illustra come utilizzare la chiamata MQOPEN per aprire una coda esistente.

Questo estratto viene estratto dall'applicazione di esempio Sfoglia (programma CSQ4BVA1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
⋮
* ------------------------------------------------------*
 WORKING-STORAGE SECTION.
* ------------------------------------------------------*
*
*    W01 - Fields derived from the command area input
*
 01  W01-OBJECT                 PIC X(48).
*
*    W02 - MQM API fields
*
 01  W02-HCONN        PIC S9(9) BINARY VALUE ZERO.
 01  W02-OPTIONS      PIC S9(9) BINARY.
 01  W02-HOBJ         PIC S9(9) BINARY.
 01  W02-COMPCODE     PIC S9(9) BINARY.
 01  W02-REASON       PIC S9(9) BINARY.
*
*    CMQODV defines the object descriptor (MQOD)
*
 01  MQM-OBJECT-DESCRIPTOR.
     COPY CMQODV.
*
* CMQV contains constants (for setting or testing
* field values) and return codes (for testing the
* result of a call)
*
 01  MQM-CONSTANTS.
 COPY CMQV SUPPRESS.
* ------------------------------------------------------*
 E-OPEN-QUEUE SECTION.
* ------------------------------------------------------*
*                                                       *
* This section opens the queue                          *
*
*    Initialize the Object Descriptor (MQOD) control
*    block
*    (The copy file initializes the remaining fields.)
*
     MOVE MQOT-Q         TO MQOD-OBJECTTYPE.
     MOVE W01-OBJECT     TO MQOD-OBJECTNAME.
*
*    Initialize W02-OPTIONS to open the queue for both
*    inquiring about and setting attributes
*
     COMPUTE W02-OPTIONS = MQOO-INQUIRE + MQOO-SET.
```

```
*
*    Open the queue
*
     CALL 'MQOPEN' USING W02-HCONN
                         MQOD
                         W02-OPTIONS
                         W02-HOBJ
                         W02-COMPCODE
                         W02-REASON.
*
*    Test the output from the open
*
*    If the completion code is not OK, display a
*    separate error message for each of the following
*    errors:
*
*  Q-MGR-NOT-AVAILABLE - MQM is not available
*  CONNECTION-BROKEN   - MQM is no longer connected to CICS
*  UNKNOWN-OBJECT-NAME - The queue does not exist
*  NOT-AUTHORIZED      - The user is not authorized to open
*                        the queue
*
* For any other error, display an error message
* showing the completion and reason codes
*
  IF W02-COMPCODE NOT = MQCC-OK
     EVALUATE TRUE
```

```
*
       WHEN W02-REASON = MQRC-Q-MGR-NOT-AVAILABLE
            MOVE M01-MESSAGE-6 TO M00-MESSAGE
*
       WHEN W02-REASON = MQRC-CONNECTION-BROKEN
            MOVE M01-MESSAGE-6 TO M00-MESSAGE
*
       WHEN W02-REASON = MQRC-UNKNOWN-OBJECT-NAME
            MOVE M01-MESSAGE-2 TO M00-MESSAGE
*
       WHEN W02-REASON = MQRC-NOT-AUTHORIZED
            MOVE M01-MESSAGE-3 TO M00-MESSAGE
*
       WHEN OTHER
            MOVE 'MQOPEN'      TO M01-MSG4-OPERATION
            MOVE W02-COMPCODE  TO M01-MSG4-COMPCODE
            MOVE W02-REASON    TO M01-MSG4-REASON
            MOVE M01-MESSAGE-4 TO M00-MESSAGE
        END-EVALUATE
     END-IF.
 E-EXIT.
*
*    Return to performing section
*
     EXIT.
     EJECT
```

Chiusura di una coda

Questo esempio mostra come utilizzare la chiamata MQCLOSE.

Le variabili utilizzate in questo estratto di codice sono quelle impostate in “Connessione a un gestore code” a pagina 23. Questo estratto viene estratto dall'applicazione di esempio Sfoglia (programma CSQ4BVA1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
⋮
*
*    Close the queue
*
     MOVE MQCO-NONE TO W03-OPTIONS.
*
     CALL 'MQCLOSE' USING W03-HCONN
                          W03-HOBJ
                          W03-OPTIONS
                          W03-COMPCODE
                          W03-REASON.
*
*    Test the output of the MQCLOSE call.  If the call
*    fails, print an error message showing the
*    completion code and reason code.
*
     IF (W03-COMPCODE NOT = MQCC-OK) THEN
        MOVE 'CLOSE'       TO W04-MSG4-TYPE
        MOVE W03-COMPCODE  TO W04-MSG4-COMPCODE
        MOVE W03-REASON    TO W04-MSG4-REASON
        MOVE W04-MESSAGE-4 TO W00-PRINT-DATA
        PERFORM PRINT-LINE
        MOVE W06-CSQ4-ERROR TO W00-RETURN-CODE
     END-IF.
*
```

Inserimento di un messaggio utilizzando MQPUT

Questo esempio mostra come utilizzare la chiamata MQPUT utilizzando il contesto.

Questo estratto viene estratto dall'applicazione di esempio Credit Check (programma CSQ4CVB1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
⋮
* ------------------------------------------------------*
 WORKING-STORAGE SECTION.
* ------------------------------------------------------*
*
```

```
*    W02 - Queues processed in this program
*
 01  W02-TEMPORARY-Q            PIC X(48).
*
*    W03 - MQM API fields
*
 01  W03-HCONN           PIC S9(9) BINARY VALUE ZERO.
 01  W03-HOBJ-INQUIRY    PIC S9(9) BINARY.
 01  W03-OPTIONS         PIC S9(9) BINARY.
 01  W03-BUFFLEN         PIC S9(9) BINARY.
 01  W03-COMPCODE        PIC S9(9) BINARY.
 01  W03-REASON          PIC S9(9) BINARY.
*
 01  W03-PUT-BUFFER.
*
     05 W03-CSQ4BIIM.
     COPY CSQ4VB1.
*
*    API control blocks
*
 01  MQM-MESSAGE-DESCRIPTOR.
     COPY CMQMDV.
 01  MQM-PUT-MESSAGE-OPTIONS.
     COPY CMQPMOV.
*
*    MQV contains constants (for filling in the
*    control blocks) and return codes (for testing
*    the result of a call).
*
 01  MQM-CONSTANTS.
     COPY CMQV SUPPRESS.
* -------------------------------------------------------*
 PROCEDURE DIVISION.
* -------------------------------------------------------*
⋮
*    Open queue and build message.
⋮
```

```
*
* Set the message descriptor and put-message options to
* the values required to create the message.
* Set the length of the message.
*
  MOVE MQMT-REQUEST         TO MQMD-MSGTYPE.
  MOVE MQCI-NONE            TO MQMD-CORRELID.
  MOVE MQMI-NONE            TO MQMD-MSGID.
  MOVE W02-TEMPORARY-Q      TO MQMD-REPLYTOQ.
  MOVE SPACES               TO MQMD-REPLYTOQMGR.
  MOVE 5                    TO MQMD-PRIORITY.
  MOVE MQPER-NOT-PERSISTENT TO MQMD-PERSISTENCE.
  COMPUTE MQPMO-OPTIONS     =  MQPMO-NO-SYNCPOINT +
                               MQPMO-DEFAULT-CONTEXT.
  MOVE LENGTH OF CSQ4BIIM-MSG TO W03-BUFFLEN.
*
     CALL 'MQPUT' USING W03-HCONN
                        W03-HOBJ-INQUIRY
                        MQMD
                        MQPMO
                        W03-BUFFLEN
                        W03-PUT-BUFFER
                        W03-COMPCODE
                        W03-REASON.
     IF W03-COMPCODE NOT = MQCC-OK
     ⋮
     END-IF.
```

Inserimento di un messaggio utilizzando MQPUT1

Questo esempio illustra come utilizzare il richiamo MQPUT1 .

Questo estratto viene estratto dall'applicazione di esempio Controllo credito (programma CSQ4CVB5) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
⋮
* -------------------------------------------------------*
```

```
 WORKING-STORAGE SECTION.
* -------------------------------------------------------*
*
*    W03 - MQM API fields
*
 01  W03-HCONN          PIC S9(9) BINARY VALUE ZERO.
 01  W03-OPTIONS        PIC S9(9) BINARY.
 01  W03-COMPCODE       PIC S9(9) BINARY.
 01  W03-REASON         PIC S9(9) BINARY.
 01  W03-BUFFLEN        PIC S9(9) BINARY.
*
 01  W03-PUT-BUFFER.
     05 W03-CSQ4BQRM.
     COPY CSQ4VB4.
```

```
*
*    API control blocks
*
 01  MQM-OBJECT-DESCRIPTOR.
     COPY CMQODV.
 01  MQM-MESSAGE-DESCRIPTOR.
     COPY CMQMDV.
 01  MQM-PUT-MESSAGE-OPTIONS.
     COPY CMQPMOV.
*
* CMQV contains constants (for filling in the
* control blocks) and return codes (for testing
* the result of a call).
*
 01  MQM-MQV.
 COPY CMQV SUPPRESS.
* -------------------------------------------------------*
 PROCEDURE DIVISION.
* -------------------------------------------------------*
⋮
*    Get the request message.
⋮
* -------------------------------------------------------*
 PROCESS-QUERY SECTION.
* -------------------------------------------------------*
⋮
*    Build the reply message.
⋮
*
* Set the object descriptor, message descriptor and
* put-message options to the values required to create
* the message.
* Set the length of the message.
*
  MOVE MQMD-REPLYTOQ    TO MQOD-OBJECTNAME.
  MOVE MQMD-REPLYTOQMGR TO MQOD-OBJECTQMGRNAME.
  MOVE MQMT-REPLY       TO MQMD-MSGTYPE.
  MOVE SPACES           TO MQMD-REPLYTOQ.
  MOVE SPACES           TO MQMD-REPLYTOQMGR.
  MOVE LOW-VALUES       TO MQMD-MSGID.
  COMPUTE MQPMO-OPTIONS = MQPMO-SYNCPOINT +
                          MQPMO-PASS-IDENTITY-CONTEXT.
  MOVE W03-HOBJ-CHECKQ  TO MQPMO-CONTEXT.
  MOVE LENGTH OF CSQ4BQRM-MSG TO W03-BUFFLEN.
*
     CALL 'MQPUT1' USING W03-HCONN
                         MQOD
                         MQMD
                         MQPMO
                         W03-BUFFLEN
                         W03-PUT-BUFFER
                         W03-COMPCODE
                         W03-REASON.
     IF W03-COMPCODE NOT = MQCC-OK
         MOVE 'MQPUT1'          TO M02-OPERATION
         MOVE MQOD-OBJECTNAME   TO M02-OBJECTNAME
         PERFORM RECORD-CALL-ERROR
         PERFORM FORWARD-MSG-TO-DLQ
     END-IF.
*
```

acquisizione di un messaggio

Questo esempio mostra come utilizzare la chiamata MQGET per rimuovere un messaggio da una coda.

Questo estratto viene estratto dall'applicazione di esempio Credit Check (programma CSQ4CVB1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
⋮
* ------------------------------------------------------*
 WORKING-STORAGE SECTION.
* ------------------------------------------------------*
*
*    W03 - MQM API fields
*
 01  W03-HCONN            PIC S9(9) BINARY VALUE ZERO.
 01  W03-HOBJ-RESPONSE    PIC S9(9) BINARY.
 01  W03-OPTIONS          PIC S9(9) BINARY.
 01  W03-BUFFLEN          PIC S9(9) BINARY.
 01  W03-DATALEN          PIC S9(9) BINARY.
 01  W03-COMPCODE         PIC S9(9) BINARY.
 01  W03-REASON           PIC S9(9) BINARY.
*
 01  W03-GET-BUFFER.
     05 W03-CSQ4BAM.
     COPY CSQ4VB2.
*
*    API control blocks
*
 01  MQM-MESSAGE-DESCRIPTOR.
     COPY CMQMDV.
 01  MQM-GET-MESSAGE-OPTIONS.
     COPY CMQGMOV.
*
*    MQV contains constants (for filling in the
*    control blocks) and return codes (for testing
*    the result of a call).
*
 01  MQM-CONSTANTS.
     COPY CMQV SUPPRESS.
* ------------------------------------------------------*
 A-MAIN SECTION.
* ------------------------------------------------------*
⋮
*    Open response queue.
⋮
* ------------------------------------------------------*
 PROCESS-RESPONSE-SCREEN SECTION.
* ------------------------------------------------------*
*                                                       *
*  This section gets a message from the response queue.  *
*                                                       *
*  When a correct response is received, it is           *
*  transferred to the map for display; otherwise        *
*  an error message is built.                           *
*                                                       *
* ------------------------------------------------------*
```

```
*
*    Set get-message options
*
  COMPUTE MQGMO-OPTIONS = MQGMO-SYNCPOINT +
                          MQGMO-ACCEPT-TRUNCATED-MSG +
                          MQGMO-NO-WAIT.
*
* Set msgid and correlid in MQMD to nulls so that any
* message will qualify.
* Set length to available buffer length.
*
     MOVE MQMI-NONE TO MQMD-MSGID.
     MOVE MQCI-NONE TO MQMD-CORRELID.
     MOVE LENGTH OF W03-GET-BUFFER TO W03-BUFFLEN.
*
     CALL 'MQGET' USING W03-HCONN
                        W03-HOBJ-RESPONSE
                        MQMD
                        MQGMO
                        W03-BUFFLEN
                        W03-GET-BUFFER
                        W03-DATALEN
                        W03-COMPCODE
```

```
                        W03-REASON.
     EVALUATE TRUE
         WHEN W03-COMPCODE NOT = MQCC-FAILED
         ⋮
*            Process the message
             ⋮
         WHEN (W03-COMPCODE = MQCC-FAILED AND
               W03-REASON = MQRC-NO-MSG-AVAILABLE)
                 MOVE M01-MESSAGE-9 TO M00-MESSAGE
                 PERFORM CLEAR-RESPONSE-SCREEN
*
         WHEN OTHER
             MOVE 'MQGET  '     TO M01-MSG4-OPERATION
             MOVE W03-COMPCODE  TO M01-MSG4-COMPCODE
             MOVE W03-REASON    TO M01-MSG4-REASON
             MOVE M01-MESSAGE-4 TO M00-MESSAGE
             PERFORM CLEAR-RESPONSE-SCREEN
     END-EVALUATE.
```

Ricezione di un messaggio utilizzando l'opzione di attesa

Questo esempio mostra come utilizzare la chiamata MQGET con l'opzione di attesa e accettare i messaggi troncati.

Questo estratto viene estratto dall'applicazione di esempio Controllo credito (programma CSQ4CVB5) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
⋮
* -------------------------------------------------------*
 WORKING-STORAGE SECTION.
* -------------------------------------------------------*
*
*    W00 - General work fields
*
 01  W00-WAIT-INTERVAL   PIC S9(09) BINARY VALUE 30000.
*
*    W03 - MQM API fields
*
 01  W03-HCONN           PIC S9(9) BINARY VALUE ZERO.
 01  W03-OPTIONS         PIC S9(9) BINARY.
 01  W03-HOBJ-CHECKQ     PIC S9(9) BINARY.
 01  W03-COMPCODE        PIC S9(9) BINARY.
 01  W03-REASON          PIC S9(9) BINARY.
 01  W03-DATALEN         PIC S9(9) BINARY.
 01  W03-BUFFLEN         PIC S9(9) BINARY.
*
 01  W03-MSG-BUFFER.
     05 W03-CSQ4BCAQ.
     COPY CSQ4VB3.
*
*    API control blocks
*
 01  MQM-MESSAGE-DESCRIPTOR.
     COPY CMQMDV.
 01  MQM-GET-MESSAGE-OPTIONS.
     COPY CMQGMOV.
*
*    CMQV contains constants (for filling in the
*    control blocks) and return codes (for testing
*    the result of a call).
*
 01  MQM-MQV.
 COPY CMQV SUPPRESS.
* -------------------------------------------------------*
 PROCEDURE DIVISION.
* -------------------------------------------------------*
⋮
*    Open input queue.
⋮
```

```
*
*    Get and process messages.
*
  COMPUTE MQGMO-OPTIONS = MQGMO-WAIT +
                          MQGMO-ACCEPT-TRUNCATED-MSG +
```

```
                          MQGMO-SYNCPOINT.
  MOVE LENGTH OF W03-MSG-BUFFER TO W03-BUFFLEN.
  MOVE W00-WAIT-INTERVAL TO MQGMO-WAITINTERVAL.
  MOVE MQMI-NONE TO MQMD-MSGID.
  MOVE MQCI-NONE TO MQMD-CORRELID.
*
*    Make the first MQGET call outside the loop.
*
     CALL 'MQGET' USING W03-HCONN
                        W03-HOBJ-CHECKQ
                        MQMD
                        MQGMO
                        W03-BUFFLEN
                        W03-MSG-BUFFER
                        W03-DATALEN
                        W03-COMPCODE
                        W03-REASON.
*
*    Test the output of the MQGET call using the
*    PERFORM loop that follows.
*
*    Perform whilst no failure occurs
*      - process this message
*      - reset the call parameters
*      - get another message
*    End-perform
*

*
*    Test the output of the MQGET call.  If the call
*    fails, send an error message showing the
*    completion code and reason code, unless the
*    completion code is NO-MSG-AVAILABLE.
*
     IF (W03-COMPCODE NOT = MQCC-FAILED) OR
        (W03-REASON NOT = MQRC-NO-MSG-AVAILABLE)
         MOVE 'MQGET '          TO M02-OPERATION
         MOVE MQOD-OBJECTNAME   TO M02-OBJECTNAME
              PERFORM RECORD-CALL-ERROR
     END-IF.
⋮
```

Ricezione di un messaggio utilizzando la segnalazione

Questo esempio dimostra come utilizzare la chiamata MQGET con la segnalazione. Questo estratto viene estratto dall'applicazione di esempio Credit Check (programma CSQ4CVB2) fornita con IBM MQ for z/OS.

La segnalazione è disponibile solo con IBM MQ for z/OS .

```


⋮
* -------------------------------------------------------*
 WORKING-STORAGE SECTION.
* -------------------------------------------------------*
*
*    W00 - General work fields
 ⋮
 01  W00-WAIT-INTERVAL    PIC S9(09) BINARY VALUE 30000.
*
*    W03 - MQM API fields
*
 01  W03-HCONN            PIC S9(9) BINARY VALUE ZERO.
 01  W03-HOBJ-REPLYQ      PIC S9(9) BINARY.
 01  W03-COMPCODE         PIC S9(9) BINARY.
 01  W03-REASON           PIC S9(9) BINARY.
 01  W03-DATALEN          PIC S9(9) BINARY.
 01  W03-BUFFLEN          PIC S9(9) BINARY.
 ⋮
 01  W03-GET-BUFFER.
     05 W03-CSQ4BQRM.
     COPY CSQ4VB4.
*
     05 W03-CSQ4BIIM REDEFINES W03-CSQ4BQRM.
     COPY CSQ4VB1.
*
     05 W03-CSQ4BPGM REDEFINES W03-CSQ4BIIM.
     COPY CSQ4VB5.
     ⋮
```

```
*    API control blocks
*
 01  MQM-MESSAGE-DESCRIPTOR.
     COPY CMQMDV.
 01  MQM-GET-MESSAGE-OPTIONS.
     COPY CMQGMOV.
     ⋮
*    MQV contains constants (for filling in the
*    control blocks) and return codes (for testing
*    the result of a call).
*
 01  MQM-MQV.
 COPY CMQV SUPPRESS.
* ------------------------------------------------------*
 LINKAGE SECTION.
* ------------------------------------------------------*
 01  L01-ECB-ADDR-LIST.
     05  L01-ECB-ADDR1        POINTER.
     05  L01-ECB-ADDR2        POINTER.
```

```
*
 01  L02-ECBS.
     05  L02-INQUIRY-ECB1     PIC S9(09) BINARY.
     05  L02-REPLY-ECB2       PIC S9(09) BINARY.
 01  REDEFINES L02-ECBS.
     05                       PIC  X(02).
     05  L02-INQUIRY-ECB1-CC  PIC S9(04) BINARY.
     05                       PIC  X(02).
     05  L02-REPLY-ECB2-CC    PIC S9(04) BINARY.
*
* ------------------------------------------------------*
 PROCEDURE DIVISION.
* ------------------------------------------------------*
⋮
* Initialize variables, open queues, set signal on
* inquiry queue.
⋮
* ------------------------------------------------------*
 PROCESS-SIGNAL-ACCEPTED SECTION.
* ------------------------------------------------------*
*  This section gets a message with signal.  If a       *
*  message is received, process it.  If the signal      *
*  is set or is already set, the program goes into      *
*  an operating system wait.                            *
*  Otherwise an error is reported and call error set.   *
* ------------------------------------------------------*
*
  PERFORM REPLYQ-GETSIGNAL.
*
  EVALUATE TRUE
      WHEN (W03-COMPCODE = MQCC-OK AND
              W03-REASON = MQRC-NONE)
          PERFORM PROCESS-REPLYQ-MESSAGE
*
      WHEN (W03-COMPCODE = MQCC-WARNING AND
              W03-REASON = MQRC-SIGNAL-REQUEST-ACCEPTED)
           OR
           (W03-COMPCODE = MQCC-FAILED AND
              W03-REASON = MQRC-SIGNAL-OUTSTANDING)
          PERFORM EXTERNAL-WAIT
*
      WHEN OTHER
          MOVE 'MQGET SIGNAL'  TO M02-OPERATION
          MOVE MQOD-OBJECTNAME TO M02-OBJECTNAME
          PERFORM RECORD-CALL-ERROR
          MOVE W06-CALL-ERROR  TO W06-CALL-STATUS
  END-EVALUATE.
*
 PROCESS-SIGNAL-ACCEPTED-EXIT.
*    Return to performing section
     EXIT.
     EJECT
*
```

```
* ------------------------------------------------------*
 EXTERNAL-WAIT SECTION.
* ------------------------------------------------------*
*  This section performs an external CICS wait on two   *
```

```
*  ECBs until at least one is posted.  It then calls     *
*  the sections to handle the posted ECB.                *
* -------------------------------------------------------*
     EXEC CICS WAIT EXTERNAL
         ECBLIST(W04-ECB-ADDR-LIST-PTR)
         NUMEVENTS(2)
     END-EXEC.
*
* At least one ECB must have been posted to get to this
* point. Test which ECB has been posted and perform
* the appropriate section.
*
     IF L02-INQUIRY-ECB1 NOT = 0
         PERFORM TEST-INQUIRYQ-ECB
     ELSE
         PERFORM TEST-REPLYQ-ECB
     END-IF.
*
 EXTERNAL-WAIT-EXIT.
*
*    Return to performing section.
*
     EXIT.
     EJECT
     ⋮
* -------------------------------------------------------*
 REPLYQ-GETSIGNAL SECTION.
* -------------------------------------------------------*
*                                                        *
* This section performs an MQGET call (in syncpoint with *
* signal) on the reply queue.  The signal field in the   *
* MQGMO is set to the address of the ECB.                *
* Response handling is done by the performing section.   *
*                                                        *
* -------------------------------------------------------*
*
     COMPUTE MQGMO-OPTIONS         =  MQGMO-SYNCPOINT +
                                      MQGMO-SET-SIGNAL.
     MOVE W00-WAIT-INTERVAL        TO MQGMO-WAITINTERVAL.
     MOVE LENGTH OF W03-GET-BUFFER TO W03-BUFFLEN.
*
     MOVE ZEROS                    TO L02-REPLY-ECB2.
     SET MQGMO-SIGNAL1 TO ADDRESS OF L02-REPLY-ECB2.
```

```
*
*    Set msgid and correlid to nulls so that any message
*    will qualify.
*
     MOVE MQMI-NONE TO MQMD-MSGID.
     MOVE MQCI-NONE TO MQMD-CORRELID.
*
     CALL 'MQGET' USING W03-HCONN
                        W03-HOBJ-REPLYQ
                        MQMD
                        MQGMO
                        W03-BUFFLEN
                        W03-GET-BUFFER
                        W03-DATALEN
                        W03-COMPCODE
                        W03-REASON.
*
 REPLYQ-GETSIGNAL-EXIT.
*
*    Return to performing section.
*
     EXIT.
     EJECT
*
     ⋮
```

Richiesta di informazioni sugli attributi di un oggetto

Questo esempio dimostra come utilizzare la chiamata MQINQ per richiedere informazioni sugli attributi di una coda.

Questo estratto viene estratto dall'applicazione di esempio Attributi coda (programma CSQ4CVC1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme tranne z/OS).

```
⋮
* -------------------------------------------------------*
 WORKING-STORAGE SECTION.
* -------------------------------------------------------*
*
*    W02 - MQM API fields
*
 01  W02-SELECTORCOUNT    PIC S9(9) BINARY VALUE 2.
 01  W02-INTATTRCOUNT     PIC S9(9) BINARY VALUE 2.
 01  W02-CHARATTRLENGTH   PIC S9(9) BINARY VALUE ZERO.
 01  W02-CHARATTRS        PIC X     VALUE LOW-VALUES.
 01  W02-HCONN            PIC S9(9) BINARY VALUE ZERO.
 01  W02-HOBJ             PIC S9(9) BINARY.
 01  W02-COMPCODE         PIC S9(9) BINARY.
 01  W02-REASON           PIC S9(9) BINARY.
 01  W02-SELECTORS-TABLE.
     05  W02-SELECTORS    PIC S9(9) BINARY OCCURS 2 TIMES
 01  W02-INTATTRS-TABLE.
     05  W02-INTATTRS     PIC S9(9) BINARY OCCURS 2 TIMES
*
*    CMQODV defines the object descriptor (MQOD).
*
 01  MQM-OBJECT-DESCRIPTOR.
     COPY CMQODV.
*
* CMQV contains constants (for setting or testing field
* values) and return codes (for testing the result of a
* call).
*
 01  MQM-CONSTANTS.
 COPY CMQV SUPPRESS.
* -------------------------------------------------------*
 PROCEDURE DIVISION.
* -------------------------------------------------------*
*
*    Get the queue name and open the queue.
*
     ⋮
*
*    Initialize the variables for the inquiry call:
*     - Set W02-SELECTORS-TABLE to the attributes whose
*    status is required
*     - All other variables are already set
*
     MOVE MQIA-INHIBIT-GET TO W02-SELECTORS(1).
     MOVE MQIA-INHIBIT-PUT TO W02-SELECTORS(2).
```

```
*
*    Inquire about the attributes.
*
     CALL 'MQINQ' USING W02-HCONN,
                        W02-HOBJ,
                        W02-SELECTORCOUNT,
                        W02-SELECTORS-TABLE,
                        W02-INTATTRCOUNT,
                        W02-INTATTRS-TABLE,
                        W02-CHARATTRLENGTH,
                        W02-CHARATTRS,
                        W02-COMPCODE,
                        W02-REASON.
*
* Test the output from the inquiry:
*
* - If the completion code is not OK, display an error
*   message showing the completion and reason codes
*
* - Otherwise, move the correct attribute status into
*   the relevant screen map fields
*
     IF W02-COMPCODE NOT = MQCC-OK
        MOVE 'MQINQ'       TO M01-MSG4-OPERATION
        MOVE W02-COMPCODE  TO M01-MSG4-COMPCODE
        MOVE W02-REASON    TO M01-MSG4-REASON
```

```
        MOVE M01-MESSAGE-4 TO M00-MESSAGE
*
     ELSE
*       Process the changes.
        ⋮
           END-IF.
        ⋮
```

Impostazione degli attributi di una coda

Questo esempio illustra come utilizzare la chiamata MQSET per modificare gli attributi di una coda.

Questo estratto viene estratto dall'applicazione di esempio Attributi coda (programma CSQ4CVC1) fornita con IBM MQ for z/OS. Per i nomi e le ubicazioni delle applicazioni di esempio su altre piattaforme, consultare Programmi di procedura di esempio (piattaforme eccetto z/OS)

```
⋮
* -------------------------------------------------------*
 WORKING-STORAGE SECTION.
* -------------------------------------------------------*
*
*    W02 - MQM API fields
*
 01  W02-SELECTORCOUNT    PIC S9(9) BINARY VALUE 2.
 01  W02-INTATTRCOUNT     PIC S9(9) BINARY VALUE 2.
 01  W02-CHARATTRLENGTH   PIC S9(9) BINARY VALUE ZERO.
 01  W02-CHARATTRS        PIC X     VALUE LOW-VALUES.
 01  W02-HCONN            PIC S9(9) BINARY VALUE ZERO.
 01  W02-HOBJ             PIC S9(9) BINARY.
 01  W02-COMPCODE         PIC S9(9) BINARY.
 01  W02-REASON           PIC S9(9) BINARY.
 01  W02-SELECTORS-TABLE.
     05  W02-SELECTORS    PIC S9(9) BINARY OCCURS 2 TIMES.
 01  W02-INTATTRS-TABLE.
     05  W02-INTATTRS     PIC S9(9) BINARY OCCURS 2 TIMES.
*
*    CMQODV defines the object descriptor (MQOD).
*
 01  MQM-OBJECT-DESCRIPTOR.
     COPY CMQODV.
*
* CMQV contains constants (for setting or testing
* field values) and return codes (for testing the
* result of a call).
*
 01  MQM-CONSTANTS.
 COPY CMQV SUPPRESS.
* -------------------------------------------------------*
 PROCEDURE DIVISION.
* -------------------------------------------------------*
```

```
*
*    Get the queue name and open the queue.
*
⋮
*
*
* Initialize the variables required for the set call:
* - Set W02-SELECTORS-TABLE to the attributes to be set
* - Set W02-INTATTRS-TABLE to the required status
* - All other variables are already set
*
     MOVE MQIA-INHIBIT-GET   TO W02-SELECTORS(1).
     MOVE MQIA-INHIBIT-PUT   TO W02-SELECTORS(2).
     MOVE MQQA-GET-INHIBITED TO W02-INTATTRS(1).
     MOVE MQQA-PUT-INHIBITED TO W02-INTATTRS(2).
*
*    Set the attributes.
*
     CALL 'MQSET' USING W02-HCONN,
                        W02-HOBJ,
                        W02-SELECTORCOUNT,
                        W02-SELECTORS-TABLE,
                        W02-INTATTRCOUNT,
                        W02-INTATTRS-TABLE,
                        W02-CHARATTRLENGTH,
```

```
                        W02-CHARATTRS,
                        W02-COMPCODE,
                        W02-REASON.
*
* Test the output from the call:
*
*  - If the completion code is not OK, display an error
*    message showing the completion and reason codes
*
*  - Otherwise, move 'INHIBITED' into the relevant
*    screen map fields
*
     IF W02-COMPCODE NOT = MQCC-OK
        MOVE 'MQSET'       TO M01-MSG4-OPERATION
        MOVE W02-COMPCODE  TO M01-MSG4-COMPCODE
        MOVE W02-REASON    TO M01-MSG4-REASON
        MOVE M01-MESSAGE-4 TO M00-MESSAGE
     ELSE
*
*       Process the changes.
⋮
     END-IF.
```

z/OS Esempi di linguaggio assembler System/390

Questa raccolta di argomenti è per lo più tratta dalle applicazioni di esempio IBM MQ for z/OS .

z/OS *Connessione a un gestore code*

Questo esempio illustra come utilizzare la chiamata MQCONN per connettere un programma a un gestore code in batch z/OS .

Questo estratto viene estratto dal programma di esempio Browse (CSQ4BAA1) fornito con IBM MQ for z/OS.

```
⋮
WORKAREA DSECT
*
PARMLIST CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
COMPCODE DS    F              Completion code
REASON   DS    F              Reason code
HCONN    DS    F              Connection handle
         ORG
PARMADDR DS    F              Address of parm field
PARMLEN  DS    H              Length of parm field
*
MQMNAME  DS    CL48           Queue manager name
*
*
************************************************************
*  SECTION NAME : MAINPARM                                 *
************************************************************
MAINPARM DS    0H
         MVI   MQMNAME,X'40'
         MVC   MQMNAME+1(L'MQMNAME-1),MQMNAME
*
* Space out first byte and initialize
*
*
* Code to address and verify parameters passed omitted
*
*
PARM1MVE DS    0H
         SR    R1,R3           Length of data
         LA    R4,MQMNAME      Address for target
         BCTR  R1,R0           Reduce for execute
         EX    R1,MOVEPARM     Move the data
*
************************************************************
* EXECUTES                                                 *
************************************************************
MOVEPARM MVC   0(*-*,R4),0(R3)
*
         EJECT
```

```
*********************************************************
*  SECTION NAME : MAINCONN                              *
*********************************************************
*
*
MAINCONN DS    0H
         XC    HCONN,HCONN       Null connection handle
*
         CALL  MQCONN,                          X
               (MQMNAME,                        X
               HCONN,                           X
               COMPCODE,                        X
               REASON),                         X
               MF=(E,PARMLIST),VL
*
         LA    R0,MQCC_OK        Expected compcode
         C     R0,COMPCODE       As expected?
         BER   R6                Yes .. return to caller
*
         MVC   INF4_TYP,=CL10'CONNECT   '
         BAL   R7,ERRCODE              Translate error
         LA    R0,8                    Set exit code
         ST    R0,EXITCODE             to 8
         B     ENDPROG                 End the program
*
```

z/OS *Disconnessione da un gestore code*

Questo esempio illustra come utilizzare la chiamata MQDISC per disconnettere un programma da un gestore code nel batch z/OS .

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
⋮
*
*        ISSUE MQI DISC REQUEST USING REENTRANT FORM
*        OF CALL MACRO
*
*        HCONN WAS SET BY A PREVIOUS MQCONN REQUEST
*        R5 = WORK REGISTER
*
DISC     DS    0H
         CALL  MQDISC,                       X
               (HCONN,                       X
               COMPCODE,                     X
               REASON),                      X
               VL,MF=(E,CALLLST)
*
         LA    R5,MQCC_OK
         C     R5,COMPCODE
         BNE   BADCALL
         ⋮
```

```
BADCALL  DS    0H
⋮
*                    CONSTANTS
*
         CMQA
*
*        WORKING STORAGE (RE-ENTRANT)
*
WEG3     DSECT
*
CALLLST  CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
HCONN    DS   F
COMPCODE DS   F
REASON   DS   F
*
*
LEG3     EQU  *-WKEG3
         END
```

z/OS *Creazione di una coda dinamica*

Questo esempio illustra come utilizzare la chiamata MQOPEN per creare una coda dinamica.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
⋮
*
*    R5 = WORK REGISTER.
*
OPEN     DS   0H
*
         MVC  WOD_AREA,MQOD_AREA INITIALIZE WORKING VERSION OF
*                                  MQOD WITH DEFAULTS
         MVC  WOD_OBJECTNAME,MOD_Q    COPY IN THE MODEL Q NAME
         MVC  WOD_DYNAMICQNAME,DYN_Q  COPY IN THE DYNAMIC Q NAME
         L    R5,=AL4(MQOO_OUTPUT)    OPEN FOR OUTPUT AND
         A    R5,=AL4(MQOO_INQUIRE)   INQUIRE
         ST   R5,OPTIONS
```

```
*
* ISSUE MQI OPEN REQUEST USING REENTRANT
* FORM OF CALL MACRO
*
         CALL MQOPEN,                           X
               (HCONN,                          X
               WOD,                             X
               OPTIONS,                         X
               HOBJ,                            X
               COMPCODE,                        X
               REASON),VL,MF=(E,CALLLST)
*
         LA  R5,MQCC_OK               CHECK THE COMPLETION CODE
         C   R5,COMPCODE              FROM THE REQUEST AND BRANCH
         BNE BADCALL                  TO ERROR ROUTINE IF NOT MQCC_OK
*
         MVC  TEMP_Q,WOD_OBJECTNAME  SAVE NAME OF TEMPORARY Q
*                            CREATED BY OPEN OF MODEL Q
*
⋮
BADCALL  DS   0H
⋮
*
*
*   CONSTANTS:
*
MOD_Q  DC   CL48'QUERY.REPLY.MODEL'  MODEL QUEUE NAME
DYN_Q  DC   CL48'QUERY.TEMPQ.*'      DYNAMIC QUEUE NAME
*
       CMQODA DSECT=NO,LIST=YES  CONSTANT VERSION OF MQOD
       CMQA                      MQI VALUE EQUATES
*
*   WORKING STORAGE
*
         DFHEISTG
HCONN    DS F                      CONNECTION HANDLE
OPTIONS  DS F                      OPEN OPTIONS
HOBJ     DS F                      OBJECT HANDLE
COMPCODE DS F                      MQI COMPLETION CODE
REASON   DS F                      MQI REASON CODE
TEMP_Q   DS CL(MQ_Q_NAME_LENGTH)   SAVED QNAME AFTER OPEN
*
WOD      CMQODA DSECT=NO,LIST=YES  WORKING VERSION OF MQOD
*
CALLLST  CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L  LIST FORM
                                                 OF CALL
*                                                 MACRO
         ⋮
         END
```

z/OS *Apertura di una coda esistente*

Questo esempio mostra come utilizzare la chiamata MQOPEN per aprire una coda che è già stata definita.

Mostra come specificare due opzioni. Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
⋮
*
*    R5 = WORK REGISTER.
*
OPEN     DS   0H
*
         MVC  WOD_AREA,MQOD_AREA  INITIALIZE WORKING VERSION OF
*                                 MQOD WITH DEFAULTS
         MVC  WOD_OBJECTNAME,Q_NAME    SPECIFY Q NAME TO OPEN
         LA   R5,MQOO_INPUT_EXCLUSIVE  OPEN FOR MQGET CALLS
*
         ST   R5,OPTIONS
*
* ISSUE MQI OPEN REQUEST USING REENTRANT FORM
* OF CALL MACRO
*
         CALL MQOPEN,                          X
               (HCONN,                         X
               WOD,                            X
               OPTIONS,                        X
               HOBJ,                           X
               COMPCODE,                       X
               REASON),VL,MF=(E,CALLLST)
*
         LA  R5,MQCC_OK         CHECK THE COMPLETION CODE
         C   R5,COMPCODE        FROM THE REQUEST AND BRANCH
         BNE BADCALL            TO ERROR ROUTINE IF NOT MQCC_OK
*
         ⋮
BADCALL  DS   0H
⋮
*
*
*   CONSTANTS:
*
Q_NAME   DC   CL48'REQUEST.QUEUE'  NAME OF QUEUE TO OPEN
*
         CMQODA DSECT=NO,LIST=YES  CONSTANT VERSION OF MQOD
         CMQA                      MQI VALUE EQUATES
*
*   WORKING STORAGE
*
         DFHEISTG
HCONN    DS F           CONNECTION HANDLE
OPTIONS  DS F           OPEN OPTIONS
HOBJ     DS F           OBJECT HANDLE
COMPCODE DS F           MQI COMPLETION CODE
REASON   DS F           MQI REASON CODE
*
WOD  CMQODA DSECT=NO,LIST=YES  WORKING VERSION OF MQOD
*
CALLLST  CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L  LIST FORM
                                                 OF CALL
*                                                 MACRO
         ⋮
         END
```

z/OS *Chiusura di una coda*

Questo esempio illustra come utilizzare la chiamata MQCLOSE per chiudere una coda.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
⋮
*
* ISSUE MQI CLOSE REQUEST USING REENTRANT FROM OF
* CALL MACRO
*
*        HCONN WAS SET BY A PREVIOUS MQCONN REQUEST
*        HOBJ  WAS SET BY A PREVIOUS MQOPEN REQUEST
*        R5 = WORK REGISTER
*
CLOSE    DS    0H
         LA    R5,MQCO_NONE        NO SPECIAL CLOSE OPTIONS
```

```
         ST    R5,OPTIONS              ARE REQUIRED.
*
         CALL  MQCLOSE,                        X
               (HCONN,                         X
               HOBJ,                           X
               OPTIONS,                        X
               COMPCODE,                       X
               REASON),                        X
               VL,MF=(E,CALLLST)
*
         LA    R5,MQCC_OK
         C     R5,COMPCODE
         BNE   BADCALL
*
         ⋮
BADCALL  DS    0H
         ⋮
*                   CONSTANTS
*
         CMQA
*
*        WORKING STORAGE (REENTRANT)
*
WEG4     DSECT
*
CALLLST  CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
HCONN    DS   F
HOBJ     DS   F
OPTIONS  DS   F
COMPCODE DS   F
REASON   DS   F
*
*
LEG4     EQU  *-WKEG4
         END
```

z/OS *Inserimento di un messaggio utilizzando MQPUT*

Questo esempio illustra come utilizzare la chiamata MQPUT per inserire un messaggio su una coda.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
⋮
*     CONNECT TO QUEUE MANAGER
*
CONN    DS  0H
⋮
*
*     OPEN A QUEUE
*
OPEN    DS  0H
⋮
*
*     R4,R5,R6,R7 = WORK REGISTER.
*
PUT  DS  0H
     LA   R4,MQMD           SET UP ADDRESSES AND
     LA   R5,MQMD_LENGTH    LENGTH FOR USE BY MVCL
     LA   R6,WMD            INSTRUCTION, AS MQMD IS
     LA   R7,WMD_LENGTH     OVER 256 BYES LONG.
     MVCL R6,R4             INITIALIZE WORKING VERSION
*                           OF MESSAGE DESCRIPTOR
*
     MVC  WPMO_AREA,MQPMO_AREA  INITIALIZE WORKING MQPMO
*

     LA   R5,BUFFER_LEN   RETRIEVE THE BUFFER LENGTH
     ST   R5,BUFFLEN      AND SAVE IT FOR MQM USE
*
     MVC  BUFFER,TEST_MSG    SET THE MESSAGE TO BE PUT
*
*    ISSUE MQI PUT REQUEST USING REENTRANT FORM
*    OF CALL MACRO
*
*        HCONN WAS SET BY PREVIOUS MQCONN REQUEST
*        HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST
*
         CALL MQPUT,                        X
```

```
               (HCONN,                         X
               HOBJ,                           X
               WMD,                            X
               WPMO,                           X
               BUFFLEN,                        X
               BUFFER,                         X
               COMPCODE,                       X
               REASON),VL,MF=(E,CALLLST)
*
         LA  R5,MQCC_OK
         C   R5,COMPCODE
         BNE BADCALL
*
         ⋮
BADCALL DS  0H
⋮
```

```
*
*   CONSTANTS
*
CMQMDA DSECT=NO,LIST=YES,PERSISTENCE=MQPER_PERSISTENT
CMQPMOA DSECT=NO,LIST=YES
CMQA
TEST_MSG DC CL80'THIS IS A TEST MESSAGE'
*
*   WORKING STORAGE DSECT
*
WORKSTG DSECT
*
COMPCODE DS F
REASON  DS F
BUFFLEN DS F
OPTIONS DS F
HCONN  DS F
HOBJ   DS F
*
BUFFER  DS CL80
BUFFER_LEN EQU *-BUFFER
*
WMD   CMQMDA DSECT=NO,LIST=NO
WPMO   CMQPMOA DSECT=NO,LIST=NO
*
CALLLST CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
⋮
END
```

z/OS *Inserimento di un messaggio utilizzando MQPUT1*

Questo esempio illustra come utilizzare la chiamata MQPUT1 per aprire una coda, inserire un singolo messaggio nella coda e chiudere la coda.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
⋮
*
*     CONNECT TO QUEUE MANAGER
*
CONN    DS  0H
⋮
*
*     R4,R5,R6,R7 = WORK REGISTER.
*
PUT     DS  0H
*
         MVC  WOD_AREA,MQOD_AREA     INITIALIZE WORKING VERSION OF
*                                    MQOD WITH DEFAULTS
         MVC  WOD_OBJECTNAME,Q_NAME  SPECIFY Q NAME FOR PUT1
*
         LA   R4,MQMD           SET UP ADDRESSES AND
         LA   R5,MQMD_LENGTH    LENGTH FOR USE BY MVCL
         LA   R6,WMD            INSTRUCTION, AS MQMD IS
         LA   R7,WMD_LENGTH     OVER 256 BYES LONG.
         MVCL R6,R4             INITIALIZE WORKING VERSION
*                               OF MESSAGE DESCRIPTOR
```

```
*
         MVC  WPMO_AREA,MQPMO_AREA       INITIALIZE WORKING MQPMO
*
         LA   R5,BUFFER_LEN           RETRIEVE THE BUFFER LENGTH
         ST   R5,BUFFLEN              AND SAVE IT FOR MQM USE
*
         MVC  BUFFER,TEST_MSG         SET THE MESSAGE TO BE PUT
*
* ISSUE MQI PUT REQUEST USING REENTRANT FORM OF CALL MACRO
*
*        HCONN WAS SET BY PREVIOUS MQCONN REQUEST
*        HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST
*
         CALL MQPUT1,                                X
               (HCONN,                               X
                LMQOD,                               X
                LMQMD,                               X
                LMQPMO,                              X
                BUFFERLENGTH,                        X
                BUFFER,                              X
                COMPCODE,                            X
                REASON),VL,MF=(E,CALLLST)
*
         LA  R5,MQCC_OK
         C   R5,COMPCODE
         BNE BADCALL
*        ⋮
BADCALL DS  0H
⋮
*
```

```
*      CONSTANTS
*
  CMQMDA DSECT=NO,LIST=YES,PERSISTENCE=MQPER_PERSISTENT
  CMQPMOA DSECT=NO,LIST=YES
  CMQODA DSECT=NO,LIST=YES
  CMQA
*
TEST_MSG DC CL80'THIS IS ANOTHER TEST MESSAGE'
Q_NAME   DC CL48'TEST.QUEUE.NAME'
*
*      WORKING STORAGE DSECT
*
WORKSTG  DSECT
*
COMPCODE DS F
REASON   DS F
BUFFLEN  DS F
OPTIONS  DS F
HCONN    DS F
HOBJ     DS F
*
BUFFER   DS CL80
BUFFER_LEN EQU *-BUFFER
*
WOD      CMQODA DSECT=NO,LIST=YES    WORKING VERSION OF MQOD
WMD      CMQMDA DSECT=NO,LIST=NO
WPMO     CMQPMOA DSECT=NO,LIST=NO
*
CALLLST  CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
         ⋮
         END
```

z/OS *acquisizione di un messaggio*

Questo esempio mostra come utilizzare la chiamata MQGET per rimuovere un messaggio da una coda.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
⋮
*
*   CONNECT TO QUEUE MANAGER
```

```
*
CONN    DS  0H
⋮
*
*     OPEN A QUEUE FOR GET
*
OPEN    DS  0H
⋮
*
*     R4,R5,R6,R7 = WORK REGISTER.
*
GET  DS  0H
     LA   R4,MQMD               SET UP ADDRESSES AND
     LA   R5,MQMD_LENGTH        LENGTH FOR USE BY MVCL
     LA   R6,WMD                INSTRUCTION, AS MQMD IS
     LA   R7,WMD_LENGTH         OVER 256 BYES LONG.
     MVCL R6,R4                 INITIALIZE WORKING VERSION
*                                  OF MESSAGE DESCRIPTOR
*
     MVC  WGMO_AREA,MQGMO_AREA  INITIALIZE WORKING MQGMO
*

     LA   R5,BUFFER_LEN         RETRIEVE THE BUFFER LENGTH
     ST   R5,BUFFLEN            AND SAVE IT FOR MQM USE
*
*
* ISSUE MQI GET REQUEST USING REENTRANT FORM OF CALL MACRO
*
*        HCONN WAS SET BY PREVIOUS MQCONN REQUEST
*        HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST
*
         CALL  MQGET,                          X
               (HCONN,                         X
               HOBJ,                           X
               WMD,                            X
               WGMO,                           X
               BUFFLEN,                        X
               BUFFER,                         X
               DATALEN,                        X
               COMPCODE,                       X
               REASON),                        X
               VL,MF=(E,CALLLST)
*
         LA  R5,MQCC_OK
         C   R5,COMPCODE
         BNE BADCALL
*        ⋮
BADCALL DS  0H
⋮
```

```
*
*      CONSTANTS
*
         CMQMDA DSECT=NO,LIST=YES
         CMQGMOA DSECT=NO,LIST=YES
         CMQA
*
*      WORKING STORAGE DSECT
*
WORKSTG  DSECT
*
COMPCODE DS F
REASON   DS F
BUFFLEN  DS F
DATALEN  DS F
OPTIONS  DS F
HCONN    DS F
HOBJ     DS F
*
BUFFER   DS CL80
BUFFER_LEN EQU *-BUFFER
*
WMD      CMQMDA DSECT=NO,LIST=NO
WGMO     CMQGMOA DSECT=NO,LIST=NO
*
CALLLST  CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
```

```
:
END
```

z/OS *Ricezione di un messaggio utilizzando l'opzione di attesa*

Questo esempio dimostra come utilizzare l'opzione di attesa della chiamata MQGET.

Questo codice accetta messaggi troncati. Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
:
*     CONNECT TO QUEUE MANAGER
CONN    DS  0H
:
*     OPEN A QUEUE FOR GET
OPEN    DS  0H
:
*     R4,R5,R6,R7 = WORK REGISTER.
GET  DS  0H
     LA   R4,MQMD              SET UP ADDRESSES AND
     LA   R5,MQMD_LENGTH       LENGTH FOR USE BY MVCL
     LA   R6,WMD               INSTRUCTION, AS MQMD IS
     LA   R7,WMD_LENGTH        OVER 256 BYES LONG.
     MVCL R6,R4                INITIALIZE WORKING VERSION
*                                  OF MESSAGE DESCRIPTOR
```

```
*
     MVC  WGMO_AREA,MQGMO_AREA   INITIALIZE WORKING MQGMO
     L    R5,=AL4(MQGMO_WAIT)
     A    R5,=AL4(MQGMO_ACCEPT_TRUNCATED_MSG)
     ST   R5,WGMO_OPTIONS
     MVC  WGMO_WAITINTERVAL,TWO_MINUTES   WAIT UP TO TWO
                                          MINUTES BEFORE
                                          FAILING THE
                                          CALL
*
     LA   R5,BUFFER_LEN     RETRIEVE THE BUFFER LENGTH
     ST   R5,BUFFLEN        AND SAVE IT FOR MQM USE
*
*  ISSUE MQI GET REQUEST USING REENTRANT FORM OF CALL MACRO
*
*        HCONN WAS SET BY PREVIOUS MQCONN REQUEST
*        HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST
*
         CALL  MQGET,                              X
               (HCONN,                             X
               HOBJ,                               X
               WMD,                                X
               WGMO,                               X
               BUFFLEN,                            X
               BUFFER,                             X
               DATALEN,                            X
               COMPCODE,                           X
               REASON),                            X
               VL,MF=(E,CALLLST)
*
  LA  R5,MQCC_OK                  DID THE MQGET REQUEST
  C   R5,COMPCODE                   WORK OK?
  BE  GETOK                       YES, SO GO AND PROCESS.
  LA  R5,MQCC_WARNING             NO, SO CHECK FOR A WARNING.
  C   R5,COMPCODE                 IS THIS A WARNING?
  BE  CHECK_W                     YES, SO CHECK THE REASON.
*
  LA  R5,MQRC_NO_MSG_AVAILABLE   IT MUST BE AN ERROR.
                                 IS IT DUE TO AN EMPTY
  C   R5,REASON                  QUEUE?
  BE  NOMSG                      YES, SO HANDLE THE ERROR
  B   BADCALL                    NO, SO GO TO ERROR ROUTINE
*
CHECK_W  DS  0H
         LA  R5,MQRC_TRUNCATED_MSG_ACCEPTED   IS THIS A
                                              TRUNCATED
         C   R5,REASON                        MESSAGE?
         BE  GETOK                  YES, SO GO AND PROCESS.
         B   BADCALL                NO, SOME OTHER WARNING
*
```

```
NOMSG    DS  0H
         ⋮
GETOK    DS  0H
         ⋮
```

```
BADCALL DS  0H
⋮
*
*      CONSTANTS
*
         CMQMDA DSECT=NO,LIST=YES
         CMQGMOA DSECT=NO,LIST=YES
         CMQA
*
TWO_MINUTES DC F'120000'       GET WAIT INTERVAL
*
*      WORKING STORAGE DSECT
```

```
*
WORKSTG  DSECT
*
COMPCODE DS F
REASON   DS F
BUFFLEN  DS F
DATALEN  DS F
OPTIONS  DS F
HCONN    DS F
HOBJ     DS F
*
BUFFER   DS CL80
BUFFER_LEN EQU *-BUFFER
*
WMD      CMQMDA DSECT=NO,LIST=NO
WGMO     CMQGMOA DSECT=NO,LIST=NO
*
CALLLST  CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
         ⋮
         END
```

z/OS *Ricezione di un messaggio utilizzando la segnalazione*

Questo esempio mostra come utilizzare la chiamata MQGET per impostare un segnale in modo da ricevere una notifica quando un messaggio adatto arriva su una coda.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
⋮
*
*     CONNECT TO QUEUE MANAGER
*
CONN    DS  0H
        ⋮
*
*     OPEN A QUEUE FOR GET
*
OPEN    DS  0H
        ⋮
*
*     R4,R5,R6,R7 = WORK REGISTER.
*
GET  DS  0H
     LA   R4,MQMD                SET UP ADDRESSES AND
     LA   R5,MQMD_LENGTH         LENGTH FOR USE BY MVCL
     LA   R6,WMD                 INSTRUCTION, AS MQMD IS
     LA   R7,WMD_LENGTH          OVER 256 BYES LONG.
     MVCL R6,R4                  INITIALIZE WORKING VERSION
*                                   OF MESSAGE DESCRIPTOR
```

```
*
     MVC  WGMO_AREA,MQGMO_AREA   INITIALIZE WORKING MQGMO
     LA   R5,MQGMO_SET_SIGNAL
```

```
     ST   R5,WGMO_OPTIONS
     MVC  WGMO_WAITINTERVAL,FIVE_MINUTES   WAIT UP TO FIVE
                                           MINUTES BEFORE
*                                          FAILING THE CALL
*
     XC   SIG_ECB,SIG_ECB   CLEAR THE ECB
     LA   R5,SIG_ECB        GET THE ADDRESS OF THE ECB
     ST   R5,WGMO_SIGNAL1   AND PUT IT IN THE WORKING
*                                 MQGMO
*

     LA   R5,BUFFER_LEN     RETRIEVE THE BUFFER LENGTH
     ST   R5,BUFFLEN        AND SAVE IT FOR MQM USE
*
*
*    ISSUE MQI GET REQUEST USING REENTRANT FORM OF CALL MACRO
*
*        HCONN WAS SET BY PREVIOUS MQCONN REQUEST
*        HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST
*
         CALL  MQGET,                          X
               (HCONN,                         X
               HOBJ,                           X
               WMD,                            X
               WGMO,                           X
               BUFFLEN,                        X
               BUFFER,                         X
               DATALEN,                        X
               COMPCODE,                       X
               REASON),                        X
               VL,MF=(E,CALLLST)
*
         LA  R5,MQCC_OK        DID THE MQGET REQUEST
         C   R5,COMPCODE       WORK OK?
         BE  GETOK             YES, SO GO AND PROCESS.
         LA  R5,MQCC_WARNING   NO, SO CHECK FOR A WARNING.
         C   R5,COMPCODE       IS THIS A WARNING?
         BE  CHECK_W           YES, SO CHECK THE REASON.
         B   BADCALL           NO, SO GO TO ERROR ROUTINE
*
```

```
CHECK_W  DS  0H
         LA  R5,MQRC_SIGNAL_REQUEST_ACCEPTED
         C   R5,REASON   SIGNAL REQUEST SIGNAL SET?
         BNE BADCALL     NO, SOME ERROR OCCURRED
         B   DOWORK      YES, SO DO SOMETHING
*                          ELSE
*
CHECKSIG DS  0H
         CLC SIG_ECB+1(3),=AL3(MQEC_MSG_ARRIVED)
                            IS A MESSAGE AVAILABLE?
         BE  GET            YES, SO GO AND GET IT
*
         CLC SIG_ECB+1(3),=AL3(MQEC_WAIT_INTERVAL_EXPIRED)
                          HAVE WE WAITED LONG ENOUGH?
         BE  NOMSG        YES, SO SAY NO MSG AVAILABLE
         B   BADCALL      IF IT'S ANYTHING ELSE
*                         GO TO ERROR ROUTINE.
*
DOWORK   DS  0H
         ⋮
         TM  SIG_ECB,X'40'   HAS THE SIGNAL ECB BEEN POSTED?
         BO  CHECKSIG        YES, SO GO AND CHECK WHY
         B   DOWORK          NO, SO GO AND DO MORE WORK
*
NOMSG    DS  0H
         ⋮
GETOK    DS  0H
         ⋮
BADCALL DS  0H
        ⋮
*
*      CONSTANTS
*
         CMQMDA DSECT=NO,LIST=YES
         CMQGMOA DSECT=NO,LIST=YES
         CMQA
*
FIVE_MINUTES DC F'300000'        GET SIGNAL INTERVAL
*
```

```
*      WORKING STORAGE DSECT
*
WORKSTG  DSECT
*
COMPCODE DS F
REASON   DS F
BUFFLEN  DS F
DATALEN  DS F
OPTIONS  DS F
HCONN    DS F
HOBJ     DS F
SIG_ECB  DS F
```

```
*
BUFFER   DS CL80
BUFFER_LEN EQU *-BUFFER
*
WMD      CMQMDA DSECT=NO,LIST=NO
WGMO     CMQGMOA DSECT=NO,LIST=NO
*
CALLLST  CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
⋮
END
```

z/OS *Richiesta e impostazione degli attributi di una coda*

Questo esempio dimostra come utilizzare la chiamata MQINQ per richiedere informazioni sugli attributi di una coda e per utilizzare la chiamata MQSET per modificare gli attributi di una coda.

Questo estratto viene estratto dall'applicazione di esempio Attributi coda (programma CSQ4CAC1) fornita con IBM MQ for z/OS.

```
⋮
DFHEISTG DSECT
         ⋮
OBJDESC  CMQODA LIST=YES   Working object descriptor
*
SELECTORCOUNT   DS F       Number of selectors
INTATTRCOUNT    DS F       Number of integer attributes
CHARATTRLENGTH  DS F       char attributes length
CHARATTRS       DS C       Area for char attributes
*
OPTIONS  DS   F            Command options
HCONN    DS   F            Handle of connection
HOBJ     DS   F            Handle of object
COMPCODE DS   F            Completion code
REASON   DS   F            Reason code
SELECTOR DS   2F           Array of selectors
INTATTRS DS   2F           Array of integer attributes
         ⋮
OBJECT   DS   CL(MQ_Q_NAME_LENGTH)   Name of queue
         ⋮
CALLLIST CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*************************************************************
*                   PROGRAM EXECUTION STARTS HERE        *
⋮
CSQ4CAC1 DFHEIENT CODEREG=(R3),DATAREG=(R13)
         ⋮
*        Initialize the variables for the set call
*
         SR   R0,R0               Clear register zero
         ST   R0,CHARATTRLENGTH   Set char length to zero
         LA   R0,2                Load to set
         ST   R0,SELECTORCOUNT    selectors add
         ST   R0,INTATTRCOUNT     integer attributes
*
         LA   R0,MQIA_INHIBIT_GET Load q attribute selector
         ST   R0,SELECTOR+0       Place in field
         LA   R0,MQIA_INHIBIT_PUT Load q attribute selector
         ST   R0,SELECTOR+4       Place in field
*
UPDTEST  DS   0H
         CLC  ACTION,CINHIB       Are we inhibiting?
         BE   UPDINHBT            Yes branch to section
*
```

```
         CLC  ACTION,CALLOW       Are we allowing?
         BE   UPDALLOW            Yes branch to section
*
         MVC  M00_MSG,M01_MSG1    Invalid request
         BR   R6                  Return to caller
*
```

```
UPDINHBT DS   0H
         MVC  UPDTYPE,CINHIBIT      Indicate action type
         LA   R0,MQQA_GET_INHIBITED Load attribute value
         ST   R0,INTATTRS+0         Place in field
         LA   R0,MQQA_PUT_INHIBITED Load attribute value
         ST   R0,INTATTRS+4         Place in field
         B    UPDCALL               Go and do call
*
UPDALLOW DS   0H
         MVC  UPDTYPE,CALLOWED      Indicate action type
         LA   R0,MQQA_GET_ALLOWED   Load attribute value
         ST   R0,INTATTRS+0         Place in field
         LA   R0,MQQA_PUT_ALLOWED   Load attribute value
         ST   R0,INTATTRS+4         Place in field
         B    UPDCALL               Go and do call
*
UPDCALL  DS   0H
         CALL MQSET,                          C
               (HCONN,                        C
               HOBJ,                          C
               SELECTORCOUNT,                 C
               SELECTOR,                      C
               INTATTRCOUNT,                  C
               INTATTRS,                      C
               CHARATTRLENGTH,                C
               CHARATTRS,                     C
               COMPCODE,                      C
               REASON),                       C
               VL,MF=(E,CALLLIST)
*
         LA   R0,MQCC_OK    Load expected compcode
         C    R0,COMPCODE   Was set successful?
         ⋮
*  SECTION NAME : INQUIRE                                  *
*  FUNCTION     : Inquires on the objects attributes       *
*  CALLED BY    : PROCESS                                  *
*  CALLS        : OPEN, CLOSE, CODES                       *
*  RETURN       : To Register 6                            *
INQUIRE  DS   0H
         ⋮
```

```
*        Initialize the variables for the inquire call
*
         SR   R0,R0               Clear register zero
         ST   R0,CHARATTRLENGTH   Set char length to zero
         LA   R0,2                Load to set
         ST   R0,SELECTORCOUNT    selectors add
         ST   R0,INTATTRCOUNT     integer attributes
*
         LA   R0,MQIA_INHIBIT_GET Load attribute value
         ST   R0,SELECTOR+0       Place in field
         LA   R0,MQIA_INHIBIT_PUT Load attribute value
         ST   R0,SELECTOR+4       Place in field
         CALL MQINQ,                          C
               (HCONN,                        C
               HOBJ,                          C
               SELECTORCOUNT,                 C
               SELECTOR,                      C
               INTATTRCOUNT,                  C
               INTATTRS,                      C
               CHARATTRLENGTH,                C
               CHARATTRS,                     C
               COMPCODE,                      C
               REASON),                       C
               VL,MF=(E,CALLLIST)
         LA   R0,MQCC_OK        Load expected compcode
         C    R0,COMPCODE       Was inquire successful?
         ⋮
```

z/OS Esempi PL/I

L'utilizzo di PL/I è supportato solo da z/OS . Questa raccolta di argomenti illustra tecniche che utilizzano esempi PL/I.

z/OS *Connessione a un gestore code*

Questo esempio illustra come utilizzare la chiamata MQCONN per connettere un programma a un gestore code in batch z/OS .

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
     %INCLUDE SYSLIB(CMQP);
     %INCLUDE SYSLIB(CMQEPP);
     :
     /*****************************************************/
     /* STRUCTURE BASED ON PARAMETER INPUT AREA (PARAM)  */
     /*****************************************************/
     DCL 1 INPUT_PARAM      BASED(ADDR(PARAM)),
           2 PARAM_LENGTH   FIXED BIN(15),
           2 PARAM_MQMNAME  CHAR(48);
           ⋮
     /*****************************************************/
     /* WORKING STORAGE DECLARATIONS                      */
     /*****************************************************/
     DCL MQMNAME                  CHAR(48);
     DCL COMPCODE                 BINARY FIXED (31);
     DCL REASON                   BINARY FIXED (31);
     DCL HCONN                    BINARY FIXED (31);
     ⋮
     /*****************************************************/
     /* COPY QUEUE MANAGER NAME PARAMETER                 */
     /* TO LOCAL STORAGE                                  */
     /*****************************************************/
     MQMNAME = ' ';
     MQMNAME = SUBSTR(PARAM_MQMNAME,1,PARAM_LENGTH);
     ⋮
     /*****************************************************/
     /* CONNECT FROM THE QUEUE MANAGER                    */
     /*****************************************************/
     CALL MQCONN (MQMNAME,     /* MQM SYSTEM NAME         */
                  HCONN,       /* CONNECTION HANDLE       */
                  COMPCODE,    /* COMPLETION CODE         */
                  REASON);     /* REASON CODE             */

     /*****************************************************/
     /* TEST THE COMPLETION CODE OF THE CONNECT CALL.     */
     /* IF THE CALL HAS FAILED ISSUE AN ERROR MESSAGE     */
     /* SHOWING THE COMPLETION CODE AND THE REASON CODE. */
     /*****************************************************/
     IF COMPCODE ¬= MQCC_OK
        THEN DO;
          ⋮
          CALL ERROR_ROUTINE;
        END;
```

z/OS *Disconnessione da un gestore code*

Questo esempio illustra come utilizzare la chiamata MQDISC per disconnettere un programma da un gestore code nel batch z/OS .

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
     %INCLUDE SYSLIB(CMQP);
     %INCLUDE SYSLIB(CMQEPP);
     :
     /*****************************************************/
     /* WORKING STORAGE DECLARATIONS                      */
     /*****************************************************/
     DCL COMPCODE                 BINARY FIXED (31);
     DCL REASON                   BINARY FIXED (31);
     DCL HCONN                    BINARY FIXED (31);
⋮
```

```
     /*******************************************************/
     /* DISCONNECT FROM THE QUEUE MANAGER                   */
     /*******************************************************/
     CALL MQDISC (HCONN,       /* CONNECTION HANDLE      */
                  COMPCODE,    /* COMPLETION CODE        */
                  REASON);     /* REASON CODE            */

/*************************************************************/
/* TEST THE COMPLETION CODE OF THE DISCONNECT CALL.          */
/* IF THE CALL HAS FAILED ISSUE AN ERROR MESSAGE             */
/* SHOWING THE COMPLETION CODE AND THE REASON CODE.          */
/*************************************************************/
     IF COMPCODE ¬= MQCC_OK
        THEN DO;
          ⋮
          CALL ERROR_ROUTINE;
        END;
```

z/OS *Creazione di una coda dinamica*

Questo esempio illustra come utilizzare la chiamata MQOPEN per creare una coda dinamica.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
     %INCLUDE SYSLIB(CMQP);
     %INCLUDE SYSLIB(CMQEPP);
     :
/*******************************************************/
/* WORKING STORAGE DECLARATIONS                        */
/*******************************************************/
DCL COMPCODE                 BINARY FIXED (31);
DCL REASON                   BINARY FIXED (31);
DCL HCONN                    BINARY FIXED (31);
DCL HOBJ                     BINARY FIXED (31);
DCL OPTIONS                  BINARY FIXED (31);
⋮
DCL MODEL_QUEUE_NAME     CHAR(48) INIT('PL1.REPLY.MODEL');
DCL DYNAMIC_NAME_PREFIX  CHAR(48) INIT('PL1.TEMPQ.*');
DCL DYNAMIC_QUEUE_NAME   CHAR(48) INIT(' ');
⋮
/*******************************************************/
/* LOCAL COPY OF OBJECT DESCRIPTOR                     */
/*******************************************************/
DCL 1 LMQOD  LIKE MQOD;
⋮
/*******************************************************/
/* SET UP OBJECT DESCRIPTOR FOR OPEN OF REPLY QUEUE    */
/*******************************************************/
LMQOD.OBJECTTYPE =MQOT_Q;
LMQOD.OBJECTNAME = MODEL_QUEUE_NAME;
LMQOD.DYNAMICQNAME = DYNAMIC_NAME_PREFIX;
OPTIONS = MQOO_INPUT_EXCLUSIVE;

     CALL MQOPEN (HCONN,
                  LMQOD,
                  OPTIONS,
                  HOBJ,
                  COMPCODE,
                  REASON);

/*******************************************************/
/* TEST THE COMPLETION CODE OF THE OPEN CALL.          */
/* IF THE CALL HAS FAILED ISSUE AN ERROR MESSAGE       */
/* SHOWING THE COMPLETION CODE AND THE REASON CODE.    */
/* IF THE CALL HAS SUCCEEDED THEN EXTRACT THE NAME OF  */
/* THE NEWLY CREATED DYNAMIC QUEUE FROM THE OBJECT     */
/* DESCRIPTOR.                                         */
/*******************************************************/
     IF COMPCODE ¬= MQCC_OK
        THEN DO;
          ⋮
          CALL ERROR_ROUTINE;
        END;
        ELSE
          DYNAMIC_QUEUE_NAME = LMQOD_OBJECTNAME;
```

z/OS *Apertura di una coda esistente*

Questo esempio illustra come utilizzare la chiamata MQOPEN per aprire una coda esistente.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
%INCLUDE SYSLIB(CMQP);
%INCLUDE SYSLIB(CMQEPP);
:
/*********************************************************/
/* WORKING STORAGE DECLARATIONS                          */
/*********************************************************/
DCL COMPCODE           BINARY FIXED (31);
DCL REASON             BINARY FIXED (31);
DCL HCONN              BINARY FIXED (31);
DCL HOBJ               BINARY FIXED (31);
DCL OPTIONS            BINARY FIXED (31);
⋮
DCL QUEUE_NAME         CHAR(48) INIT('PL1.LOCAL.QUEUE');
⋮
/*********************************************************/
/* LOCAL COPY OF OBJECT DESCRIPTOR                       */
/*********************************************************/
DCL 1 LMQOD  LIKE MQOD;
⋮
/*********************************************************/
/* SET UP OBJECT DESCRIPTOR FOR OPEN OF REPLY QUEUE      */
/*********************************************************/
LMQOD.OBJECTTYPE = MQOT_Q;
LMQOD.OBJECTNAME = QUEUE_NAME;
OPTIONS = MQOO_INPUT_EXCLUSIVE;

CALL MQOPEN (HCONN,
             LMQOD,
             OPTIONS,
             HOBJ,
             COMPCODE,
             REASON);

/*********************************************************/
/* TEST THE COMPLETION CODE OF THE OPEN CALL.            */
/* IF THE CALL HAS FAILED ISSUE AN ERROR MESSAGE         */
/* SHOWING THE COMPLETION CODE AND THE REASON CODE.      */
/*********************************************************/
     IF COMPCODE ¬= MQCC_OK
        THEN DO;
          ⋮
          CALL ERROR_ROUTINE;
        END;
```

z/OS *Chiusura di una coda*

Questo esempio mostra come utilizzare la chiamata MQCLOSE.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
%INCLUDE SYSLIB(CMQP);
%INCLUDE SYSLIB(CMQEPP);
:
/*********************************************************/
/* WORKING STORAGE DECLARATIONS                          */
/*********************************************************/
DCL COMPCODE                  BINARY FIXED (31);
DCL REASON                    BINARY FIXED (31);
DCL HCONN                     BINARY FIXED (31);
DCL HOBJ                      BINARY FIXED (31);
DCL OPTIONS                   BINARY FIXED (31);
⋮
/*********************************************************/
/* SET CLOSE OPTIONS                                     */
/*********************************************************/
OPTIONS=MQCO_NONE;

/*********************************************************/
/* CLOSE QUEUE                                           */
/*********************************************************/
   CALL MQCLOSE (HCONN,      /* CONNECTION HANDLE        */
```

```
                HOBJ,       /* OBJECT HANDLE            */
                OPTIONS,    /* CLOSE OPTIONS            */
                COMPCODE,   /* COMPLETION CODE          */
                REASON);    /* REASON CODE              */

/*************************************************************/
/* TEST THE COMPLETION CODE OF THE CLOSE CALL.               */
/* IF THE CALL HAS FAILED ISSUE AN ERROR MESSAGE             */
/* SHOWING THE COMPLETION CODE AND THE REASON CODE.          */
/*************************************************************/
     IF COMPCODE ¬= MQCC_OK
        THEN DO;
          ⋮
          CALL ERROR_ROUTINE;
        END;
```

z/OS *Inserimento di un messaggio utilizzando MQPUT*

Questo esempio mostra come utilizzare la chiamata MQPUT utilizzando il contesto.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
%INCLUDE SYSLIB(CMQP);
%INCLUDE SYSLIB(CMQEPP);
:
/*************************************************************/
/* WORKING STORAGE DECLARATIONS                              */
/*************************************************************/
DCL COMPCODE                  BINARY FIXED (31);
DCL REASON                    BINARY FIXED (31);
DCL HCONN                     BINARY FIXED (31);
DCL HOBJ                      BINARY FIXED (31);
DCL OPTIONS                   BINARY FIXED (31);
DCL BUFFLEN                   BINARY FIXED (31);
DCL BUFFER                    CHAR(80);
⋮
DCL PL1_TEST_MESSAGE          CHAR(80)
INIT('*****     THIS IS A TEST MESSAGE     *****');
⋮
/*************************************************************/
/* LOCAL COPY OF MESSAGE DESCRIPTOR                          */
/* AND PUT MESSAGE OPTIONS                                   */
/*************************************************************/
DCL 1 LMQMD  LIKE MQMD;
DCL 1 LMQPMO LIKE MQPMO;
⋮
/*************************************************************/
/* SET UP MESSAGE DESCRIPTOR                                 */
/*************************************************************/
LMQMD.MSGTYPE = MQMT_DATAGRAM;
LMQMD.PRIORITY = 1;
LMQMD.PERSISTENCE = MQPER_PERSISTENT;
LMQMD.REPLYTOQ = ' ';
LMQMD.REPLYTOQMGR = ' ';
LMQMD.MSGID = MQMI_NONE;
LMQMD.CORRELID = MQCI_NONE;

/*************************************************************/
/* SET UP PUT MESSAGE OPTIONS                                */
/*************************************************************/
LMQPMO.OPTIONS = MQPMO_NO_SYNCPOINT;

/*************************************************************/
/* SET UP LENGTH OF MESSAGE BUFFER AND THE MESSAGE           */
/*************************************************************/
BUFFLEN = LENGTH(BUFFER);
BUFFER = PL1_TEST_MESSAGE;
/*************************************************************/
/*                                                           */
/* HCONN WAS SET BY PREVIOUS MQCONN REQUEST.                 */
/* HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST.                  */
/*                                                           */
/*************************************************************/
CALL MQPUT (HCONN,
            HOBJ,
            LMQMD,
            LMQPMO,
            BUFFLEN,
            BUFFER,
```

```
            COMPCODE,
            REASON);
```

```
/*************************************************************/
/* TEST THE COMPLETION CODE OF THE PUT CALL.                 */
/* IF THE CALL HAS FAILED ISSUE AN ERROR MESSAGE             */
/* SHOWING THE COMPLETION CODE AND THE REASON CODE.          */
/*************************************************************/
    IF COMPCODE ¬= MQCC_OK
       THEN DO;
          ⋮
          CALL ERROR_ROUTINE;
        END;
```

z/OS *Inserimento di un messaggio utilizzando MQPUT1*

Questo esempio illustra come utilizzare il richiamo MQPUT1 .

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
%INCLUDE SYSLIB(CMQEPP);
%INCLUDE SYSLIB(CMQP);
⋮
/*************************************************************/
/* WORKING STORAGE DECLARATIONS              */
/*************************************************************/
DCL COMPCODE      BINARY FIXED (31);
DCL REASON       BINARY FIXED (31);
DCL HCONN        BINARY FIXED (31);
DCL OPTIONS       BINARY FIXED (31);
DCL BUFFLEN       BINARY FIXED (31);
DCL BUFFER       CHAR(80);
⋮
DCL REPLY_TO_QUEUE   CHAR(48) INIT('PL1.REPLY.QUEUE');
DCL QUEUE_NAME     CHAR(48) INIT('PL1.LOCAL.QUEUE');
DCL PL1_TEST_MESSAGE  CHAR(80)
INIT('***** THIS IS ANOTHER TEST MESSAGE *****');
⋮
/*************************************************************/
/* LOCAL COPY OF OBJECT DESCRIPTOR, MESSAGE DESCRIPTOR  */
/* AND PUT MESSAGE OPTIONS                */
/*************************************************************/
DCL 1 LMQOD LIKE MQOD;
DCL 1 LMQMD LIKE MQMD;
DCL 1 LMQPMO LIKE MQPMO;
⋮
/*************************************************************/
/* SET UP OBJECT DESCRIPTOR AS REQUIRED.         */
/*************************************************************/
LMQOD.OBJECTTYPE = MQOT_Q;
LMQOD.OBJECTNAME = QUEUE_NAME;

/*************************************************************/
/* SET UP MESSAGE DESCRIPTOR AS REQUIRED.        */
/*************************************************************/
LMQMD.MSGTYPE = MQMT_REQUEST;
LMQMD.PRIORITY = 5;
LMQMD.PERSISTENCE = MQPER_PERSISTENT;
LMQMD.REPLYTOQ = REPLY_TO_QUEUE;
LMQMD.REPLYTOQMGR = ' ';
LMQMD.MSGID = MQMI_NONE;
LMQMD.CORRELID = MQCI_NONE;
```

```
/*************************************************************/
/* SET UP PUT MESSAGE OPTIONS AS REQUIRED        */
/*************************************************************/
LMQPMO.OPTIONS = MQPMO_NO_SYNCPOINT;

/*************************************************************/
/* SET UP LENGTH OF MESSAGE BUFFER AND THE MESSAGE    */
/*************************************************************/
BUFFLEN = LENGTH(BUFFER);
BUFFER = PL1_TEST_MESSAGE;

CALL MQPUT1 (HCONN,
```

```
LMQOD,
LMQMD,
LMQPMO,
BUFFLEN,
BUFFER,
COMPCODE,
REASON);

/*************************************************************/
/* TEST THE COMPLETION CODE OF THE PUT1 CALL.      */
/* IF THE CALL HAS FAILED ISSUE AN ERROR MESSAGE SHOWING */
/* THE COMPLETION CODE AND THE REASON CODE.       */
/*************************************************************/
IF COMPCODE ¬= MQCC_OK
THEN DO;
⋮
CALL ERROR_ROUTINE;
END;
```

z/OS *acquisizione di un messaggio*

Questo esempio mostra come utilizzare la chiamata MQGET per rimuovere un messaggio da una coda.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
     %INCLUDE SYSLIB(CMQP);
     %INCLUDE SYSLIB(CMQEPP);
     ⋮
/*************************************************************/
/* WORKING STORAGE DECLARATIONS                              */
/*************************************************************/
     DCL COMPCODE                 BINARY FIXED (31);
     DCL REASON                   BINARY FIXED (31);
     DCL HCONN                    BINARY FIXED (31);
     DCL HOBJ                     BINARY FIXED (31);
     DCL BUFFLEN                  BINARY FIXED (31);
     DCL DATALEN                  BINARY FIXED (31);
     DCL BUFFER                   CHAR(80);
⋮
```

```
/*************************************************************/
/* LOCAL COPY OF MESSAGE DESCRIPTOR AND                      */
/* GET MESSAGE OPTIONS                                       */
/*************************************************************/
     DCL 1 LMQMD  LIKE MQMD;
     DCL 1 LMQGMO LIKE MQGMO;
     ⋮
/*************************************************************/
/* SET UP MESSAGE DESCRIPTOR AS REQUIRED.                    */
/* MSGID AND CORRELID IN MQMD SET TO NULLS SO FIRST      */
/* AVAILABLE MESSAGE WILL BE RETRIEVED.                      */
/*************************************************************/
     LMQMD.MSGID = MQMI_NONE;
     LMQMD.CORRELID = MQCI_NONE;

/*************************************************************/
/* SET UP GET MESSAGE OPTIONS AS REQUIRED.                   */
/*************************************************************/
     LMQGMO.OPTIONS = MQGMO_NO_SYNCPOINT;

/*************************************************************/
/* SET UP LENGTH OF MESSAGE BUFFER.                          */
/*************************************************************/
     BUFFLEN = LENGTH(BUFFER);
/*************************************************************/
/*                                                           */
/* HCONN WAS SET BY PREVIOUS MQCONN REQUEST.                 */
/* HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST.                  */
/*                                                           */
/*************************************************************/

     CALL MQGET (HCONN,
                 HOBJ,
                 LMQMD,
                 LMQGMO,
                 BUFFERLEN,
                 BUFFER,
```

```
                DATALEN,
                COMPCODE,
                REASON);

/*************************************************************/
/* TEST THE COMPLETION CODE OF THE GET CALL.                 */
/* IF THE CALL HAS FAILED ISSUE AN ERROR MESSAGE             */
/* SHOWING THE COMPLETION CODE AND THE REASON CODE.          */
/*************************************************************/
     IF COMPCODE ¬= MQCC_OK
        THEN DO;
          ⋮
          CALL ERROR_ROUTINE;
        END;
```

z/OS *Ricezione di un messaggio utilizzando l'opzione di attesa*

Questo esempio mostra come utilizzare la chiamata MQGET con l'opzione di attesa e accettare i messaggi troncati.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
     %INCLUDE SYSLIB(CMQP);
     %INCLUDE SYSLIB(CMQEPP);
     ⋮
/*************************************************************/
/* WORKING STORAGE DECLARATIONS                              */
/*************************************************************/
     DCL COMPCODE                 BINARY FIXED (31);
     DCL REASON                   BINARY FIXED (31);
     DCL HCONN                    BINARY FIXED (31);
     DCL HOBJ                     BINARY FIXED (31);
     DCL BUFFLEN                  BINARY FIXED (31);
     DCL DATALEN                  BINARY FIXED (31);
     DCL BUFFER                   CHAR(80);
     ⋮
/*************************************************************/
/* LOCAL COPY OF MESSAGE DESCRIPTOR AND GET MESSAGE          */
/* OPTIONS                                                   */
/*************************************************************/
     DCL 1 LMQMD  LIKE MQMD;
     DCL 1 LMQGMO LIKE MQGMO;
     ⋮
/*************************************************************/
/* SET UP MESSAGE DESCRIPTOR AS REQUIRED.                    */
/* MSGID AND CORRELID IN MQMD SET TO NULLS SO FIRST          */
/* AVAILABLE MESSAGE WILL BE RETRIEVED.                      */
/*************************************************************/
     LMQMD.MSGID = MQMI_NONE;
     LMQMD.CORRELID = MQCI_NONE;

/*************************************************************/
/* SET UP GET MESSAGE OPTIONS AS REQUIRED.                   */
/* WAIT INTERVAL SET TO ONE MINUTE.                          */
/*************************************************************/
     LMQGMO.OPTIONS = MQGMO_WAIT +
                      MQGMO_ACCEPT_TRUNCATED_MSG +
                      MQGMO_NO_SYNCPOINT;
     LMQGMO.WAITINTERVAL=60000;

/*************************************************************/
/* SET UP LENGTH OF MESSAGE BUFFER.                          */
/*************************************************************/
     BUFFLEN = LENGTH(BUFFER);
```

```
/*************************************************************/
/*                                                           */
/* HCONN WAS SET BY PREVIOUS MQCONN REQUEST.                 */
/* HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST.                  */
/*                                                           */
/*************************************************************/

     CALL MQGET (HCONN,
                 HOBJ,
                 LMQMD,
                 LMQGMO,
```

```
                BUFFERLEN,
                BUFFER,
                DATALEN,
                COMPCODE,
                REASON);

/*********************************************************/
/* TEST THE COMPLETION CODE OF THE GET CALL.             */
/* TAKE APPROPRIATE ACTION BASED ON COMPLETION CODE AND  */
/* REASON CODE.                                          */
/*********************************************************/

     SELECT(COMPCODE);
       WHEN (MQCC_OK) DO;   /* GET WAS SUCCESSFUL */
       ⋮
       END;
       WHEN (MQCC_WARNING) DO;
         IF REASON = MQRC_TRUNCATED_MSG_ACCEPTED
           THEN DO;         /* GET WAS SUCCESSFUL */
           ⋮
           END;
           ELSE DO;
             ⋮
             CALL ERROR_ROUTINE;
           END;
       END;
       WHEN (MQCC_FAILED) DO;
           ⋮
           CALL ERROR_ROUTINE;
         END;
       END;
       OTHERWISE;
     END;
```

z/OS *Ricezione di un messaggio utilizzando la segnalazione*

Un estratto di codice che illustra come utilizzare la chiamata MQGET con la segnalazione.

La segnalazione è disponibile solo con IBM MQ for z/OS .

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
     %INCLUDE SYSLIB(CMQP);
     %INCLUDE SYSLIB(CMQEPP);
     :
/*********************************************************/
/* WORKING STORAGE DECLARATIONS                          */
/*********************************************************/
     DCL COMPCODE                 BINARY FIXED (31);
     DCL REASON                   BINARY FIXED (31);
     DCL HCONN                    BINARY FIXED (31);
     DCL HOBJ                     BINARY FIXED (31);
     DCL DATALEN                  BINARY FIXED (31);
     DCL BUFFLEN                  BINARY FIXED (31);
     DCL BUFFER                   CHAR(80);
     ⋮
     DCL ECB_FIXED           FIXED BIN(31);
     DCL 1 ECB_OVERLAY BASED(ADDR(ECB_FIXED)),
           3 ECB_WAIT    BIT,
           3 ECB_POSTED  BIT,
           3 ECB_FLAG3_8 BIT(6),
           3 ECB_CODE  PIC'999';
     ⋮
/*********************************************************/
/* LOCAL COPY OF MESSAGE DESCRIPTOR AND GET MESSAGE      */
/* OPTIONS                                               */
/*********************************************************/
     DCL 1 LMQMD  LIKE MQMD;
     DCL 1 LMQGMO LIKE MQGMO;
     ⋮
/*********************************************************/
/* CLEAR ECB FIELD.                                      */
/*********************************************************/
     ECB_FIXED = 0;
     ⋮
/*********************************************************/
/* SET UP MESSAGE DESCRIPTOR AS REQUIRED.                */
/* MSGID AND CORRELID IN MQMD SET TO NULLS SO FIRST      */
/* AVAILABLE MESSAGE WILL BE RETRIEVED.                  */
```

```
/*********************************************************/
     LMQMD.MSGID = MQMI_NONE;
     LMQMD.CORRELID = MQCI_NONE;
/*********************************************************/
/* SET UP GET MESSAGE OPTIONS AS REQUIRED.               */
/* WAIT INTERVAL SET TO ONE MINUTE.                      */
/*********************************************************/
     LMQGMO.OPTIONS = MQGMO_SET_SIGNAL +
                      MQGMO_NO_SYNCPOINT;
     LMQGMO.WAITINTERVAL=60000;
     LMQGMO.SIGNAL1 = ADDR(ECB_FIXED);
```

```
/*********************************************************/
/* SET UP LENGTH OF MESSAGE BUFFER.                      */
/* CALL MESSAGE RETRIEVAL ROUTINE.                       */
/*********************************************************/
     BUFFLEN = LENGTH(BUFFER);
     CALL GET_MSG;

/*********************************************************/
/* TEST THE COMPLETION CODE OF THE GET CALL.             */
/* TAKE APPROPRIATE ACTION BASED ON COMPLETION CODE AND  */
/* REASON CODE.                                          */
/*********************************************************/

     SELECT;
       WHEN ((COMPCODE = MQCC_OK) &
             (REASON = MQCC_NONE)) DO
         ⋮
         CALL MSG_ROUTINE;
         ⋮
       END;
       WHEN ((COMPCODE = MQCC_WARNING) &
             (REASON = MQRC_SIGNAL_REQUEST_ACCEPTED)) DO;
         ⋮
         CALL DO_WORK;
         ⋮
       END;
       WHEN  ((COMPCODE = MQCC_FAILED) &
             (REASON = MQRC_SIGNAL_OUTSTANDING)) DO;
         ⋮
         CALL DO_WORK;
         ⋮
       END;
       OTHERWISE DO;        /* FAILURE CASE */
/*********************************************************/
/* ISSUE AN ERROR MESSAGE SHOWING THE COMPLETION CODE    */
/* AND THE REASON CODE.                                  */
/*********************************************************/
          ⋮
          CALL ERROR_ROUTINE;
          ⋮
       END;
     END;
     ⋮
```

```
 DO_WORK: PROC;
     ⋮
     IF ECB_POSTED
        THEN DO;
          SELECT(ECB_CODE);
            WHEN(MQEC_MSG_ARRIVED) DO;
              ⋮
              CALL GET_MSG;
              ⋮
            END;
            WHEN(MQEC_WAIT_INTERVAL_EXPIRED) DO;
              ⋮
              CALL NO_MSG;
              ⋮
            END;
            OTHERWISE DO;        /* FAILURE CASE */
/*********************************************************/
/* ISSUE AN ERROR MESSAGE SHOWING THE COMPLETION CODE  */
/* AND THE REASON CODE.                                  */
/*********************************************************/
              ⋮
```

```
              CALL ERROR_ROUTINE;
              :
            END;

           END;

        END;
     :
 END DO_WORK;

 GET_MSG: PROC;
```

```
/*******************************************************/
/*                                                     */
/* HCONN WAS SET BY PREVIOUS MQCONN REQUEST.           */
/* HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST.            */
/* MD AND GMO SET UP AS REQUIRED.                      */
/*                                                     */
/*******************************************************/

     CALL MQGET (HCONN,
                 HOBJ,
                 LMQMD,
                 LMQGMO,
                 BUFFLEN,
                 BUFFER,
                 DATALEN,
                 COMPCODE,
                 REASON);

 END GET_MSG;

 NO_MSG: PROC;
 :
 END NO_MSG;
```

z/OS *Richiesta di informazioni sugli attributi di un oggetto*

Questo esempio dimostra come utilizzare la chiamata MQINQ per richiedere informazioni sugli attributi di una coda.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
     %INCLUDE SYSLIB(CMQP);
     %INCLUDE SYSLIB(CMQEPP);
     :
/*******************************************************/
/* WORKING STORAGE DECLARATIONS                        */
/*******************************************************/
     DCL COMPCODE                  BINARY FIXED (31);
     DCL REASON                    BINARY FIXED (31);
     DCL HCONN                     BINARY FIXED (31);
     DCL HOBJ                      BINARY FIXED (31);
     DCL OPTIONS                   BINARY FIXED (31);
     DCL SELECTORCOUNT             BINARY FIXED (31);
     DCL INTATTRCOUNT              BINARY FIXED (31);
     DCL 1 SELECTOR_TABLE,
         3 SELECTORS(5)            BINARY FIXED (31);
     DCL 1 INTATTR_TABLE,
         3 INTATTRS(5)             BINARY FIXED (31);
     DCL CHARATTRLENGTH            BINARY FIXED (31);
     DCL CHARATTRS                 CHAR(100);
     :

/*******************************************************/
/* SET VARIABLES FOR INQUIRE CALL                      */
/* INQUIRE ON THE CURRENT QUEUE DEPTH                  */
/*******************************************************/

     SELECTORS(01) = MQIA_CURRENT_Q_DEPTH;

     SELECTORCOUNT  = 1;
     INTATTRCOUNT   = 1;

     CHARATTRLENGTH = 0;
/*******************************************************/
```

```
/*                                                            */
/* HCONN WAS SET BY PREVIOUS MQCONN REQUEST.                  */
/* HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST.                   */
/*                                                            */
/************************************************************/
     CALL MQINQ (HCONN,
                 HOBJ,
                 SELECTORCOUNT,
                 SELECTORS,
                 INTATTRCOUNT,
                 INTATTRS,
                 CHARATTRLENGTH,
                 CHARATTRS,
                 COMPCODE,
                 REASON);
```

```
/************************************************************/
/* TEST THE COMPLETION CODE OF THE INQUIRE CALL.            */
/* IF THE CALL HAS FAILED ISSUE AN ERROR MESSAGE SHOWING    */
/* THE COMPLETION CODE AND THE REASON CODE.                 */
/************************************************************/
     IF COMPCODE ¬= MQCC_OK
        THEN DO;
          :
          CALL ERROR_ROUTINE;
        END;
```

z/OS *Impostazione degli attributi di una coda*

Questo esempio illustra come utilizzare la chiamata MQSET per modificare gli attributi di una coda.

Questo estratto non viene estratto dalle applicazioni di esempio fornite con IBM MQ.

```
     %INCLUDE SYSLIB(CMQP);
     %INCLUDE SYSLIB(CMQEPP);
     :
/************************************************************/
/* WORKING STORAGE DECLARATIONS                             */
/************************************************************/
     DCL COMPCODE                  BINARY FIXED (31);
     DCL REASON                    BINARY FIXED (31);
     DCL HCONN                     BINARY FIXED (31);
     DCL HOBJ                      BINARY FIXED (31);
     DCL OPTIONS                   BINARY FIXED (31);
     DCL SELECTORCOUNT             BINARY FIXED (31);
     DCL INTATTRCOUNT              BINARY FIXED (31);
     DCL 1 SELECTOR_TABLE,
         3 SELECTORS(5)            BINARY FIXED (31);
     DCL 1 INTATTR_TABLE,
         3 INTATTRS(5)             BINARY FIXED (31);
     DCL CHARATTRLENGTH            BINARY FIXED (31);
     DCL CHARATTRS                 CHAR(100);
     :

/************************************************************/
/* SET VARIABLES FOR SET CALL                               */
/* SET GET AND PUT INHIBITED                                */
/************************************************************/

     SELECTORS(01) = MQIA_INHIBIT_GET;
     SELECTORS(02) = MQIA_INHIBIT_PUT;

     INTATTRS(01) = MQQA_GET_INHIBITED;
     INTATTRS(02) = MQQA_PUT_INHIBITED;

     SELECTORCOUNT  = 2;
     INTATTRCOUNT   = 2;

     CHARATTRLENGTH = 0;
```

```
/************************************************************/
/*                                                          */
/* HCONN WAS SET BY PREVIOUS MQCONN REQUEST.                */
/* HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST.                 */
/*                                                          */
```

```
/*********************************************************/
     CALL MQSET (HCONN,
                 HOBJ,
                 SELECTORCOUNT,
                 SELECTORS,
                 INTATTRCOUNT,
                 INTATTRS,
                 CHARATTRLENGTH,
                 CHARATTRS,
                 COMPCODE,
                 REASON);

/*********************************************************/
/* TEST THE COMPLETION CODE OF THE SET CALL.             */
/* IF THE CALL HAS FAILED ISSUE AN ERROR MESSAGE SHOWING */
/* THE COMPLETION CODE AND THE REASON CODE.              */
/*********************************************************/
     IF COMPCODE ¬= MQCC_OK
        THEN DO;
          ⋮
          CALL ERROR_ROUTINE;
        END;
```

Costanti

Utilizzare le informazioni di riferimento fornite in questa sezione per svolgere le attività utili alle proprie esigenze aziendali.

File IBM MQ COPY, intestazione, inclusione e modulo

Queste informazioni sono informazioni sull'interfaccia di programmazione di uso generale.

Questa sezione contiene informazioni che consentono di utilizzare MQI per diversi linguaggi di programmazione, come riportato di seguito.

File di intestazione C

I file di intestazione vengono forniti per consentire la scrittura di programmi di applicazione C che utilizzano MQI.

I file di intestazione C sono riepilogati nella seguente tabella:

Tabella 1. File di intestazione C - richiama prototipi, tipi di dati, codici di ritorno, costanti e strutture

Nome file	Descrizione	IBM i	Sistemi AIX and Linux®	Windows	z/OS
Chiama prototipi, tipi di dati, codici di ritorno, costanti e strutture					
CMQC	Definizioni MQI	C	C	C	C
CMQBC	Definizioni MQAI	C	C	C	
CMQEC	Definizione punti di ingresso interfaccia (include CMQC, CMQXC e CMQZC)		C	C	
CMQCFC	Definizioni PCF	C	C	C	C
CMQPSC	Definizioni di pubblicazione / sottoscrizione	C	C	C	C
CMQXC	Definizioni di canale e uscita	C	C	C	C
CMQZC	Definizioni dei servizi installabili	C	C	C	
Chiave: C= File forniti					

File COBOL COPY

Vengono forniti diversi file COPY per consentire la scrittura di programmi applicativi COBOL che utilizzano MQI.

Tabella 2. File di copia COBOL - codici di ritorno, costanti e strutture

Nome file	Descrizione	IBM i	AIX and Linux	Windows	z/OS
Codici di ritorno e costanti					
CMQX	Definizioni MQI	V	V	V	V
CMQCFX	Definizioni PCF	V	V	V	V
CMQPSx	Definizioni di pubblicazione / sottoscrizione	V	V	V	V
CMQX	Definizioni di canale e uscita	V	V	V	V
Strutture					
CMQAIRx	MQAIR - Record informazioni di autenticazione		V L	V L	
CMQBOX	MQBO - Opzioni di inizio	V L	V L	V L	
CMQCDX	MQCD - Definizione canale	V L	V L	V L	V L
CMQCFBFx	MQCFBF - Parametro filtro stringa byte PCF	V L	V L	V L	V L
CMQCFBSx	MQCFBS - parametro stringa byte PCF	V L	V L	V L	V L
CMQCFGRx	MQCFGR - Parametro gruppo PCF	V L	V L	V L	V L
CMQCFHx	Intestazione MQCFH - PCF	V L	V L	V L	V L
CMQCFIFx	MQCFIF - Parametro filtro numero intero PCF	V L	V L	V L	V L
CMQCFILx	MQCFIL - Parametro elenco numeri interi PCF	V L	V L	V L	V L
CMQCFINx	MQCFIN - Parametro intero PCF	V L	V L	V L	V L
CMQCFSFx	MQCFSF - Parametro filtro stringa PCF	V L	V L	V L	V L
CMQCFSLx	MQCFSL - Parametro elenco stringhe PCF	V L	V L	V L	V L
CMQCFSTx	MQCFST - parametro stringa PCF	V L	V L	V L	V L
CMQCFXLx	MQCFIL64 - parametro elenco numeri interi PCF a 64 bit	V L	V L	V L	V L
CMQCFXNx	MQCFIN64 - Parametro intero PCF a 64 bit	V L	V L	V L	V L
CMQCHRVx	MQCHARV - Stringa di lunghezza variabile	V L	V L	V L	V L
CMQCIHx	Intestazione MQCIH - CICS bridge	V L	V L	V L	V L
CMQCNOx	MQCNO - Opzioni di connessione	V L	V L	V L	V L

Tabella 2. File di copia COBOL - codici di ritorno, costanti e strutture (Continua)					
Nome file	**Descrizione**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
CMQCSPX	MQCSP - Parametri di sicurezza	V L	V L	V L	V L
CMQCXPx	MQCXP - Parametri uscita canale	V L			V L
CMQDHx	MQDH - Intestazione di distribuzione	V L	V L	V L	V L
CMQDLHx	MQDLH - Intestazione non instradabile	V L	V L	V L	V L
CMQDXPx	MQDXP - Parametri di uscita conversione dati	V L		V L	
CMQEPHx	MQEPH - Intestazione PCF integrata	V L	V L	V L	V L
CMQGMOx	MQGMO - Richiama opzioni messaggio	V L	V L	V L	V L
CMQIIHx	MQIIH - Intestazione informazioni IMS	V L	V L	V L	V L
CMQMDx	MQMD - Descrittore messaggi	V L	V L	V L	V L
CMQMD1x	MQMD1 - Descrittore messaggi versione 1	V L	V L	V L	V L
CMQMD2x	MQMD2 - Descrittore del messaggio versione 2	V L	V L	V L	V L
CMQMDEx	MQMDE - Descrittore messaggi esteso	V L	V L	V L	V L
CMQODx	MQOD - Descrittore oggetto	V L	V L	V L	V L
CMQORx	MQOR - Record oggetto	V L	V L	V L	V L
CMQPMOX	MQPMO - Opzioni inserimento messaggio	V L	V L	V L	V L
CMQRFHx	MQRFH - Regole e intestazione di formattazione	V L	V L	V L	V L
CMQRFH2x	MQRFH2 - Regole e intestazione di formattazione 2	V L	V L	V L	V L
CMQRMHx	MQRMH - Intestazione messaggio di riferimento	V L	V L	V L	V L
CMQRRX	MQRR - Record risposta	V L	V L	V L	
CMQSCOx	MQSCO - Opzioni di configurazione TLS		V L	V L	
CMQTMX	MQTM - Messaggio trigger	V L		V L	V L
CMQTMCx	MQTMC - Carattere messaggio trigger	V L	V L		
CMQTMC2x	MQTMC2 - Messaggio di attivazione 2 caratteri	V L	V L	V L	V L

Tabella 2. File di copia COBOL - codici di ritorno, costanti e strutture (Continua)					
Nome file	**Descrizione**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
CMQWIHx	MQWIH - Intestazione informazioni di lavoro	V L	V L	V L	V L
CMQXQHx	MQXQH - Intestazione coda di trasmissione	V L	V L	V L	V L
Chiave: • File con valori iniziali forniti, x = V • File senza valori iniziali forniti, x = L					

z/OS *PL/I include files*

A number of INCLUDE files are provided for the PL/I programming language. These files are available on z/OS only.

Table 3. PL/I include files - data types, return codes, constants, and structures					
File name	**Description**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
Data types, return codes, constants, and structures					
CMQP	MQI definitions				P
CMQCFP	PCF definitions				P
CMQEPP	Entry point definitions				P
CMQPSP	Publish/Subscribe definitions				P
CMQXP	Channel and exit definitions				P
Key: P= File provided					

IBM i *File di copia RPG*

I file RPG COPY vengono forniti per il linguaggio di programmazione RPG. Questi file sono disponibili solo su IBM i.

Tabella 4. File di copia RPG - codici di ritorno, costanti e strutture					
Nome file	**Descrizione**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
Codici di ritorno e costanti					
CMQX	Definizioni MQI	R			
CMQCFX	Definizioni PCF	G			
CMQPSx	Definizioni di pubblicazione / sottoscrizione	G			
CMQX	Definizioni di canale e uscita	R			
Strutture					
CMQBOX	MQBO - Opzioni di inizio	G H			
CMQCDX	MQCD - Definizione canale	G H R			

Tabella 4. File di copia RPG - codici di ritorno, costanti e strutture (Continua)					
Nome file	**Descrizione**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
CMQCFBFx	MQCFBF - Parametro filtro stringa byte PCF	G H			
CMQCFBSx	MQCFBS - parametro stringa byte PCF	G H			
CMQCFGRx	MQCFGR - Parametro gruppo PCF	G H			
CMQCFHx	Intestazione MQCFH - PCF	G H			
CMQCFIFx	MQCFIF - Parametro filtro numero intero PCF	G H			
CMQCFILx	MQCFIL - Parametro elenco numeri interi PCF	G H			
CMQCFINx	MQCFIN - Parametro intero PCF	G H			
CMQCFSFx	MQCFSF - Parametro filtro stringa PCF	G H			
CMQCFSLx	MQCFSL - Parametro elenco stringhe PCF	G H			
CMQCFSTx	MQCFST - parametro stringa PCF	G H			
CMQCFXLx	MQCFIL64 - parametro elenco numeri interi PCF a 64 bit	G H			
CMQCFXNx	MQCFIN64 - Parametro intero PCF a 64 bit	G H			
CMQCARVX	MQCHARV - Stringa di lunghezza variabile	G H			
CMQCIHx	Intestazione MQCIH - CICS bridge	G H			
CMQCNOx	MQCNO - Opzioni di connessione	G H			
CMQCSPX	MQCSP - Parametri di sicurezza	G H			
CMQCXPx	MQCXP - Parametri uscita canale	G H R			
CMQDHx	MQDH - Intestazione di distribuzione	G H R			
CMQDLHx	MQDLH - Intestazione non instradabile	G H R			
CMQDXPx	MQDXP - Parametri di uscita conversione dati	G H R			
CMQEPHx	MQEPH - Intestazione PCF integrata	G H			
CMQGMOx	MQGMO - Richiama opzioni messaggio	G H R			
CMQIIHx	MQIIH - Intestazione informazioni IMS	G H R			
CMQMDx	MQMD - Descrittore messaggi	G H R			

Tabella 4. File di copia RPG - codici di ritorno, costanti e strutture (Continua)

Nome file	Descrizione	IBM i	AIX and Linux	Windows	z/OS
CMQMD1x	MQMD1 - Descrittore messaggi versione 1	G H R			
CMQMD2x	MQMD2 - Descrittore del messaggio versione 2	G H			
CMQMDEx	MQMDE - Descrittore messaggi esteso	G H R			
CMQODx	MQOD - Descrittore oggetto	G H R			
CMQORx	MQOR - Record oggetto	G H R			
CMQPMOX	MQPMO - Opzioni inserimento messaggio	G H R			
CMQPXPx	MQPXP - Parametri di uscita di instradamento di pubblicazione / sottoscrizione	G H			
CMQRFHx	MQRFH - Regole e intestazione di formattazione	G H			
CMQRFH2x	MQRFH2 - Regole e intestazione di formattazione 2	G H			
CMQRMHx	MQRMH - Intestazione messaggio di riferimento	G H R			
CMQRRX	MQRR - Record risposta	G H R			
CMQTMX	MQTM - Messaggio trigger	G H R			
CMQTMCx	MQTMC - Carattere messaggio trigger	G H R			
CMQTMC2x	MQTMC2 - Messaggio di attivazione 2 caratteri	G H R			
CMQWIHx	MQWIH - Intestazione informazioni di lavoro	G H			
CMQXQHx	MQXQH - Intestazione coda di trasmissione	G H R			

Chiave:

- File per collegamento statico, inizializzato, fornito x = G
- File per il collegamento statico, non inizializzato, fornito x = H
- File per collegamento dinamico, inizializzato, fornito, x = R

Windows *File del modulo Visual Basic*

I file di intestazione (o modulo) vengono forniti per consentire la scrittura di programmi applicativi Visual Basic che utilizzano MQI. Questi file di intestazione vengono forniti solo in versioni a 32 bit.

Tabella 5. File del modulo Visual Basic - dichiarazioni di chiamata, tipi di dati, codici di ritorno, costanti e strutture					
Nome file	**Descrizione**	**IBM i**	**Sistemi AIX and Linux**	**Windows**	**z/OS**
Dichiarazioni di chiamate, tipi di dati, codici di ritorno, costanti e strutture					
CMQB	Definizioni MQI			B	
CMQB	Definizioni MQAI			B	
CMQCFB	Definizioni PCF			B	
CMQXB	Definizioni di canale e uscita			B	
Chiave: B= File fornito					

z/OS *z/OS Assembler COPY files*

Various COPY files are provided to help you write z/OS Assembler application programs that use the MQI.

Table 6. z/OS Assembler copy files - data types, return codes, constants, and structures					
File name	**Description**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
Data types, return codes, and constants					
CMQA	MQI definitions				A
CMQCFA	PCF definitions				A
CMQPSA	Publish/Subscribe definitions				A
CMQVERA	Structure version control				A
CMQXA	Channel and exit definitions				A
Structures					
CMQCDA	MQCD - Channel definition				
CMQCFBFA	MQCFBF - PCF byte string filter parameter				
CMQCFBSA	MQCFBS - PCF byte string parameter				A
CMQCFGRA	MQCFGR - PCF group parameter				A
CMQCFHA	MQCFH - PCF header				A
CMQCFIFA	MQCFIF - PCF integer filter parameter				A
CMQCFILA	MQCFIL - PCF integer list parameter				A
CMQCFINA	MQCFIN - PCF integer parameter				A
CMQCFSFA	MQCFSF - PCF string filter parameter				A
CMQCFSLA	MQCFSL - PCF string list parameter				A
CMQCFSTA	MQCFST - PCF string parameter				A

Table 6. z/OS Assembler copy files - data types, return codes, constants, and structures (continued)					
File name	**Description**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
CMQCFXLA	MQCFIL64 - PCF 64-bit integer list parameter				A
CMQCFXNA	MQCFIN64 - PCF 64-bit integer parameter				A
CMQCHARVA	MQCHARV - Variable length string				A
CMQCIHA	MQCIH - CICS bridge header				A
CMQCNOA	MQCNO - Connect options				A
CMQCSPA	MQCSP - Security parameters				A
CMQCXPA	MQCXP - Channel exit parameters				A
CMQDHA	MQDH - Distribution header				A
CMQDLHA	MQDLH - Dead-letter header				A
CMQDXPA	MQDXP - Data conversion exit parameters				A
CMQEPHA	MQEPH - Embedded PCF header				A
CMQGMOA	MQGMO - Get message options				A
CMQIIHA	MQIIH - IMS information header				A
CMQMDA	MQMD - Message descriptor				A
CMQMD1A	MQMD1 - Message descriptor version 1				A
CMQMD2A	MQMD2 - Message descriptor version 2				A
CMQMDEA	MQMDE - Message descriptor extended				A
CMQODA	MQOD - Object descriptor				A
CMQORA	MQOR - Object record				A
CMQPMOA	MQPMO - Put message options				A
CMQRFHA	MQRFH - Rules and formatting header				A
CMQRFH2A	MQRFH2 - Rules and formatting header 2				A
CMQRMHA	MQRMH - Reference message header				A
CMQTMA	MQTM - Trigger message				A
CMQTMC2A	MQTMC2 - Trigger message 2 character				A
CMQWCRA	MQWCR - Cluster workload cluster record				A

Table 6. z/OS Assembler copy files - data types, return codes, constants, and structures (continued)					
File name	**Description**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
CMQWDRA	MQWDR - Cluster workload destination record				A
CMQWDR1A	MQWDR1 - Cluster workload destination record version 1				A
CMQWDR2A	MQWDR2 - Cluster workload destination record version 2				A
CMQWIHA	MQWIH - Work information header				A
CMQWQRA	MQWQR - Cluster workload queue record				A
CMQWQR1A	MQWQR1 - Cluster workload queue record version 1				A
CMQWQR2A	MQWQR2 - Cluster workload queue record version 2				A
CMQWXPA	MQWXP - Cluster workload exit parameters				A
CMQWXP1A	MQWXP1 - Cluster workload exit parameters version 1				A
CMQWXP2A	MQWXP2 - Cluster workload exit parameters version 2				A
CMQWXP3A	MQWXP3 - Cluster workload exit parameters version 3				A
CMQXPA	MQXP - CICS API-crossing exit parameters				A
CMQXQHA	MQXQH - Transmission queue header				A
CMQXWDA	MQXWD - Exit wait descriptor				A
Key: A= File provided					

MQ_ * (Lunghezze stringa)

Tabella 7. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQ_ABEND_CODE_LENGTH	4	X'00000004'
MQ_ACCOUNTING_TOKEN_LENGTH	32	X'00000020'
LUNGHEZZA_NOME_FUNZIONE_APPL_MQ	10	X'0000000A'
LUNGHEZZA_DATI_IDENTITÀ_APPL_MQ	32	X'00000020'
LUNGHEZZA_NOME_APPL_MQ	28	X'0000001C'
LUNGHEZZA_DATI_ORIGINE_APPL_MQ	4	X'00000004'
LUNGHEZZA_TAG_APPL_MQ_	28	X'0000001C'
LUNGHEZZA_SUFFISSO_ARM_MQ	2	X'00000002'

Tabella 7. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
LUNGHEZZA_ID_ATTENZIONE_MQ	4	X'00000004'
LUNGHEZZA_NOME_CONN_AUTH_MQ_	264	X'00000108'
MQ_AUTH_INFO_DESC_LENGTH	64	X'00000040'
LUNGHEZZA_NOME_INFORMAZIONI_MQ_AUTH_	48	X'00000030'
LUNGHEZZA_URL_OCSP_AUTH_MQ_AUTH_INFO_	256	X'00000100'
LUNGHEZZA_AUTENTICATORE_MQ	8	X'00000008'
LUNGHEZZA_CATALOGO_RIORG_AUTO_MQ_	44	X'0000002C'
MQ_AUTO_REORG_TIME_LENGTH	4	X'00000004'
LUNGHEZZA_ID_INTERFACCIA_BATCH	8	X'00000008'
LUNGHEZZA_NOME_BRIDGE	24	X'00000018'
MQ_CANCEL_CODE_LENGTH	4	X'00000004'
MQ_CF_STRUC_DESC_LENGTH	64	X'00000040'
LUNGHEZZA_NOME_STRUTTURA_CF_MQ	12	X'0000000C'
MQ_CHANNEL_DATE_LENGTH	12	X'0000000C'
LUNGHEZZA_DESC_CANALE_MQ.	64	X'00000040'
LUNGHEZZA_NOME_CANALE_MQ	20	X'00000014'
MQ_CHANNEL_TIME_LENGTH	8	X'00000008'
LUNGHEZZA_PARO_SERVIZIO_MQ_CHINIT_	32	X'00000020'
LUNGHEZZA_NOME_FILE_MQ_CICS	8	X'00000008'
LUNGHEZZA_ID_CLIENT_MQ	23	X'00000017'
LUNGHEZZA_NOME_CLUSTER_MQ_	48	X'00000030'
LUNGHEZZA_NOME_CONN_MQ	264	X'00000108'
MQ_CONN_TAG_LENGTH	128	X'00000080'
LUNGHEZZA_ID_CONNESSIONE_MQ	24	X'00000018'
LUNGHEZZA_ID_CORRELATA	24	X'00000018'
CREAZIONE MQ_DATE_LENGTH	12	X'0000000C'
MQ_CREATION_TIME_LENGTH	8	X'00000008'
MQ_DATE_LENGTH	12	X'0000000C'
LUNGHEZZA_NOME_DISTINGUITO_MQ_	1024	X'00000400'
LUNGHEZZA_NOME_GRUPPO_DNS	18	X'00000012'
MQ_EXIT_DATA_LENGTH	32	X'00000020'
LUNGHEZZA_NOME_INFORMAZIONI_ESIT_MQ	48	X'00000030'
LUNGHEZZA_NOME_ESSITA_MQ	(value differs by platform or version)	
MQ_EXIT_PD_AREA_LENGTH	48	X'00000030'
LUNGHEZZA_AREA_UTENTE_EXIT_MQ_	16	X'00000010'
LUNGHEZZA_FACILITAZIONE_MQ	8	X'00000008'
LUNGHEZZA_FACILITA_MQ_SIMILE	4	X'00000004'
MQ_FORMAT_LENGTH	8	X'00000008'
LUNGHEZZA_FUNZIONE_MQ	4	X'00000004'

Tabella 7. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
LUNGHEZZA_ID_GRUPPO_MQ.	24	X'00000018'
MQ_LDAP_PASSWORD_LENGTH	32	X'00000020'
LUNGHEZZA_NOME_LISTENER_MQ	48	X'00000030'
LUNGHEZZA_DESC_LISTENER_MQ	64	X'00000040'
MQ_LOCAL_ADDRESS_LENGTH	48	X'00000030'
MQ_LTERM_OVERRIDE_LENGTH	8	X'00000008'
LUNGHEZZA_NOME_MQ_LU_	8	X'00000008'
LUNGHEZZA_LARGHEZZA_MQ	16	X'00000010'
MQ_MAX_EXIT_NAME_LENGTH	128	X'00000080'
MQ_MAX_MCA_USER_ID_LENGTH	64	X'00000040'
MQ_MAX_PROPERTY_NAME_LENGTH	4095	X'00000FFF'
LUNGHEZZA_MAX_ID_UTENTE	64	X'00000040'
MQ_MCA_JOB_NAME_LENGTH	28	X'0000001C'
MQ_MCA_NAME_LENGTH	20	X'00000014'
MQ_MCA_USER_DATA_LENGTH	32	X'00000020'
MQ_MCA_USER_ID_LENGTH	(value differs by platform or version)	(value differs by platform or version)
MQ_MFS_MAP_NAME_LENGTH	8	X'00000008'
LUNGHEZZA_NOME_MODALITÀ_MQ.	8	X'00000008'
MQ_MSG_HEADER_LENGTH	4000	X'00000FA0'
ID_MQ_MSG_LENGTH	24	X'00000018'
MQ_MSG_TOKEN_LENGTH	16	X'00000010'
LUNGHEZZA_DESC_NOME_MQ_NAMELIST_	64	X'00000040'
LUNGHEZZA_NOME_ELENCO_NOMI_MQ	48	X'00000030'
LUNGHEZZA_NOME_ISTANZA_MQ_NHA_	48	X'00000030'
LUNGHEZZA_ID_ISTANZA_OGGETTO_MQ	24	X'00000018'
LUNGHEZZA_NOME_OGGETTO_MQ	48	X'00000030'
LUNGHEZZA_MQ_PASS_TICKET_APPL_	8	X'00000008'
MQ_PASSWORD_LENGTH	12	X'0000000C'
LUNGHEZZA_ID_APPL_PROCESSO_MQ	256	X'00000100'
LUNGHEZZA_DESC_PROCESSO_MQ	64	X'00000040'
LUNGHEZZA_DATI_INVIO_PROCESSO	128	X'00000080'
LUNGHEZZA_NOME_PROCESSO_MQ	48	X'00000030'
LUNGHEZZA_DATI_PROCESSO_MQ	128	X'00000080'
LUNGHEZZA_NOME_PROGRAMMA_MQ	20	X'00000014'
LUNGHEZZA_NOME_APPL_MQ_PUT_	28	X'0000001C'
LUNGHEZZA_DATA_PUT_MQ	8	X'00000008'
LUNGHEZZA_TEMPO_PUT_MQ	8	X'00000008'
LUNGHEZZA_DESC_Q_MQ	64	X'00000040'
LUNGHEZZA_Q_MGR_DESC_MQ	64	X'00000040'

Tabella 7. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQ_Q_MGR_IDENTIFIER_LENGTH	48	X'00000030'
LUNGHEZZA_NOME_MQ_Q_MGR_	48	X'00000030'
LUNGHEZZA_NOME_Q_MQ	48	X'00000030'
LUNGHEZZA_NOME_QSG_MQ	4	X'00000004'
MQ_REMOTE_SYS_ID_LENGTH	4	X'00000004'
LUNGHEZZA_ID_SICUREZZA_MQ	40	X'00000028'
MQ_SELECTOR_LENGTH	10240	X'00002800'
LUNGHEZZA_ARG_SERVIZIO_MQ.	255	X'000000FF'
LUNGHEZZA_COMANDI_SERVIZIO_MQ	255	X'000000FF'
LUNGHEZZA_DESC_SERVIZIO_MQ	64	X'00000040'
LUNGHEZZA_NOME_SERVIZIO_MQ	32	X'00000020'
LUNGHEZZA_PERCORSO_SERVIZIO_MQ	255	X'000000FF'
LUNGHEZZA_PASSO_SERVIZIO_MQ.	8	X'00000008'
Lunghezza nome connessione MQ_SHORT_CONN_	20	X'00000014'
LUNGHEZZA_DNAME_MQ_BREVE	256	X'00000100'
MQ_SSL_CIPHER_SPEC_LENGTH	32	X'00000020'
MQ_SSL_CRYPTO_HARDWARE_LENGTH	256	X'00000100'
LUNGHEZZA_STAGE_HANDSHAKE_SSL_MQ_	32	X'00000020'
MQ_SSL_KEY_LIBRARY_LENGTH	44	X'0000002C'
MQ_SSL_KEY_MEMBER_LENGTH	8	X'00000008'
MQ_SSL_KEY_REPOSITORY_LENGTH	256	X'00000100'
MQ_SSL_PEER_NAME_LENGTH	1024	X'00000400'
MQ_SSL_SHORT_PEER_NAME_LENGTH	256	X'00000100'
LUNGHEZZA_CODICE_INIZIALE_MQ	4	X'00000004'
LUNGHEZZA_DESC_CLASSE_ARCHIVIAZIONE_MQ	64	X'00000040'
LUNGHEZZA_CLASSE_ARCHIVIAZIONE_MQ	8	X'00000008'
LUNGHEZZA_IDENTITÀ_SOTTO_MQ_	128	X'00000080'
LUNGHEZZA_PUNTO_SECONDARIO_MQ_	128	X'00000080'
MQ_SUITE_B_128_BIT	2	X'00000002'
MQ_SUITE_B_192_BIT	4	X'00000004'
MQ_SUITE_B_NONE	1	X'00000001'
MQ_SUITE_B_NON_DISPONIBILE	0	X'00000000'
MQ_TCP_NAME_LENGTH	8	X'00000008'
MQ_ORA_LENGTH	8	X'00000008'
LUNGHEZZA_DESC_TOPICA_MQ	64	X'00000040'
LUNGHEZZA_NOME_ARGOMENTO_MQ_	48	X'00000030'
LUNGHEZZA_STR_ARGOMENTO_MQ_	10240	X'00002800'
MQ_TOTAL_EXIT_DATA_LENGTH	999	X'000003E7'
MQ_TOTAL_EXIT_NAME_LENGTH	999	X'000003E7'

Tabella 7. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
LUNGHEZZA_NOME_TP	64	X'00000040'
LUNGHEZZA_NOME_TPIPE_MQ	8	X'00000008'
LUNGHEZZA_ID_ISTANZA_TRAN_MQ.	16	X'00000010'
ID_TRANSAZIONE MQ_LENGTH	4	X'00000004'
LUNGHEZZA_DATI_TRIGGER_MQ	64	X'00000040'
MQ_LUNGHEZZA_NOME_PROGRAMMA_TRIGGER	8	X'00000008'
LUNGHEZZA_MQ_TRIGGER_TERM_ID_	4	X'00000004'
LUNGHEZZA_ID_TRIGGER_MQ_	4	X'00000004'
LUNGHEZZA_ID_UTENTE_MQ	12	X'0000000C'
LUNGHEZZA_VERSIONE_MQ	8	X'00000008'
MQ_XCF_NOME_GRUPPO_LUNGHEZZA	8	X'00000008'
MQ_XCF_MEMBER_NAME_LENGTH	16	X'00000010'

*MQ_ * (Lunghezza stringa formato comando)*

Tabella 8. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQ_ARCHIVE_PFX_LENGTH	36	X'00000024'
LUNGHEZZA_UNITÀ_ARCHIVIAZIONE_MQ.	8	X'00000008'
MQ_ASID_LENGTH	4	X'00000004'
LUNGHEZZA_NOME_PROFILO_AUTENTICAZIONE	48	X'00000030'
MQ_CF_LEID_LENGTH	12	X'0000000C'
MQ_COMMAND_MQSC_LENGTH	32768	X'00008000'
LUNGHEZZA_NOME_SET_DATI_MQ	44	X'0000002C'
MQ_DB2_NAME_LENGTH	4	X'00000004'
LUNGHEZZA_NOME_DSG	8	X'00000008'
LUNGHEZZA_NOME_ENTITÀ_MQ.	1024	X'00000400'
LUNGHEZZA_INFO_INVIO_MQ	96	X'00000060'
LUNGHEZZA_INDIRIZZO_IP_MQ_	48	X'00000030'
LUNGHEZZA_ID_CORRELATO_LOG_MQ_	8	X'00000008'
MQ_LOG_EXTENT_NAME_LENGTH	24	X'00000018'
LUNGHEZZA_PERCORSO_LOG_MQ	1024	X'00000400'
LENGTH MQ_LRSN_	12	X'0000000C'
LUNGHEZZA_NOME_ORIGINE_MQ	8	X'00000008'
LUNGHEZZA_NOME_PSB_MQ	8	X'00000008'
ID_PST_MQ_LENGTH	8	X'00000008'
MQ_Q_MGR_CPF_LENGTH	4	X'00000004'
ID_MQ_RESPONSE_LENGTH	24	X'00000018'
MQ_RBA_LENGTH	16	X'00000010'
LUNGHEZZA_PROFILO_SICUREZZA_MQ	40	X'00000028'
LUNGHEZZA_COMPONENTE_SERVIZIO_MQ	48	X'00000030'

Tabella 8. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
LUNGHEZZA_NOME_SECONDARIO_MQ_	10240	X'00002800'
LUNGHEZZA_SERVIZIO_SYSP_MQ	32	X'00000020'
LUNGHEZZA_NOME_SISTEMA_MQ	8	X'00000008'
LUNGHEZZA_NUMERO_ATTIVITÀ_MQ	8	X'00000008'
MQ_TPIPE_PFX_LENGTH	4	X'00000004'
LUNGHEZZA_ID_UOW	256	X'00000100'
LUNGHEZZA_DATI_UTENTE_MQ	10240	X'00002800'
LENGTH MQ_VOLSER_	6	X'00000006'

MQACH_ * (struttura intestazione area concatenamento uscita API)

Tabella 9. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQACH_	"ACH¬"
MATRICE MQACH_STRUC_ID_ARRAY	'A','C','H','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 10. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQACH_VERSION_1	1	X'00000001'
VERSIONE MQACH_CURRENT_	1	X'00000001'
MQACH_LENGTH_1	(value differs by platform or version)	(value differs by platform or version)
LENGTH - MQACH_CURRENT_	(value differs by platform or version)	(value differs by platform or version)

MQACT_ * (Token di account)

Tabella 11. Nomi e valori costanti	
Nome	**Valore**
MQACT_NONE	X'00...00' (32 valori null)
MQAC_NON_ARRAY	'\0','\0',... (32 valori null)

MQACT_ * (Opzioni azione formato comando)

Tabella 12. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQACT_FORCE_REMOVE	1	X'00000001'
DADVANCE_MQACT_LOG	2	X'00000002'
STATISTICHE COLLECT_MQACT_	3	X'00000003'
MQACT_SUB	4	X'00000004'

MQACTP_ * (Azione)

Tabella 13. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
NUOVO	0	X'00000000'
FORWARD MQACTP	1	X'00000001'
MQACTP_REPLY	2	X'00000002'
PROSPETTO MQACTP_REPORT	3	X'00000003'

MQACTT_ * (tipi di token di account)

Tabella 14. Valori delle costanti

Nome	Valore esadecimale
MQACTT_SCONOSCIUTO	X'00'
ID_CICS_LUOW MQACTT_	X'01'
MQACTT_OS2_DEFAULT	X'04'
MQACTT_DOS_DEFAULT	X'05'
ID_NUMERIC_UNIX_MQACTT_	X'06'
MQACTT_OS400_ACCOUNT_TOKEN	X'08'
MQACTT_WINDOWS_DEFAULT	X'09'
ID_MQACTT_NT_SECURITY_	X'0B'
UTENTE MQACT	X'19'

MQADOPT_ * (adotta nuovi controlli MCA e adotta nuovi tipi MCA)

Adotta nuovi controlli MCA

Tabella 15. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQADOPT_CHECK_NONE	0	X'00000000'
CHECK_MQADOPT_ALL	1	X'00000001'
MQADOPT_CHECK_Q_MGR_NAME	2	X'00000002'
MQADOPT_CHECK_NET_ADDR	4	X'00000004'

Adotta nuovi tipi MCA

Tabella 16. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQADOPT_TYPE_NO	0	X'00000000'
TIPO_MQADOPT_ALL	1	X'00000001'
SVR TYPE_MQADOPT_	2	X'00000002'
SDR TYPE_MQADOPT_	4	X'00000004'
MQADOPT_TYPE_RCVR	8	X'00000008'
MQADOPT_TYPE_CLUSRCVR	16	X'00000010'

MQAIR_ * (Struttura record informazioni di autenticazione)

Tabella 17. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQAIR	"AIR¬"
ARRAY MQAIR_STRUC_ID_ARRAY	'A','I','R','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 18. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQAIR_VERSION_1	1	X'00000001'
MQAIR_VERSION_2	2	X'00000002'
VERSIONE MQAIR_CURRENT_	2	X'00000002'

MQAIT_ * (Tipo di informazioni di autenticazione)

Tabella 19. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQAIT_ALL	0	X'00000000'
LDAP CRL MQAIT_	1	X'00000001'
OOCSP MQAIT	2	X'00000002'
Sistema operativo MQAIT_IDPW_OS	3	X'00000003'
IDPW_MQAIT_LDAP	4	X'00000004'

MQAS_ * (Formato del comando Valori di stato asincrono)

Tabella 20. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQAS_NONE	0	X'00000000'
MQAS_STARTED	1	X'00000001'
MQAS_START_WAIT	2	X'00000002'
MQAS_STOPPED	3	X'00000003'
MQAS_SOSPESO	4	X'00000004'
MQAS_SUSPENDED_TEMPORARY	5	X'00000005'
MQAS_ATTIVO	6	X'00000006'
MQAS_INACTIVE	7	X'00000007'

MQAT_ * (Tipi di applicazione Put)

Tabella 21. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQAT_SCONOSCIUTO	-1	X'FFFFFFFF'
MQAT_NO_CONTEXT	0	X'00000000'
MQAT_CICS	1	X'00000001'
MVS MQAT	2	X'00000002'
MQAT_OS390	2	X'00000002'

Tabella 21. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
ZOS MQAT	2	X'00000002'
IMS MQAT	3	X'00000003'
MQAT_OS2	4	X'00000004'
DOS MQAT	5	X'00000005'
AIX MQAT	6	X'00000006'
UNIX MQAT	6	X'00000006'
Gestore code MQAT	7	X'00000007'
MQAT_OS400	8	X'00000008'
WINDOWS MQAT	9	X'00000009'
CICS_VSE MQAT	10	X'0000000A'
MQAT_WINDOWS_NT	11	X'0000000B'
VMS MQAT	12	X'0000000C'
TUTORE_QAT_MQ	13	X'0000000D'
NSK MQAT	13	X'0000000D'
VOS MQAT	14	X'0000000E'
MQAT_OPEN_TP1	15	X'0000000F'
VM MQAT	18	X'00000012'
MQAT_IM_Bridge	19	X'00000013'
XCF MQAT	20	X'00000014'
BRIDGE - MQAT_CICS_BRIDGE	21	X'00000015'
MQAT_NOTES_AGENT	22	X'00000016'
TPF MQAT	23	X'00000017'
UTENTE MQAT	25	X'00000019'
MEDIA_MQAT_BROKER	26	X'0000001A'
MQAT_QMGR_PUBLISH	26	X'0000001A'
JAVA MQAT	28	X'0000001C'
DQM MQAT	29	X'0000001D'
Iniziatore MQAT_CHANNEL_INITIATOR	30	X'0000001E'
WLM MQAT	31	X'0000001F'
MQAT_BATCH	32	X'00000020'
BATCH_RRS_MQAT	33	X'00000021'
SIB MQAT	34	X'00000022'
MQAT_PREDEFINITO	(value differs by platform or version)	(value differs by platform or version)
MQAT_USER_FIRST	65536	X'00010000'
MQAT_USER_LAST	999999999	X'3B9AC9FF'

MQAUTH_ * (Valori autorità formato comando)

Tabella 22. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
NONE MQAUTH	0	X'00000000'
MQAUTH_ALT_USER_AUTHORITY	1	X'00000001'
BROWSE MQAUTH	2	X'00000002'
MODIFICA_AUTORI_QUERY	3	X'00000003'
CLEAR_MQAUTH	4	X'00000004'
MQAUTO_CONNECT	5	X'00000005'
CREA_MQAUTH	6	X'00000006'
MQAUTO_DELETE	7	X'00000007'
MQAUTO_DISPLAY	8	X'00000008'
INPUT MQAUTH	9	X'00000009'
INQUIRE MQAUTH	10	X'0000000A'
OUTPUT MQAUTH	11	X'0000000B'
MQAUTH_PASS_ALL_CONTEXT	12	X'0000000C'
MQAUTH_PASS_IDENTITY_CONTEXT	13	X'0000000D'
MQAUTO_SET	14	X'0000000E'
MQAUTH_SET_ALL_CONTEXT	15	X'0000000F'
MQAUTH_SET_IDENTITY_CONTEXT	16	X'00000010'
CONTROL MQAUTH	17	X'00000011'
MQAUTH_CONTROL_EXTENDED	18	X'00000012'
MQAUTH_PUBBLICA	19	X'00000013'
MQAUT_SUBSCRIBE	20	X'00000014'
RESUME MQAUTH	21	X'00000015'
SISTEMA MQAUTH	22	X'00000016'

MQAUTHOPT_ * (Opzioni autorità formato comando)

Tabella 23. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQAUTHOPT_CUMULATIVO	256	X'00000100'
MQAUTHOPT_ENTITY_EXPLICIT	1	X'00000001'
MQAUTOPT_ENTITA_SET	2	X'00000002'
MQAUTHOPT_NAME_ALL_MATCHING	32	X'00000020'
MQAUTHOPT_NAME_AS_WILDCARD	64	X'00000040'
MQAUTHOPT_NAME_EXPLICIT	16	X'00000010'

MQAXC_ * (struttura contesto uscita API)

Tabella 24. Strutture di costanti

Nome	Struttura
ID_STRUC_MQAXC	"AXC¬"

Tabella 24. Strutture di costanti (Continua)	
Nome	**Struttura**
MATRICE MQAXC_STRUC_ID_ARRAY	'A','X','C','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 25. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQAXC_VERSION_1	1	X'00000001'
VERSIONE MQAXC_CURRENT_	1	X'00000001'

MQAXP_ * (struttura parametro uscita API)

Tabella 26. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQAXP	"AXP¬"
MATRA_ID_STRUTTURA_MQAXP	'A','X','P','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 27. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQAXP_VERSION_1	1	X'00000001'
MQAXP_VERSION_2	2	X'00000002'
VERSIONE MQAXP_CURRENT_	2	X'00000002'

MQBA_ * (Selettori attributo byte)

Tabella 28. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQBA_FIRST	6001	X'00001771'
MQBA_LAST	8000	X'00001F40'

MQBACF_ * (Tipi di parametro byte in formato comando)

Tabella 29. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQBACF_PRIMO	7001	X'00001B59'
MQBACF_EVENT_ACCOUNTING_TOKEN	7001	X'00001B59'
ID SICUREZZA MQBACF_EVENT_	7002	X'00001B5A'
IMPOSTA_RISPOSTA MQBACF	7003	X'00001B5B'
ID_RISPOSTA MQBACF	7004	X'00001B5C'
ID_UOW_ESTERNO_MQBACF_	7005	X'00001B5D'
ID_CONNESSIONE_MQBACF	7006	X'00001B5E'
ID_CONNESSIONE_GENERICA_MQBACF	7007	X'00001B5F'
ID_UO_ORIGINE_MQBACF_	7008	X'00001B60'
ID_UOO_MGR_MQBACF_Q	7009	X'00001B61'

Tabella 29. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQBACF_ACCOUNTING_TOKEN	7010	X'00001B62'
ID_CORREL_MQBACF	7011	X'00001B63'
ID_GROUP_MQBACF	7012	X'00001B64'
ID_MSG_MQBACF	7013	X'00001B65'
LEID_CF_MQBACF	7014	X'00001B66'
ID_CORRELATO_DESTINAZIONE_MQBACF	7015	X'00001B67'
ID_SOTTO_MQBAC	7016	X'00001B68'
MQBACF_LAST_UTENTE	7016	X'00001B68'

MQBL_ * (Lunghezza buffer per la stringa mqAdde la stringa mqSet)

Tabella 30. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQBL_NULL_TERMINATO	-1	X'FFFFFFFF'

MQBMHO_ * (Buffer per gestire le opzioni e la struttura del messaggio)

Buffer per la struttura delle opzioni di gestione del messaggio

Tabella 31. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQBMHO_	"BMHO"
ID_STRUC_MQBMHO_ARRAY	'B','M','H','O'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 32. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQBMHO_VERSION_1	1	X'00000001'
VERSIONE MQBMHO_CURRENT_	1	X'00000001'

Buffer per opzioni di gestione messaggi

Tabella 33. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQBMHO_NONE	0	X'00000000'
MQBMHO_DELETE_PROPERTIES	1	X'00000001'

MQBND_ * (Bind predefiniti)

Tabella 34. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQBND_BIND_ON_OPEN	0	X'00000000'
MQBND_BIND_NO_FIXED	1	X'00000001'
MQBND_BIND_ON_XX_ENCODE_CASE_ONE gruppo	2	X'00000002'

MQBO_ * (Inizio opzioni e struttura)

Struttura delle opzioni di inizio

Tabella 35. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQBO	"BO¬¬"
ARRAY MQBO_STRUC_ID_ARRAY	'B','O','¬','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 36. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQBO_VERSION_1	1	X'00000001'
VERSIONE MQBO_CURRENT_	1	X'00000001'

Opzioni di inizio

Tabella 37. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQBO_NONE	0	X'00000000'

MQBT_ * (Tipi di bridge in formato comando)

Tabella 38. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
OTMA MQBT	1	X'00000001'

MQCA_ * (Selettori attributi carattere)

Tabella 39. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCA_ADMIN_TOPIC_NAME	2105	X'00000839'
MQCA_ALTERATION_DATE	2027	X'000007EB'
MQCA_ALTERATION_TIME	2028	X'000007EC'
MQCA_APPL_ID	2001	X'000007D1'
MQCA_AUTH_INFO_CONN_NAME	2053	X'00000805'
MQCA_AUTH_INFO_DESC	2046	X'000007FE'
MQCA_AUTH_INFO_NAME	2045	X'000007FD'
MQCA_AUTH_INFO_OCSP_URL	2109	X'0000083D'
MQCA_AUTO_REORG_CATALOG	2091	X'0000082B'
MQCA_AUTO_REORG_START_TIME	2090	X'0000082A'
MQCA_BACKOUT_REQ_Q_NAME	2019	X'000007E3'
MQCA_BASE_OBJECT_NAME	2002	X'000007D2'
MQCA_BASE_Q_NAME	2002	X'000007D2'
MQCA_BATCH_INTERFACE_ID	2068	X'00000814'
MQCA_CF_STRUC_DESC	2052	X'00000804'

Tabella 39. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQCA_CF_STRUC_NAME	2039	X'000007F7'
MQCA_CHANNEL_AUTO_DEF_EXIT	2026	X'000007EA'
MQCA_CHILD	2101	X'00000835'
MQCA_CHINIT_SERVICE_PARM	2076	X'0000081C'
MQCA_CICS_FILE_NAME	2060	X'0000080C'
MQCA_CLUS_CHL_NAME	2124	X'0000084C'
MQCA_CLUSTER_DATE	2037	X'000007F5'
MQCA_CLUSTER_NAME	2029	X'000007ED'
MQCA_CLUSTER_NAMELIST	2030	X'000007EE'
MQCA_CLUSTER_Q_MGR_NAME	2031	X'000007EF'
MQCA_CLUSTER_TIME	2038	X'000007F6'
MQCA_CLUSTER_WORKLOAD_DATA	2034	X'000007F2'
MQCA_CLUSTER_WORKLOAD_EXIT	2033	X'000007F1'
MQCA_COMMAND_INPUT_Q_NAME	2003	X'000007D3'
MQCA_COMMAND_REPLY_Q_NAME	2067	X'00000813'
MQCA_CREATION_DATE	2004	X'000007D4'
MQCA_CREATION_TIME	2005	X'000007D5'
MQCA_DEAD_LETTER_Q_NAME	2006	X'000007D6'
MQCA_DEF_XMIT_Q_NAME	2025	X'000007E9'
MQCA_DNS_GROUP	2071	X'00000817'
MQCA_ENV_DATA	2007	X'000007D7'
MQCA_FIRST	2001	X'000007D1'
MQCA_IGQ_USER_ID	2041	X'000007F9'
MQCA_INITIATION_Q_NAME	2008	X'000007D8'
MQCA_LAST	4000	X'00000FA0'
MQCA_LAST_USED	2109	X'0000083D'
MQCA_LDAP_PASSWORD	2048	X'00000800'
MQCA_LDAP_USER_NAME	2047	X'000007FF'
MQCA_LU_GROUP_NAME	2072	X'00000818'
MQCA_LU_NAME	2073	X'00000819'
MQCA_LU62_ARM_SUFFIX	2074	X'0000081A'
MQCA_MODEL_DURABLE_Q	2096	X'00000830'
MQCA_MODEL_NON_DURABLE_Q	2097	X'00000831'
MQCA_MONITOR_Q_NAME	2066	X'00000812'
MQCA_NAMELIST_DESC	2009	X'000007D9'
MQCA_NAMELIST_NAME	2010	X'000007DA'
MQCA_NAMES	2020	X'000007E4'
MQCA_PARENT	2102	X'00000836'
MQCA_PASS_TICKET_APPL	2086	X'00000826'

Tabella 39. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQCA_PROCESS_DESC	2011	X'000007DB'
MQCA_PROCESS_NAME	2012	X'000007DC'
MQCA_Q_DESC	2013	X'000007DD'
MQCA_Q_MGR_DESC	2014	X'000007DE'
MQCA_Q_MGR_IDENTIFIER	2032	X'000007F0'
MQCA_Q_MGR_NAME	2015	X'000007DF'
MQCA_Q_NAME	2016	X'000007E0'
MQCA_QSG_NAME	2040	X'000007F8'
MQCA_REMOTE_Q_MGR_NAME	2017	X'000007E1'
MQCA_REMOTE_Q_NAME	2018	X'000007E2'
MQCA_REPOSITORY_NAME	2035	X'000007F3'
MQCA_REPOSITORY_NAMELIST	2036	X'000007F4'
MQCA_RESUME_DATE	2098	X'00000832'
MQCA_RESUME_TIME	2099	X'00000833'
MQCA_SERVICE_DESC	2078	X'0000081E'
MQCA_SERVICE_NAME	2077	X'0000081D'
MQCA_SERVICE_START_ARGS	2080	X'00000820'
MQCA_SERVICE_START_COMMAND	2079	X'0000081F'
MQCA_SERVICE_STOP_ARGS	2082	X'00000822'
MQCA_SERVICE_STOP_COMMAND	2081	X'00000821'
MQCA_STDERR_DESTINATION	2084	X'00000824'
MQCA_STDOUT_DESTINATION	2083	X'00000823'
MQCA_SSL_CRL_NAMELIST	2050	X'00000802'
MQCA_SSL_CRYPTO_HARDWARE	2051	X'00000803'
MQCA_SSL_KEY_LIBRARY	2069	X'00000815'
MQCA_SSL_KEY_MEMBER	2070	X'00000816'
MQCA_SSL_KEY_REPOSITORY	2049	X'00000801'
MQCA_STORAGE_CLASS	2022	X'000007E6'
MQCA_STORAGE_CLASS_DESC	2042	X'000007FA'
MQCA_SYSTEM_LOG_Q_NAME	2065	X'00000811'
MQCA_TCP_NAME	2075	X'0000081B'
MQCA_TOPIC_DESC	2093	X'0000082D'
MQCA_TOPIC_NAME	2092	X'0000082C'
MQCA_TOPIC_STRING_FILTER	2108	X'0000083C'
MQCA_TOPIC_STRING	2094	X'0000082E'
MQCA_TPIPE_NAME	2085	X'00000825'
MQCA_TRIGGER_CHANNEL_NAME	2064	X'00000810'
MQCA_TRIGGER_DATA	2023	X'000007E7'
MQCA_TRIGGER_PROGRAM_NAME	2062	X'0000080E'

Tabella 39. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQCA_TRIGGER_TERM_ID	2063	X'0000080F'
MQCA_TRIGGER_TRANS_ID	2061	X'0000080D'
MQCA_USER_DATA	2021	X'000007E5'
MQCA_USER_LIST	4000	X'00000FA0'
MQCA_VERSION	2120	X'00000848'
MQCA_XCF_GROUP_NAME	2043	X'000007FB'
MQCA_XCF_MEMBER_NAME	2044	X'000007FC'
MQCA_XMIT_Q_NAME	2024	X'000007E8'

MQCACF_ * (Tipo di parametro carattere formato comando)

Tabella 40. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCACF_PRIMO	3001	X'00000BB9'
MQCACF_FROM_Q_NAME	3001	X'00000BB9'
MQCACF_TO_Q_NAME	3002	X'00000BBA'
MQCACF_DA_NOME_PROCESSO	3003	X'00000BBB'
MQCACF_TO_PROCESS_NAME	3004	X'00000BBC'
MQCACF_FROM_NAMELIST_NAME	3005	X'00000BBD'
MQCACF_TO_NAMELIST_NAME	3006	X'00000BBE'
MQCACF_FROM_CHANNEL_NAME	3007	X'00000BBF'
MQCACF_TO_CHANNEL_NAME	3008	X'00000BC0'
MQCACF_FROM_AUTH_INFO_NAME	3009	X'00000BC1'
MQCACF_TO_AUTH_INFO_NAME	3010	X'00000BC2'
Q_NAMES MQCACF	3011	X'00000BC3'
NOMI_PROCESSO_MQCACF	3012	X'00000BC4'
NAMELIST_MQCACF_NAMES	3013	X'00000BC5'
MQCACF_TESTO_ESCAPE	3014	X'00000BC6'
MQCACF_LOCAL_Q_NAMES	3015	X'00000BC7'
NOMI_Q_MODELLO_MQCACF	3016	X'00000BC8'
MQCACF_ALIAS_Q_NAMES	3017	X'00000BC9'
MQCACF_REMOTE_Q_NAMES	3018	X'00000BCA'
MQCACF_NOMI_CANALE_MITTENTE	3019	X'00000BCB'
NOMI_CANALE_SERVER_MQCACF_	3020	X'00000BCC'
MQCACF_REQUESTER_CHANNEL_NAMES	3021	X'00000BCD'
NAMES MQCACF_RECEIVER_CHANNEL_NAMES	3022	X'00000BCE'
MQCACF_OBJECT_Q_MGR_NAME	3023	X'00000BCF'
MQCACF_APPL_NAME	3024	X'00000BD0'
IDENTIFICATIVO_UTENTE_MQCACF	3025	X'00000BD1'
MQCACF_AUX_ERROR_DATA_STR_1	3026	X'00000BD2'

Tabella 40. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQCACF_AUX_ERROR_DATA_STR_2	3027	X'00000BD3'
MQCACF_AUX_ERROR_DATA_STR_3	3028	X'00000BD4'
MQCACF_NOME_BRIDGE	3029	X'00000BD5'
MQCACF_NOME_FLUSSO	3030	X'00000BD6'
MQCACF_ARGOMENTO	3031	X'00000BD7'
MQCACF_PARENT_Q_MGR_NAME	3032	X'00000BD8'
ID CORREL_MQCACF	3033	X'00000BD9'
DATAORA_PUBBLICAZIONE_MQCACF	3034	X'00000BDA'
DATI_STRING_MQCACF	3035	X'00000BDB'
MQCACF_SUPPORTED_STREAM_NAME	3036	X'00000BDC'
MQCACF_REG_TOPIC	3037	X'00000BDD'
ORA_REG_MQCACF	3038	X'00000BDE'
ID UTENTE MQCACF_REG_	3039	X'00000BDF'
MQCACF_CHILD_Q_MGR_NAME	3040	X'00000BE0'
MQCACF_REG_STREAM_NAME	3041	X'00000BE1'
MQCACF_REG_Q_MGR_NAME	3042	X'00000BE2'
MQCACF_REG_Q_NAME	3043	X'00000BE3'
ID_CORREL_REG_MQCACF	3044	X'00000BE4'
ID UTENTE MQCACF_EVENT_	3045	X'00000BE5'
MQCACF_XX_ENCODE_CASE_ONE nome_oggetto	3046	X'00000BE6'
MQCACF_EVENT_Q_MGR	3047	X'00000BE7'
MQCACF_AUTH_INFO_NAMES	3048	X'00000BE8'
MQCACF_EVENT_APPL_IDENTITY	3049	X'00000BE9'
MQCACF_EVENT_APPL_NAME	3050	X'00000BEA'
MQCACF_EVENT_APPL_ORIGIN	3051	X'00000BEB'
MQCACF_NOME_SOTTOSCRIZIONE	3052	X'00000BEC'
MQCACF_REG_SUB_NAME	3053	X'00000BED'
IDENTITÀ_SOTTOSCRIZIONE_MQCACF_	3054	X'00000BEE'
MQCACF_REG_SUB_IDENTITY	3055	X'00000BEF'
MQCACF_SOTTOSCRIZIONE_DATI_UTENTE	3056	X'00000BF0'
DATI_REG_MQCACF_SUB_USER_DATA	3057	X'00000BF1'
MQCACF_APPL_TAG	3058	X'00000BF2'
MQCACF_DATA_SET_NAME	3059	X'00000BF3'
MQCACF_UOW_START_DATE	3060	X'00000BF4'
ORA_INIZIO_UOW_MQCACF_	3061	X'00000BF5'
MQCACF_UOW_LOG_START_DATE	3062	X'00000BF6'
MQCACF_UOW_LOG_START_TIME	3063	X'00000BF7'
MQCACF_UOW_LOG_EXTENT_NAME	3064	X'00000BF8'
MQCACF_princiPAL_ENTITY_NAMES	3065	X'00000BF9'

Tabella 40. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQCACF_GROUP_ENTITY_NAMES	3066	X'00000BFA'
NOME MQCACF_AUTH_PROFILE	3067	X'00000BFB'
MQCACF_XX_ENCODE_CASE_ONE nome_invio	3068	X'00000BFC'
MQCACF_COMPONENTE_SERVIZIO	3069	X'00000BFD'
MQCACF_RESPONSE_Q_MGR_NAME	3070	X'00000BFE'
MQCACF_CURRENT_LOG_EXTENT_NAME	3071	X'00000BFF'
MQCACF_RESTART_LOG_EXTENT_NAME	3072	X'00000C00'
MQCACF_MEDIA_LOG_EXTENT_NAME	3073	X'00000C01'
PERCORSO MQCACF_LOG_PATH	3074	X'00000C02'
MQCACF_COMMAND_MQSC	3075	X'00000C03'
MQCACF_Q_MGR_CPF	3076	X'00000C04'
MQCACF_USAGE_LOG_RBA	3078	X'00000C06'
LRSN LOG_USAGE_MQCACF_	3079	X'00000C07'
AMBITO comando MQCACF_	3080	X'00000C08'
ASID_MQCACF	3081	X'00000C09'
NOME MQCACF_PSB	3082	X'00000C0A'
ID_PST_MQCACF	3083	X'00000C0B'
MQCACF_TASK_NUMBER	3084	X'00000C0C'
ID_TRANSAZIONE MQCACF	3085	X'00000C0D'
ID_UOW_MGR_MQCACF_Q	3086	X'00000C0E'
MQCACF_ORIGIN_NAME	3088	X'00000C10'
MQCACF_ENV_INFO	3089	X'00000C11'
MQCACF_PROFILO_SICUREZZA	3090	X'00000C12'
DATA_CONFIGURAZIONE_MQCACF	3091	X'00000C13'
ORA_CONFIGURAZIONE_MQCACF	3092	X'00000C14'
MQCACF_FROM_CF_STRUC_NAME	3093	X'00000C15'
MQCACF_TO_CF_STRUC_NAME	3094	X'00000C16'
MQCACF_CF_STRUC_NAMES	3095	X'00000C17'
DATA_NON_RIUSCITA MQCACF	3096	X'00000C18'
TEMpo_NON RIUSCITO MQCACF	3097	X'00000C19'
DATA_BACKUP_MQCAC	3098	X'00000C1A'
ORA_BACKUP_MQCACF	3099	X'00000C1B'
MQCACF_NOME_SISTEMA	3100	X'00000C1C'
MQCACF_CF_STRUC_BACKUP_START	3101	X'00000C1D'
MQCACF_CF_STRUC_BACKUP_END	3102	X'00000C1E'
MQCACF_CF_STRUC_LOG_Q_MGRS	3103	X'00000C1F'
MQCACF_DA_CLASSE_ARCHIVIAZIONE	3104	X'00000C20'
MQCACF_TO_STORAGE_CLASS	3105	X'00000C21'
MQCACF_STORAGE_CLASS_NAMES	3106	X'00000C22'

Tabella 40. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
Nome_DSG_MQCACF	3108	X'00000C24'
MQCACF_DB2_NAME	3109	X'00000C25'
ID_USER_SYSP_MQCACF	3110	X'00000C26'
MQCACF_SYSP_XX_ENCODE_CASE_ONE gruppo_organizzato	3111	X'00000C27'
MQCACF_SYSP_OTMA_MEMBER	3112	X'00000C28'
MQCACF_SYSP_OTMA_DRU_EXIT	3113	X'00000C29'
MQCACF_SYSP_OTMA_TPIPE_PFX	3114	X'00000C2A'
MQCACF_SYSP_ARCHIVE_PFX1	3115	X'00000C2B'
MQCACF_SYSP_ARCHIVE_UNIT1	3116	X'00000C2C'
ID_CORREL_LOG_MQCACF_SYSP_	3117	X'00000C2D'
VOLSER MQCACF_SYSP_UNIT_	3118	X'00000C2E'
MQCACF_SYSP_Q_MGR_TIME	3119	X'00000C2F'
MQCACF_SYSP_Q_MGR_DATA	3120	X'00000C30'
MGR_RBA MQCACF_SYSP_Q_	3121	X'00000C31'
RBA LOG_SYSP_MQCACF	3122	X'00000C32'
SERVICE_SYSP_MQCACF	3123	X'00000C33'
MQCACF_FROM_LISTENER_NAME	3124	X'00000C34'
MQCACF_TO_LISTENER_NAME	3125	X'00000C35'
MQCACF_FROM_XX_ENCODE_CASE_ONE nome_servizio	3126	X'00000C36'
MQCACF_TO_SERVICE_NAME	3127	X'00000C37'
DATA_DATA_LAST_MQCACF	3128	X'00000C38'
TEMPO_UT_AST_MQCACF	3129	X'00000C39'
DATA GET MQCACF_LAST_	3130	X'00000C3A'
ORA GET MQCACF_LAST_	3131	X'00000C3B'
DATA OPERAZIONE MQCACF	3132	X'00000C3C'
ORA_OPERAZIONE MQCACF	3133	X'00000C3D'
DESC_ATTIVITA_MQCACF	3134	X'00000C3E'
DATI_IDENTITÀ_APPL_MQCACF_	3135	X'00000C3F'
MQCACF_APPL_ORIGIN_DATA	3136	X'00000C40'
DATA_PUT_MQCACF	3137	X'00000C41'
ORA_PUT_MQCACF	3138	X'00000C42'
MQCACF_REPLY_TO_Q	3139	X'00000C43'
MQCACF_REPLY_TO_Q_MGR	3140	X'00000C44'
MQCACF_RESOLVED_Q_NAME	3141	X'00000C45'
ID_STRUC_MQCACF	3142	X'00000C46'
NOME MQCACF_VALUE_	3143	X'00000C47'
DATA_INIZIO_SERVIZIO_MQCACF_	3144	X'00000C48'
ORA_INIZIO_SERVIZIO_MQCACF	3145	X'00000C49'
MQCACF_SYSP_OFFLINE_RBA	3146	X'00000C4A'

Tabella 40. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQCACF_SYSP_ARCHIVE_PFX2	3147	X'00000C4B'
MQCACF_SYSP_ARCHIVE_UNIT2	3148	X'00000C4C'
MQCACF_TO_TOPIC_NAME	3149	X'00000C4D'
MQCACF_DA_NOME_ARGOMENTO	3150	X'00000C4E'
MQCACF_TOPIC_NAMES	3151	X'00000C4F'
MQCACF_SUB_NAME	3152	X'00000C50'
MQCACF_DESTINATION_Q_MGR	3153	X'00000C51'
DESTINAZIONE_MQCACF	3154	X'00000C52'
ID UTENTE MQCACF_SUB_	3156	X'00000C54'
DATI_USER_MQCACF	3159	X'00000C57'
MQCACF_SUB_SELECTOR	3160	X'00000C58'
DATA_MQCACF_LAST_PUB_DATA	3161	X'00000C59'
ORA_MQCACF_LAST_ORA_PUBG	3162	X'00000C5A'
MQCACF_FROM_XX_ENCODE_CASE_ONE nome_secondario	3163	X'00000C5B'
MQCACF_TO_SUB_NAME	3164	X'00000C5C'
TEMPO MQCACF_LAST_MSG_	3167	X'00000C5F'
DATA_MSG_AST_MQCACF	3168	X'00000C60'
PUNTO_SOTTOSCRIZIONE_MQCACF_	3169	X'00000C61'
FILTER MQCACF	3170	X'00000C62'
MQCACF_NONE	3171	X'00000C63'
MQCACF_ADMIN_TOPIC_NAMES	3172	X'00000C64'
MQCACF_ROUTING_IMPRONTA digitale	3173	X'00000C65'
DESC APPL_MQCACF	3174	X'00000C66'
MQCACF_Q_MGR_START_DATE	3175	X'00000C67'
MQCACF_Q_MGR_START_TIME	3176	X'00000C68'
MQCACF_FROM_COMM_INFO_NAME	3177	X'00000C69'
MQCACF_TO_COMM_INFO_NAME	3178	X'00000C6A'
MQCACF_CF_OFFLOAD_SIZE1	3179	X'00000C6B'
MQCACF_CF_OFFLOAD_SIZE2	3180	X'00000C6C'
MQCACF_CF_OFFLOAD_SIZE3	3181	X'00000C6D'
MQCACF_CF_SMDS_GENERIC_NAME	3182	X'00000C6E'
SMDS CF MQCACF	3183	X'00000C6F'
DATA_MQCACF_RECOVERY_DATE	3184	X'00000C70'
MQCACF_RECOVERY_TIME	3185	X'00000C71'
MQCACF_CF_SMDSCONN	3186	X'00000C72'
MQCACF_CF_STRUC_NAME	3187	X'00000C73'
MQCACF_ALTERNATE_USERID	3188	X'00000C74'
MQCACF_CHAR_ATTRS	3189	X'00000C75'
MQCACF_DYNAMIC_Q_NAME	3190	X'00000C76'

Tabella 40. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQCACF_NOME_HOST	3191	X'00000C77'
MQCACF_MQCB_NAME	3192	X'00000C78'
MQCACF_STRINGA_OGGETTO	3193	X'00000C79'
MQCACF_RESOLVED_LOCAL_Q_MGR	3194	X'00000C7A'
MQCACF_RESOLVED_LOCAL_Q_NAME	3195	X'00000C7B'
MQCACF_RESOLVED_OBJECT_STRING	3196	X'00000C7C'
MQCACF_RESOLVED_Q_MGR	3197	X'00000C7D'
MQCACF_SELECTION_STRING	3198	X'00000C7E'
INFO_XA_MQCACF	3199	X'00000C7F'
MQCACF_APPL_FUNCTION	3200	X'00000C80'
MQCACF_XQH_REMOTE_Q_NAME	3201	X'00000C81'
MQCACF_XQH_REMOTE_Q_MGR	3202	X'00000C82'
MQCACF_XQH_PUT_TIME	3203	X'00000C83'
MQCACF_XQH_PUT_DATE	3204	X'00000C84'
MQCACF_EXCL_OPERATOR_MESSAGES	3205	X'00000C85'
MQCACF_CSP_USER_IDENTIFIER	3206	X'00000C86'
ID_CLI_MQCACF_AMQP_ID	3207	X'00000C87'
MQCACF_ARCHIVE_LOG_EXTENT_NAME	3208	X'00000C88'
MQCACF_APPL_IMMOVABLE_DATE	3209	X'00000C89'
MQCACF_APPL_IMMOVABLE_TIME	3210	X'00000C8A'
MQCACF_NHA_INSTANCE_NAME	3211	X'00000C8B'
MQCACF_LAST_UTENTE	3211	X'00000C8B'

MQCACH_ * (Formato del comando Tipi di parametro del canale di caratteri)

Tabella 41. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCACH_PRIMO	3501	X'00000DAD'
MQCACH_CHANNEL_NAME	3501	X'00000DAD'
DESC MQCACH_	3502	X'00000DAE'
NOME MQCACH_MODE	3503	X'00000DAF'
TP_MQCACH_NOME	3504	X'00000DB0'
MQCACH_XMIT_Q_NAME	3505	X'00000DB1'
MQCACH_CONNECTION_NAME	3506	X'00000DB2'
MQCACH_NOME	3507	X'00000DB3'
NOME MQCACH_SEC_EXIT_	3508	X'00000DB4'
MQCACH_MSG_EXIT_NAME	3509	X'00000DB5'
MQCACH_XX_ENCODE_CASE_ONE nome_uscita	3510	X'00000DB6'
MQCACH_RCV_NOME	3511	X'00000DB7'
MQCACH_CHANNEL_NAMES	3512	X'00000DB8'

Tabella 41. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
DATI_SEC_MQCACH_EXIT_USER_DATA	3513	X'00000DB9'
MQCACH_MSG_EXIT_USER_DATA	3514	X'00000DBA'
DATI USER_FINE MQCACH_SEND_	3515	X'00000DBB'
MQCACH_RCV_EXIT_USER_DATA	3516	X'00000DBC'
ID_UTENTE MQCACH_	3517	X'00000DBD'
PASSWORD_MQCACH	3518	X'00000DBE'
ADDRESS MQCACH_LOCAL_	3520	X'00000DC0'
NOME MQCACH_LOCAL_NAME	3521	X'00000DC1'
MQCACH_LAST_MSG_TIME	3524	X'00000DC4'
DATA_MSG_AST_MQCACH_	3525	X'00000DC5'
ID UTENTE MCA_MQCACH_	3527	X'00000DC7'
ORA MQCACH_CHANNEL_START_TIME	3528	X'00000DC8'
DATA DI INIZIO MODIFICA MQCACH_	3529	X'00000DC9'
MQCACH_MCA_XX_ENCODE_CASE_ONE nome_lavoro	3530	X'00000DCA'
LAST_MQCACH_LUWID	3531	X'00000DCB'
ID LUW corrente MQCACH_	3532	X'00000DCC'
MQCACH_FORMAT_NAME	3533	X'00000DCD'
MQCACH_MR_EXIT_NAME	3534	X'00000DCE'
DATI_MR_EXIT_UTENTE MQCACH_	3535	X'00000DCF'
MQCACH_SSL_CIPHER_SPEC	3544	X'00000DD8'
MQCACH_SSL_PEER_NAME	3545	X'00000DD9'
MQCACH_SSL_HANDSHAKE_STAGE	3546	X'00000DDA'
MQCACH_SSL_SHORT_PEER_NAME	3547	X'00000DDB'
MQCACH_REMOTE_APPL_TAG	3548	X'00000DDC'
ID_UTENTE MQCACH_SSL_CERT_	3549	X'00000DDD'
MQCACH_SSL_CERT_ISSUER_NAME	3550	X'00000DDE'
NOME MQCACH_LU_	3551	X'00000DDF'
ADDRESS IP_MQCACH	3552	X'00000DE0'
MQCACH_TCP_NAME	3553	X'00000DE1'
NOME MQCACH_LISTENER_	3554	X'00000DE2'
ELENCO TABELLE MQCACH_DESC	3555	X'00000DE3'
DATA_STARE_LISTENER MQCACH_	3556	X'00000DE4'
DATA/ORA_DI_AVVIO MQCACH_LISTENER_START_TIME	3557	X'00000DE5'
MQCACH_SSL_KEY_RESET_DATA	3558	X'00000DE6'
MQCACH_SSL_KEY_RESET_TIME	3559	X'00000DE7'
MQCACH_LAST_UTENTE	3559	X'00000DE7'

MQCADSD_ * (Descrittori ADS intestazione informazioni CICS)

Tabella 42. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCADSD_NONE	0	X'00000000'
MQCADSD_INVIA	1	X'00000001'
MQCADSD_RECV	16	X'00000010'
MQCADSD_MSGFORMATO	256	X'00000100'

MQCAFTY_ * (Valori di affinità connessione)

Tabella 43. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCAFTI_NONE	0	X'00000000'
MQCAFTY_PREFERRED	1	X'00000001'

MQCAMO_ * (Formato del comando Tipi di parametri di monitoraggio carattere)

Tabella 44. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCAMO_FIRST	2701	X'00000A8D'
DATA_CLOSE_MQCAMO	2701	X'00000A8D'
ORA_CLOSE_MQCAMO	2702	X'00000A8E'
DATA_COLLEGA_MQCAMO	2703	X'00000A8F'
ORA_COLLEGA_MQCAMO	2704	X'00000A90'
DATA_DISCO MQCAMO	2705	X'00000A91'
TEMPO_DIS_CAMPO_DI_TABELLA	2706	X'00000A92'
DATA_FINE MQCAMO	2707	X'00000A93'
ORA_END_MQCAMO	2708	X'00000A94'
DATA_OPEN_MQCAMO	2709	X'00000A95'
ORA_OPEN_MQCAMO	2710	X'00000A96'
DATA STAR MQCAMO	2711	X'00000A97'
ORA_START_MQCAMO	2712	X'00000A98'
MQCAMO_LAST_UTENTE	2712	X'00000A98'

MQCBC_ * (struttura delle costanti MQCBC)

Tabella 45. Strutture di costanti

Nome	Struttura
ID_STRUC_MQCBC	"CBC¬"
ID_STRUC_MQCBC_ARRAY	'C','B','C','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 46. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCBC_VERSION_1	1	X'00000001'
VERSIONE MQCBC_CURRENT_	1	X'00000001'

MQCBCF_ * (Indicatori costanti MQCBC)

Tabella 47. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCBCF_NONE	0	X'00000000'
MQCBCF_READA_BUFFER_EMPTY	1	X'00000001'

MQCBCT_ * (tipo di callback costanti MQCBC)

Tabella 48. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCBCT_START_CALL	1	X'00000001'
MQCBCT_STOP_CALL	2	X'00000002'
MQCBC_REGISTER_CALL	3	X'00000003'
MQCBC_DEREGISTER_CALL	4	X'00000004'
MQCBCT_EVENT_CALL	5	X'00000005'
MQCBC_MSG_REMOVED	6	X'00000006'
MQCBCT_MSG_NOT_REMOVED	7	X'00000007'

MQCBD_ * (struttura costanti MQCBD)

Tabella 49. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQCBD	"CBD¬"
MQCBD_STRUC_ID_ARRAY	'C','B','D','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 50. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCBD_VERSION_1	1	X'00000001'
VERSIONE MQCBD_CURRENT_	1	X'00000001'

MQCBDO_ * (Opzioni di callback costanti MQCBD)

Tabella 51. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCBDO_NONE	0	X'00000000'
MQCBDO_START_CALL	1	X'00000001'
CALL MQCBDO_STOP_	4	X'00000004'
REGISTER_MQCBDO_CALL	256	X'00000100'
MQCBDO_DEREGISTER_CALL	512	X'00000200'

Tabella 51. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQCBDO_FAIL_IF_QUIESCING	8192	X'00002000'

MQCBO_ * (Opzioni di creazione - sacchetto per il sacchetto mqCreate)

Tabella 52. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCBO_NONE	0	X'00000000'
BAG MQCBO_USER_	0	X'00000000'
BAG MQCBO_ADMIN_	1	X'00000001'
BAG MQCBO_COMMAND_	16	X'00000010'
BAG MQCBO_SYSTEM_	32	X'00000020'
BAG MQCBO_GROUP_	64	X'00000040'
MQCBO_LIST_FORM_ALLOWED	2	X'00000002'
MQCBO_LIST_FORM_INIBITO	0	X'00000000'
MQCBO_REORDER_AS_REQUIRED	4	X'00000004'
MQCBO_DO_NO_REORDER	0	X'00000000'
MQCBO_CHECK_SELECTORS	8	X'00000008'
MQCBO_DO_NOT_CHECK_SELECTORS	0	X'00000000'

MQCBT_ * (costanti MQCBD Questo è il tipo di funzione di callback)

Tabella 53. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCBT_MESSAGE_CONSUMER	1	X'00000001'
HANDLER EVENTO MQCBT_	2	X'00000002'

MQCC_ * (codici di completamento)

Tabella 54. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCC_OK	0	X'00000000'
MQCC_AVVERTENZA	1	X'00000001'
MQCC_NON RIUSCITO	2	X'00000002'
MQCC_INATTIVO	-1	X'FFFFFFFF'

MQCCSI_ * (Coded Character Set Identifier)

Tabella 55. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCCSI_UNDEFINED	0	X'00000000'
MQCCSI_DEULT	0	X'00000000'
MQCCSI_Q_MGR	0	X'00000000'
MQCCSI_INHERIT	-2	X'FFFFFFFE'
MQCCSI_EMBEDDED	-1	X'FFFFFFFF'

Tabella 55. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
APPL MQCCSI	-3	X'FFFFFFFD'

MQCCT_ * (Opzioni attività di conversazione intestazione informazioni CICS)

Tabella 56. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
SÌ MQCC	1	X'00000001'
MQCCT_NO	0	X'00000000'

MQCD_ * (Struttura definizione canale)

Tabella 57. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCD_VERSION_1	1	X'00000001'
MQCD_VERSION_2	2	X'00000002'
MQCD_VERSION_3	3	X'00000003'
MQCD_VERSION_4	4	X'00000004'
MQCD_VERSION_5	5	X'00000005'
MQCD_VERSION_6	6	X'00000006'
MQCD_VERSION_7	7	X'00000007'
MQCD_VERSION_8	8	X'00000008'
MQCD_VERSION_9	9	X'00000009'
MQCD_VERSION_10	10	X'0000000A'
Multi MQCD_VERSION_11	11	X'0000000B'
Multi VERSIONE MQCD_CURRENT_	11	X'0000000B'
z/OS MQCD_VERSION_12	12	X'0000000C'
z/OS VERSIONE MQCD_CURRENT_	12	X'0000000C'
MQCD_LENGTH_4	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_5	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_6	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_7	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_8	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_9	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_10	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_11	(value differs by platform or version)	(value differs by platform or version)

Tabella 57. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
z/OS MQCD_LENGTH_12	(value differs by platform or version)	(value differs by platform or version)
LENGTH MQCD_XX_ENCODE_CASE_ONE corrente	(value differs by platform or version)	(value differs by platform or version)

MQCDC_ * (Conversione dati canale)

Tabella 58. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
CONVERSIONE MQCDC_SENDER_	1	X'00000001'
MQCDC_NO_SENDER_CONVERSIONE	0	X'00000000'

MQCERT_ * (tipo di criterio di convalida certificato)

MQ_CERT_VAL_POLICY_DEFAULT	0	X'00000000'
MQ_CERT_VAL_POLICY_ANY	0	X'00000000'
MQ_CERT_VAL_POLICY_RFC5280	1	X'00000001'
V 9.4.0 V 9.4.0 MQ_CERT_VAL_POLICY_NONE	2	X'00000002'

MQCF_ * (Indicatori funzionalità)

Tabella 59. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCF_NONE	0	X'00000000'
MQCF_DIST_LISTS	1	X'00000001'

MQCFAC_ * (CICS information header Facility)

Tabella 60. Nomi e valori costanti	
Nome	**Valore esadecimale**
MQCFAC_NONE	X'00...00' (8 valori null)
MQCFAC_NON_ARRAY	'\0','\0',... (8 valori null)

MQCFBF_ * (Struttura parametro filtro stringa di byte formato comando)

Tabella 61. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCFBF_STRUC_LENGTH_FIXED	20	X'00000014'

MQCFBS_ * (Struttura parametro stringa di byte formato comando)

Tabella 62. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCFBS_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFC_ * (opzioni di controllo dell'intestazione del formato del comando)

Tabella 63. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCF_LAST	1	X'00000001'
MQCF_NO_LAST	0	X'00000000'

MQCFGR_ * (Struttura parametro gruppo formato comando)

Tabella 64. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCFGR_STRUC_LENGTH	16	X'00000010'

MQCFH_ * (Struttura intestazione formato comando)

Tabella 65. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
LUNGHEZZA_STRUTTURA_MQCFH_STRUCT	36	X'00000024'
MQCFH_VERSION_1	1	X'00000001'
MQCFH_VERSION_2	2	X'00000002'
MQCFH_VERSION_3	3	X'00000003'
VERSIONE MQCFH_CURRENT_	3	X'00000003'

MQCFIF_ * (Struttura parametro filtro numero intero formato comando)

Tabella 66. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
LUNGHEZZA_STRUTTURA_MQCFIF_	20	X'00000014'

MQCFIL_ * (Formato del comando: struttura del parametro dell'elenco di numeri interi)

Tabella 67. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCFIL_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFIL64_* (Formato del comando Struttura di parametri dell'elenco di numeri interi a 64 bit)

Tabella 68. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCFIL64_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFIN_ * (Formato del comando: struttura del parametro intero)

Tabella 69. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
LUNGHEZZA_STRUTTURA_MQCFIN_	16	X'00000010'

MQCFIN64_* (Formato del comando: struttura del parametro intero a 64 bit)

Tabella 70. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCFIN64_STRUC_LENGTH	24	X'00000018'

MQCFO_ * (Formato del comando Aggiorna opzioni repository e formato del comando Rimuovi opzioni code)

Formato del comando Aggiorna opzioni repository

Tabella 71. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCFO_REFRESH_REPOSITORY_YES	1	X'00000001'
MQCFO_REFRESH_REPOSITORY_NO	0	X'00000000'

Formato del comando Opzioni di rimozione code

Tabella 72. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCFO_REMOVE_QUEUES_SÌ	1	X'00000001'
MQCFO_REMOVE_QUEUES_NO	0	X'00000000'

MQCFOP_ * (operatori filtro formato comando)

Tabella 73. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
LESS MQCFOP	1	X'00000001'
MQCFOP_EQUAL	2	X'00000002'
MQCFOP_VERDE	4	X'00000004'
MQCFOP_NO_LESS	6	X'00000006'
MQCFOP_NOT_EQUAL	5	X'00000005'
MQCFOP_NO_GREATER	3	X'00000003'
MQCFOP_LIKE	18	X'00000012'
MQCFOP_NOT_LIKE	21	X'00000015'
CONTAINS MQCFOP	10	X'0000000A'
MQCFOP_EXCLUDI	13	X'0000000D'
MQCFOP_CONTAINS_GEN	26	X'0000001A'
MQCFOP_EXCLUDES_GEN	29	X'0000001D'

MQCFR_ * (capacità di recupero CF)

Tabella 74. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
SÌ MQCFR	1	X'00000001'
MQCFR_NO	0	X'00000000'

MQCFSF_ * (Struttura parametro filtro stringa formato comando)

Tabella 75. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCFSF_STRUC_LENGTH_FIXED	24	X'00000018'

MQCFSL_ * (Struttura parametro elenco stringa formato comando)

Tabella 76. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCFSL_STRUC_LENGTH_FIXED	24	X'00000018'

MQCFST_ * (Struttura parametro stringa formato comando)

Tabella 77. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCFST_STRUC_LENGTH_FIXED	20	X'00000014'

MQCFSTATUS_ * (Stato CF formato comando)

Tabella 78. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCFSTATUS_NO_FOUND	0	X'00000000'
MQCFSTATO_ATTIVO	1	X'00000001'
MQCFSTATUS_IN_RECOVER	2	X'00000002'
MQCFSTATA_IN_BACKUP	3	X'00000003'
MQCFSTATUS_NON RIUSCITO	4	X'00000004'
MQCFSTATO_NONE	5	X'00000005'
MQCFSTATUS_UNKNOWN	6	X'00000006'
MQCFSTATUS_ADMIN_INCOMPLETE	20	X'00000014'
MQCFSTATUS_NEVER_UTENTE	21	X'00000015'
MQCFSTATO_NO_BACKUP	22	X'00000016'
MQCFSTATUS_NOT_FAILED	23	X'00000017'
MQCFSTATUS_NOT_RECOVERABLE	24	X'00000018'
ERRORE XES_MQCFSTATUS	25	X'00000019'

MQCFT_ * (Formato del comando Tipi di struttura)

Tabella 79. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCFT_NONE	0	X'00000000'
COMANDO MQCFT	1	X'00000001'
MQCF_XX_ENCODE_CASE_ONE risposta	2	X'00000002'
MQCFT_INTEGER	3	X'00000003'
MQCFT_STRING	4	X'00000004'
ELENCO_INTEGER_MQCFT	5	X'00000005'

Tabella 79. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
ELENCO STRINGA_MQCFT	6	X'00000006'
EVENTO MQCFT	7	X'00000007'
UTENTE MQCFT	8	X'00000008'
MQCFT_BYTE_STRING	9	X'00000009'
MQCF_TRACE_ROUTE	10	X'0000000A'
REPORT MQCFT	12	X'0000000C'
FILTER INTEGER_MQCFT	13	X'0000000D'
FILTRO MQCFT_STRING_FILTER	14	X'0000000E'
MQCFT_BYTE_STRING_FILTER	15	X'0000000F'
MQCF_COMMAND_XR	16	X'00000010'
MQCF_XR_MSG	17	X'00000011'
MQCFT_XR_ITEM	18	X'00000012'
SOMMARIO MQCFT_XR_	19	X'00000013'
Gruppo_MQCF	20	X'00000014'
STATISTICHE MQCFT	21	X'00000015'
CONSTATORI_MQCFT	22	X'00000016'
MQCFT_INTEGER64	23	X'00000017'
MQCFT_INTEGER64_LIST	25	X'00000019'

MQCFTYPE_ * (Tipi di CF in formato comando)

Tabella 80. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
APPL MQCFTYPE_	0	X'00000000'
MQCFTYPE_ADMIN	1	X'00000001'

MQCFUNC_ * (Funzioni intestazione informazioni CICS)

Tabella 81. Strutture di costanti	
Nome	**Struttura**
MQCFUN_MQCONN	"CONN"
MQCFUN_MQGET	"GET¬"
MQCFUN_MQINQ	"INQ¬"
MQCFUN_MQOPEN	"OPEN"
MQCFUN_MQPUT	"PUT¬"
MQCFUNC_MQPUT1	"PUT1"
MQCFUN_NONE	"¬¬¬¬"
MQCFUN_MQCONN_ARRAY	'C','O','N','N'
MQCFUN_MQGET_ARRAY	'G','E','T','¬'
MQCFUN_MQINQ_ARRAY	'I','N','Q','¬'
MQCFUN_MQOPEN_ARRAY	'O','P','E','N'

Tabella 81. Strutture di costanti (Continua)	
Nome	**Struttura**
MQCFUN_MQPUT_ARRAY	'P','U','T','¬'
MQCFUNC_MQPUT1_ARRAY	'P','U','T','1'
MQCFUNC_NONE_ARRAY	'¬','¬','¬','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

MQCGWI_ * (CICS information header Get Wait Interval)

Tabella 82. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCGWI_DEFAULT	-2	X'FFFFFFFE'

MQCHAD_ * (Definizione automatica canale)

Tabella 83. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
DISABLE_MQCHAD	0	X'00000000'
ENABLE_MQCHAD	1	X'00000001'

MQCHIDS_ * (Stato in dubbio formato comando)

Tabella 84. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCHIDS_NO_INDOUBT	0	X'00000000'
MQCHID_INDOUBT	1	X'00000001'

MQCHLD_ * (Formato del comando Disposizioni canale)

Tabella 85. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCHLD_ALL	-1	X'FFFFFFFF'
MQCHLD_PREDEFINITO	1	X'00000001'
MQCHLD_SHARED	2	X'00000002'
PRIVATE MQCHLD	4	X'00000004'
MQCHLD_FIXSHARED	5	X'00000005'

MQCHS_ * (Formato del comando Stato canale)

Tabella 86. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCHS_INACTIVE	0	X'00000000'
MQCHS_BINDING	1	X'00000001'
MQCHS_STARTING	2	X'00000002'
MQCHS_RUNNING	3	X'00000003'
MQCHS_STOPPING	4	X'00000004'

Tabella 86. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQCHS_RETRYING	5	X'00000005'
MQCHS_STOPPED	6	X'00000006'
MQCHS_REQUESTING	7	X'00000007'
MQCHS_PAUSED	8	X'00000008'
MQCHS_INITIALIZING	13	X'0000000D'
MQCHS_SWITCHING	14	X'0000000E'

MQCHSH_ * (Opzioni di riavvio condiviso del canale in formato comando)

Tabella 87. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCHSH_RESTART_NO	0	X'00000000'
MQCHSH_RESTART_SÌ	1	X'00000001'

MQCHSR_ * (Formato del comando Opzioni di arresto canale)

Tabella 88. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCHSR_STOP_NON_RICHIESTO	0	X'00000000'
MQCHSR_STOP_REQUESTED	1	X'00000001'

MQCHSSTATE_ * (Formato del comando Sottostati del canale)

Tabella 89. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCHSSTATE_ALTRO	0	X'00000000'
MQCHSSTATE_END_OF_BATCH	100	X'00000064'
MQCHSSTATE_SENDING	200	X'000000C8'
MQCHSSTATE_RICEZIONE	300	X'0000012C'
MQCHSSTATE_SERIALIZING	400	X'00000190'
MQCHSSTATE_RESYNCHING	500	X'000001F4'
MQCHSSTATE_HEARTBEAT	600	X'00000258'
MQCHSSTATE_IN_SCYEXIT	700	X'000002BC'
MQCHSSTATE_IN_RCVEXIT	800	X'00000320'
MQCHSSTATE_IN_SENDEXIT	900	X'00000384'
MQCHSSTATE_IN_MSGEXIT	1000	X'000003E8'
MQCHSSTATE_IN_MREXIT	1100	X'0000044C'
MQCHSSTATE_IN_CHADEXIT	1200	X'000004B0'
MQCHSSTATE_NET_CONN.	1250	X'000004E2'
MQCHSSTATE_SSL_HANDSHAKING	1300	X'00000514'
SERVER MQCHSSTATE_NAME_SERVER	1400	X'00000578'
MQCHSSTATE_IN_MQPUT	1500	X'000005DC'
MQCHSSTATE_IN_MQGET	1600	X'00000640'

Tabella 89. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQCHSSTATE_IN_MQI_CALL	1700	X'000006A4'
MQCHSSTATE_COMPRESSING	1800	X'00000708'

MQCHT_ * (Tipi di canale)

Tabella 90. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCH_SENDER	1	X'00000001'
SERVER MQCHT	2	X'00000002'
MQCH_DESTINATARIO	3	X'00000003'
RICHIESTA MQCHT_ER	4	X'00000004'
MQCHT_ALL	5	X'00000005'
CLNTCONN MQCHT	6	X'00000006'
SVRCONN MQCHT	7	X'00000007'
CLUSRCVR MQCHT	8	X'00000008'
MQCHT_CLUSSDR	9	X'00000009'

MQCHTAB_ * (Formato del comando Tipi di tabella del canale)

Tabella 91. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCHTAB_Q_MGR	1	X'00000001'
CNTCONN MQCHTAB_	2	X'00000002'

MQCI_ * (Identificativo correlazione)

Tabella 92. Nomi e valori costanti	
Nome	**Valore**
MQCI_NONE	X'00...00' (24 valori null)
MQCI_NONE_ARRAY	'\0','\0',... (24 valori null)
SESSIONE MQCI_NEW_	X'414D5121...'
MQCI_NUOVA_SESSIONE_ARRAY	'\x41','\x4D','\51','\x21',...

MQCIH_ * (Indicatori e struttura intestazione informazioni CICS)

Struttura dell'intestazione delle informazioni CICS

Tabella 93. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQCIH	"CIH¬"
MATRICE MQCIH_STRUC_ID_ARRAY	'C','I','H','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 94. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCIH_VERSION_1	1	X'00000001'
MQCIH_VERSION_2	2	X'00000002'
VERSIONE MQCIH_CURRENT_	2	X'00000002'
MQCIH_LENGTH_1	164	X'000000A4'
MQCIH_LENGTH_2	180	X'000000B4'
MQCIH_CURRENT_LENGTH	180	X'000000B4'

Indicatori intestazione informazioni CICS

Tabella 95. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCIH_NONE	0	X'00000000'
SCADENZA_PASS_MQCIH	1	X'00000001'
MQCIH_UNLIMITED_EXPIRATION	0	X'00000000'
MQCIH_REPLY_WITHOUT_NULLS	2	X'00000002'
MQCIH_REPLY_WITH_NULLS	0	X'00000000'
RETURN MQCIH_SYNC_ON	4	X'00000004'
MQCIH_NO_SYNC_ON_RETURN	0	X'00000000'

MQCLCT_ * (Tipi di cache cluster)

Tabella 96. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCLCT_STATICO	0	X'00000000'
MQCLCT_DYNAMIC	1	X'00000001'

MQCLRS_ * (Formato del comando Cancella ambito stringa argomento)

Tabella 97. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
LOCALE MQCLRS	1	X'00000001'
MQCLRS_GLOBAL	2	X'00000002'

MQCLRT_ * (Formato del comando Cancella tipo di stringa argomento)

Tabella 98. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCLR_CONSERVATO	1	X'00000001'

MQCLT_ * (Tipi di collegamento intestazione informazioni CICS)

Tabella 99. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
PROGRAMMA_MQCL	1	X'00000001'
TRANSAZIONE MQCLT	2	X'00000002'

MQCLWL_ * (Workload cluster)

Tabella 100. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCLWL_USEQ_LOCALE	0	X'00000000'
MQCLWL_USEQ_ANY	1	X'00000001'
MQCLWL_USEQ_AS_Q_MGR	-3	X'FFFFFFFD'

MQCLXQ_ * (Tipo di coda di trasmissione cluster)

MQCLXQ_* sono i valori che è possibile impostare nell'attributo del gestore code DEFCLXQ . L'attributo **DEFCLXQ** controlla quale coda di trasmissione è selezionata per default dai canali mittenti del cluster da cui ricevere i messaggi, per inviare i messaggi ai canali riceventi del cluster.

Tabella 101. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCLXQ_SCTQ	0	X'00000000'
MQCLXQ_CHANNEL	1	X'00000001'

Riferimenti correlati

“DefClusterXmitQueue(MQLONG)” a pagina 837
L'attributo DefClusterXmitQueue controlla la coda di trasmissione selezionata per impostazione predefinita dai canali mittenti del cluster da cui richiamare i messaggi, per inviare i messaggi ai canali riceventi del cluster.

Modifica gestore code

Interrogazione gestore code

Interroga gestore code (Risposta)

“MQINQ - Richiedi attributi oggetto” a pagina 723
La chiamata MQINQ restituisce un array di numeri interi e una serie di stringhe di carattere contenenti attributi di un oggetto.

MQCMD_ * (Codici comando)

Tabella 102. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCMD_NONE	0	X'00000000'
MQCMD_CHANGE_Q_MGR	1	X'00000001'
MQCMD_INQUIRE_Q_MGR	2	X'00000002'
MQCMD_XX_ENCODE_CASE_ONE modifica_processo	3	X'00000003'
MQCMD_COPY_PROCESS	4	X'00000004'
MQCMD_CREA_PROCESSO	5	X'00000005'
MQCMD_XX_ENCODE_CASE_ONE elimina_processo	6	X'00000006'
MQCMD_INQUIRE_PROCESSO	7	X'00000007'
MQCMD_CHANGE_Q	8	X'00000008'
MQCMD_CLEAR_Q	9	X'00000009'
MQCMD_COPY_Q	10	X'0000000A'
MQCMD_CREA_Q	11	X'0000000B'
MQCMD_DELETE_Q	12	X'0000000C'

Tabella 102. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQCMD_INQUIRE_Q	13	X'0000000D'
MQCMD_REFRESH_Q_MGR	16	X'00000010'
Q_STATS MQCMD_RESET_	17	X'00000011'
MQCMD_INQUIRE_Q_NAMES	18	X'00000012'
MQCMD_INQUIRE_PROCESS_NAMES	19	X'00000013'
MQCMD_INQUIRE_CHANNEL_NAMES	20	X'00000014'
MQCMD_CHANGE_CHALLEGATO	21	X'00000015'
MQCMD_COPY_CHALLEGATO	22	X'00000016'
MQCMD_CREA_XX_ENCODE_CASE_ONE canalizzata	23	X'00000017'
MQCMD_DELETE_CHALLEGATO	24	X'00000018'
MQCMD_INQUIRE_CHANNEL	25	X'00000019'
CANALE MQCMD_PING_	26	X'0000001A'
MQCMD_RESET_XX_ENCODE_CASE_ONE canalizzata	27	X'0000001B'
MQCMD_START_XX_ENCODE_CASE_ONE canalizzata	28	X'0000001C'
MQCMD_STOP_XX_ENCODE_CASE_ONE canalizzata	29	X'0000001D'
MQCMD_START_CHANNEL_INIT	30	X'0000001E'
MQCMD_START_CHANNEL_LISTENER	31	X'0000001F'
ELENCO NOMI COMANDO MQCMD_CHANGE_NAMELIST	32	X'00000020'
MQCMD_COPY_NAMELIST	33	X'00000021'
ELENCO NOMI MQCMD_CREATE_NAMELIST	34	X'00000022'
MQCMD_DELETE_NAMELIST	35	X'00000023'
MQCMD_INQUIRE_NAMELIST	36	X'00000024'
MQCMD_INQUIRE_NAMELIST_NAMES	37	X'00000025'
MQCMD_ESCAPE	38	X'00000026'
MQCMD_RESOLVE_CHALLEGATO	39	X'00000027'
MQCMD_PING_Q_MGR	40	X'00000028'
MQCMD_INQUIRE_Q_STATO	41	X'00000029'
MQCMD_INQUIRE_CHANNEL_STATUS	42	X'0000002A'
MQCMD_CONFIG_EVENT	43	X'0000002B'
MQCMD_Q_MGR_EVENT	44	X'0000002C'
MQCMD_PERFM_EVENT	45	X'0000002D'
EVENTO MQCMD_CHANNEL_EVENT	46	X'0000002E'
MQCMD_DELETE_PUBLICATION	60	X'0000003C'
MQCMD_DEREGISTER_PUBLISHER	61	X'0000003D'
MQCMD_DEREGISTER_SUBSCRIBER	62	X'0000003E'
MQCMD_PUBLISH	63	X'0000003F'
MQCMD_REGISTER_PUBLISHER	64	X'00000040'
MQCMD_REGISTER_SUBSCRIBER	65	X'00000041'
MQCMD_XX_ENCODE_CASE_ONE richiesta_di_aggiornamento	66	X'00000042'

Tabella 102. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQCMD_BROKER_INTERNO	67	X'00000043'
MQCMD_ATTIVITY_MSG	69	X'00000045'
MQCMD_INQUIRE_CLUSTER_Q_MGR	70	X'00000046'
MQCMD_RESUME_Q_MGR_CLUSTER	71	X'00000047'
MQCMD_SUSPEND_Q_MGR_CLUSTER	72	X'00000048'
MQCMD_REFRESH_CLUSTER	73	X'00000049'
MQCMD_RESET_CLUSTER	74	X'0000004A'
MQCMD_TRACE_ROUTE	75	X'0000004B'
MQCMD_REFRESH_SECURITY	78	X'0000004E'
MQCMD_CHANGE_AUTH_INFO	79	X'0000004F'
MQCMD_COPY_AUTH_INFO	80	X'00000050'
MQCMD_CREA_AUT_INFO	81	X'00000051'
MQCMD_DELETE_AUTH_INFO	82	X'00000052'
MQCMD_INQUIRE_AUTH_INFO	83	X'00000053'
MQCMD_INQUIRE_AUTH_INFO_NAMES	84	X'00000054'
MQCMD_INQUIRE_CONNECTION	85	X'00000055'
MQCMD_STOP_CONNECZIONE	86	X'00000056'
MQCMD_INQUIRE_AUTH_RECS	87	X'00000057'
MQCMD_INQUIRE_ENTITY_AUTH	88	X'00000058'
MQCMD_DELETE_AUTH_REC	89	X'00000059'
MQCMD_SET_AUTO_REC	90	X'0000005A'
MQCMD_LOGGER_EVENT	91	X'0000005B'
MQCMD_RESET_MGR	92	X'0000005C'
MQCMD_CHANGE_LISTENER	93	X'0000005D'
MQCMD_COPY_LISTENER	94	X'0000005E'
MQCMD_CREATE_LISTENER	95	X'0000005F'
MQCMD_DELETE_LISTENER	96	X'00000060'
MQCMD_INQUIRE_LISTENER	97	X'00000061'
STATO IN CODA MQCMD_INQUIRE_LISTENER_STATUS	98	X'00000062'
MQCMD_COMMAND_EVENT	99	X'00000063'
MQCMD_CHANGE_SECURITY	100	X'00000064'
MQCMD_CHANGE_CF_STRUC	101	X'00000065'
MQCMD_CHANGE_STG_CLASS	102	X'00000066'
MQCMD_CHANGE_TRACE	103	X'00000067'
MQCMD_ARCHIVE_LOG	104	X'00000068'
MQCMD_BACKUP_CF_STRUC	105	X'00000069'
MQCMD_CREATE_BUFFER_POOL	106	X'0000006A'
MQCMD_CREATE_PAGE_SET	107	X'0000006B'
MQCMD_CREATE_CF_STRUC	108	X'0000006C'

Tabella 102. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQCMD_CREATE_STG_CLASS	109	X'0000006D'
MQCMD_COPY_CF_STRUC	110	X'0000006E'
MQCMD_COPY_STG_CLASS	111	X'0000006F'
MQCMD_DELETE_CF_STRUC	112	X'00000070'
MQCMD_DELETE_STG_CLASS	113	X'00000071'
ARCHIVIO_RICHIESA_MQCMD	114	X'00000072'
MQCMD_INQUIRE_CF_STRUC	115	X'00000073'
MQCMD_INQUIRE_CF_STRUC_STATUS	116	X'00000074'
SERVER MQCMD_INQUIRE_CMD_SERVER	117	X'00000075'
MQCMD_INQUIRE_CHANNEL_INIT	118	X'00000076'
MQCMD_INQUIRE_QSG	119	X'00000077'
LOG INQUIMQ	120	X'00000078'
MQCMD_INQUIRE_SECURITY	121	X'00000079'
MQCMD_INQUIRE_STG_CLASS	122	X'0000007A'
SISTEMA MQCMD_INQUIRE	123	X'0000007B'
MQCMD_INQUIRE_THREAD	124	X'0000007C'
MQCMD_INQUIRE_TRACE	125	X'0000007D'
MQCMD_INQUIRE_USO	126	X'0000007E'
MQCMD_MOVE_Q	127	X'0000007F'
MQCMD_RECOVER_BSDS	128	X'00000080'
MQCMD_RECOVER_CF_STRUC	129	X'00000081'
MQCMD_RESET_TPIPE	130	X'00000082'
MQCMD_RESOLVE_INDOUBT	131	X'00000083'
MQCMD_RESUME_Q_MGR	132	X'00000084'
MQCMD_REVERIFY_SECURITY	133	X'00000085'
MQCMD_SET_ARCHIVE	134	X'00000086'
LOG SET_MQCM	136	X'00000088'
SISTEMA MQCMD_SET	137	X'00000089'
MQCMD_START_CMD_SERVER	138	X'0000008A'
MQCMD_START_Q_MGR	139	X'0000008B'
MQCMD_START_TRACE	140	X'0000008C'
MQCMD_STOP_CHANNEL_INIT	141	X'0000008D'
MQCMD_STOP_CHANNEL_LISTENER	142	X'0000008E'
MQCMD_STOP_CMD_SERVER	143	X'0000008F'
MQCMD_STOP_Q_MGR	144	X'00000090'
TRACCIA MQCMD_STOP	145	X'00000091'
MQCMD_SUSPEND_Q_MGR	146	X'00000092'
MQCMD_INQUIRE_CF_STRUC_NAMES	147	X'00000093'
MQCMD_INQUIRE_STG_CLASS_NAMES	148	X'00000094'

Tabella 102. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQCMD_CHANGE_SERVICE	149	X'00000095'
MQCMD_XX_ENCODE_CASE_ONE servizio	150	X'00000096'
MQCMD_CREA_SERVICE	151	X'00000097'
MQCMD_XX_ENCODE_CASE_ONE elimina_servizio	152	X'00000098'
MQCMD_INQUIRE_SERVICE	153	X'00000099'
MQCMD_INQUIRE_SERVICE_STATUS	154	X'0000009A'
MQCMD_START_SERVICE	155	X'0000009B'
SERVICE_STOP_MQCMD_	156	X'0000009C'
MQCMD_DELETE_BUFFER_POOL	157	X'0000009D'
MQCMD_DELETE_PAGE_SET	158	X'0000009E'
MQCMD_CHANGE_BUFFER_POOL	159	X'0000009F'
MQCMD_CHANGE_PAGE_SET	160	X'000000A0'
MQCMD_INQUIRE_Q_MGR_STATUS	161	X'000000A1'
MQCMD_CREA_LOG	162	X'000000A2'
MQCMD_STATISTICS_MQI	164	X'000000A4'
MQCMD_STATISTICS_Q	165	X'000000A5'
MQCMD_STATISTICS_CHALLEGATO	166	X'000000A6'
MQCMD_ACCOUNTING_MQI	167	X'000000A7'
MQCMD_XX_ENCODE_CASE_ONE conteggio_coda	168	X'000000A8'
MQCMD_INQUIRE_AUTH_SERVICE	169	X'000000A9'
MQCMD_CHANGE_TOPIC	170	X'000000AA'
MQCMD_COPY_TOPIC	171	X'000000AB'
MQCMD_CREA_TOPIC	172	X'000000AC'
MQCMD_DELETE_TOPIC	173	X'000000AD'
MQCMD_INQUIRE_TOPIC	174	X'000000AE'
MQCMD_INQUIRE_TOPIC_NAMES	175	X'000000AF'
MQCMD_INQUIRE_SUBSCRIPTION	176	X'000000B0'
MQCMD_CREATE_SUBSCRIPTION	177	X'000000B1'
MQCMD_CHANGE_SUBSCRIPTION	178	X'000000B2'
SOTTOSCRIZIONE_ELIMINAZIONE_MQCMD	179	X'000000B3'
MQCMD_COPY_SUBSCRIPTION	181	X'000000B5'
MQCMD_INQUIRE_SUB_STATUS	182	X'000000B6'
MQCMD_INQUIRE_TOPIC_STATO	183	X'000000B7'
MQCMD_CLEAR_TOPIC_STRING	184	X'000000B8'
MQCMD_INQUIRE_PUBSUB_STATUS	185	X'000000B9'
MQCMD_PURGE_XX_ENCODE_CASE_ONE canalizzata	195	X'000000C3'

MQCMDI_ * (Valori informazioni comando in formato comando)

Tabella 103. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCMDI_CMDSCOPE_ACCEPTED	1	X'00000001'
MQCMDI_CMDSCOPE_GENERATED	2	X'00000002'
MQCMDI_CMDSCOPE_COMPLETED	3	X'00000003'
MQCMDI_QSG_DISP_COMPLETED	4	X'00000004'
COMMAND_ACCEPTED MQCMDI	5	X'00000005'
MQCMDI_CLUSTER_RICHIESTA_ACCODATA	6	X'00000006'
MQCMDI_CHANNEL_INIT_STARTED	7	X'00000007'
MQCMDI_RECOVER_STARTED	11	X'0000000B'
MQCMDI_BACKUP_STARTED	12	X'0000000C'
MQCMDI_RECOVER_COMPLETED	13	X'0000000D'
MQCMDI_SEC_TIMER_ZERO	14	X'0000000E'
MQCMDI_REFRESH_CONFIGURATION	16	X'00000010'
ERRORE MQCMDI_SEC_SIGNOFF_	17	X'00000011'
MQCMDI_IMS_BRIDGE_SUSPENDED	18	X'00000012'
MQCMDI_DB2_SUSPENDED	19	X'00000013'
MQCMDI_DB2_OBSOLETE_MSGS	20	X'00000014'
MQCMDI_SEC_UPPERCASE	21	X'00000015'
MQCMDI_SEC_MIXEDCASE	22	X'00000016'

MQCMDL_ * (Livelli di comando)

Tabella 104. Nomi e valori costanti

Nome	Valore
MQCMDL_LEVEL_800	800
MQCMDL_LEVEL_801	801
MQCMDL_LEVEL_802	802
MQCMDL_LEVEL_900	900
MQCMDL_LEVEL_901	901
MQCMDL_LEVEL_902	902
MQCMDL_LEVEL_903	903
MQCMDL_LEVEL_904	904
MQCMDL_LEVEL_905	905
MQCMDL_LEVEL_910	910
MQCMDL_LEVEL_912	912
MQCMDL_LEVEL_913	913
MQCMDL_LEVEL_914	914
MQCMDL_LEVEL_915	915
MQCMDL_LEVEL_920	920
MQCMDL_LEVEL_921	921

Tabella 104. Nomi e valori costanti (Continua)	
Nome	**Valore**
MQCMDL_LEVEL_922	922
MQCMDL_LEVEL_923	923
MQCMDL_LEVEL_924	924
MQCMDL_LEVEL_925	925
MQCMDL_LEVEL_930	930
MQCMDL_LEVEL_931	931
MQCMDL_LEVEL_932	932

MQCMHO_ * (Creazione della struttura e delle opzioni dell'handle del messaggio)

Crea struttura di opzioni di gestione messaggi

Tabella 105. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQCMHO_	"CMHO"
MATRICE MQCMHO_STRUC_ID_ARRAY	'C','M','H','O'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 106. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCMHO_VERSION_1	1	X'00000001'
VERSIONE MQCMHO_CURRENT_	1	X'00000001'

Crea opzioni di gestione messaggi

Tabella 107. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCMHO_DEFAULT_VALIDATION	0	X'00000000'
MQCMHO_NO_VALIDATION	1	X'00000001'
VALIDATE MQCMHO_	2	X'00000002'
MQCMHO_NONE	0	X'00000000'

MQCNO_ * (Opzioni e struttura di connessione)

Struttura delle opzioni di collegamento

Tabella 108. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQCNO	"CNO¬"
ARRAY - MQCNO_STRUC_ID_ARRAY	'C','N','O','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 109. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCNO_VERSION_1	1	X'00000001'
MQCNO_VERSION_2	2	X'00000002'
MQCNO_VERSION_3	3	X'00000003'
MQCNO_VERSION_4	4	X'00000004'
MQCNO_VERSION_5	5	X'00000005'
VERSIONE MQCNO_CURRENT_	5	X'00000005'

Opzioni di connessione

Tabella 110. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCNO_STANDARD_BINDING	0	X'00000000'
MQCNO_FASTPATH_BINDING	1	X'00000001'
MQCNO_SERIALIZE_CONN_TAG_Q_MGR	2	X'00000002'
MQCNO_SERIALIZE_CONN_TAG_QSG	4	X'00000004'
MQCNO_RESTRICT_CONN_TAG_Q_MGR	8	X'00000008'
MQCNO_RESTRICT_CONN_TAG_QSG	16	X'00000010'
MQCNO_HANDLE_SHARE_NONE	32	X'00000020'
MQCNO_HANDLE_SHARE_BLOCK	64	X'00000040'
MQCNO_HANDLE_SHARE_NO_BLOCK	128	X'00000080'
MQCNO_SHARED_BINDING	256	X'00000100'
MQCNO_ISOLATED_BINDING	512	X'00000200'
BINDING MQCNO_LOCAL_	1024	X'00000400'
MQCNO_CLIENT_BINDING	2048	X'00000800'
MQCNO_ACCOUNTING_MQI_ENABLED	4096	X'00001000'
MQCNO_ACCOUNTING_MQI_DISABLED	8192	X'00002000'
MQCNO_ACCOUNTING_Q_ENABLED	16384	X'00004000'
MQCNO_ACCOUNTING_Q_DISABLED	32768	X'00008000'
MQCNO_NO_CONV_SHARING	65536	X'00010000'
MQCNO_ALL_CONVS_SHARE	262144	X'00040000'
MQCNO_CD_FOR_OUTPUT_ONLY	524288	X'00080000'
MQCNO_USE_CD_SELECTION	1048576	X'00100000'
MQCNO_RECONNECT	16777216	X'01000000'
MQCNO_RECONNECT_AS_DEF	0	X'00000000'
MQCNO_RECONNECT_DISABLED	33554432	X'02000000'
MQCNO_RECONNECT_Q_MGR	67108864	X'04000000'
MQCNO_ACTIVITY_TRACE_ENABLED	134217728	X'08000000'
MQCNO_ACTIVITY_TRACE_DISABLED	268435456	X'10000000'
MQCNO_NONE	0	X'00000000'

MQCO_* (Opzioni di chiusura)

Tabella 111. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCO_PRIMO	0	X'00000000'
MQCO_NONE	0	X'00000000'
MQCO_DELETE	1	X'00000001'
MQCO_DELETE_PURGE	2	X'00000002'
SUB MQCO_KEEP_	4	X'00000004'
MQCO_REMOVE_SUB	8	X'00000008'
MQCO_QUIESCE	32	X'00000020'

MQCODL_ * (Lunghezza dati di output intestazione informazioni CICS)

Tabella 112. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCODL_AS_INPUT	-1	X'FFFFFFFF'

MQCOMPRESS_ * (Compressione canale)

Tabella 113. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCOMPRESS_NON_DISPONIBILE	-1	X'FFFFFFFF'
MQCOMPRESS_NONE	0	X'00000000'
RLE MQCOMPRESS	1	X'00000001'
MQCOMPRESS_ZLIBFAST	2	X'00000002'
MQCOMPRESS_ZLIBHIGH	4	X'00000004'
SISTEMA MQCOMPRESS	8	X'00000008'
MQCOMPRESS_QUALSIASI	268435455	X'0FFFFFFF'

MQCONNID_ * (Identificatore connessione)

Tabella 114. Nomi e valori costanti

Nome	Valore
MQCONNID_NONE	X'00...00' (24 valori null)
MQCONNID_NON_ARRAY	'\0','\0',... (24 valori null)

MQCOPY_ * (Opzioni di copia proprietà)

Tabella 115. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQCOPY_NONE	0	X'00000000'
MQCOPY_ALL	1	X'00000001'
MQCOPY_FORWARD	2	X'00000002'
MQCOPY_PUBLISH	4	X'00000004'
MQCOPY_REPLY	8	X'00000008'

Tabella 115. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
PROSPETTO MQCOPY_REPORT	16	X'00000010'
MQCOPY_DEFAULT	22	X'00000016'

MQCQT_ * (Tipi di coda cluster)

Tabella 116. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
Coda MQCQT_LOCAL_Q	1	X'00000001'
MQCQ_ALIAS Q	2	X'00000002'
MQCQ_REMOTE_Q	3	X'00000003'
MQCQ_Q_MGR_ALIAS	4	X'00000004'

MQCRC_ * (Codici di ritorno intestazione informazioni CICS)

Tabella 117. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
OK MQCRC	0	X'00000000'
ERRORE MQCRC_CICS_EXEC	1	X'00000001'
ERRORE MQCR_MQ_API	2	X'00000002'
ERRORE MQCRC_BRIDGE_	3	X'00000003'
MQCRC_BRIDGE_ABEND	4	X'00000004'
FINE ANOMALA APPLICAZIONE MQCR	5	X'00000005'
ERRORE MQCR_SECURITY_ERROR	6	X'00000006'
PROGRAMMA_MQCR_NOT_AVAILABLE	7	X'00000007'
MQCRC_BRIDGE_TIMEOUT	8	X'00000008'
MQCRC_TRANSID_NOT_AVAILABLE	9	X'00000009'

MQCS_ * (Stato consumer costanti MQCBC)

Tabella 118. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCS_NONE	0	X'00000000'
MQCS_SUSPENDED_TEMPORARY	1	X'00000001'
MQCS_SUSPENDED_USER_ACTION	2	X'00000002'
MQCS_SUSPENDED	3	X'00000003'
MQCS_STOPPED	4	X'00000004'

MQCSC_ * (Codici di inizio intestazione informazioni CICS)

Tabella 119. Strutture di costanti	
Nome	**Struttura**
INIZIO_MQCSC	"S¬¬¬"
DATI STAR MQCSC	"SD¬¬"
MQCSC_TERMINPUT	"TD¬¬"

Tabella 119. Strutture di costanti (Continua)	
Nome	**Struttura**
MQCSC_NONE	"¬¬¬¬"
MQCSC_START_ARRAY	'S','¬','¬','¬'
MQCSC_STARTDATA_ARRAY	'S','D','¬','¬'
MATRA_TERMINAZIONE_MQCSC_FINALE	'T','D','¬','¬'
MQCSC_NON_ARRAY	'¬','¬','¬','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

MQCSP_ * (Struttura dei parametri di sicurezza della connessione e Tipi di autenticazione)

Struttura dei parametri di sicurezza della connessione

Tabella 120. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQCSP	"CSP¬"
MATRICE MQCSP_STRUC_ID_ARRAY	'C','S','P','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 121. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCSP_VERSION_1	1	X'00000001'
MQCSP_VERSION_2	2	X'00000002'
MQCSP_VERSION_3	3	X'00000003'
VERSIONE MQCSP_CURRENT_	3	X'00000003'

Tipi di autenticazione dei parametri di sicurezza della connessione

Tabella 122. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCSP_AUTH_NONE	0	X'00000000'
MQCSP_AUTH_USER_ID_AND_PWD	1	X'00000001'
ID_AUTORE MQCSP_TOKEN	2	X'00000002'

MQCSRV_ * (Opzioni server dei comandi)

Tabella 123. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCSRV_CONVERT_NO	0	X'00000000'
MQCSRV_CONVERT_SÌ	1	X'00000001'
MQCSRV_DLQ_NO	0	X'00000000'
MQCSRV_DLQ_SÌ	1	X'00000001'

MQCT_ * (Tag connessione gestore code)

Tabella 124. Nomi e valori costanti	
Nome	**Valore**
MQCT_NONE	X'00...00' (128 valori null)
MQCT_NONE_ARRAY	'\0','\0',... (128 valori null)

MQCTES_ * (Stato fine attività intestazione informazioni CICS)

Tabella 125. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
NOSYNC MQCTES	0	X'00000000'
COMMIT MQCTES	256	X'00000100'
BACKOUT MQCTES	4352	X'00001100'
ENDTASK MQCTES	65536	X'00010000'

MQCTLO_ * (Struttura delle opzioni MQCTL e Opzioni di controllo consumer)

Struttura delle opzioni MQCTL

Tabella 126. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQCTLO_	"CTLO"
MATRICE MQCTLO_STRUC_ID_ARRAY	'C','T','L','O'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 127. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCTLO_VERSION_1	1	X'00000001'
VERSIONE MQCTLO_CURRENT_	1	X'00000001'

Opzioni MQCTL Opzioni di controllo utente

Tabella 128. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCTLO_NONE	0	X'00000000'
MQCTLO_THREAD_AFFINITY	1	X'00000001'
MQCTLO_FAIL_IF_QUIESCING	8192	X'00002000'

MQCUOWC_ * (Controlli unità di lavoro intestazione informazioni CICS)

Tabella 129. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
SOLO MQCUOWC_	273	X'00000111'
MQCUOWC_CONTINUA	65536	X'00010000'
MQCUOWC_FIRST	17	X'00000011'
MQCUOWC_MEDIO	16	X'00000010'

Tabella 129. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
LAST MQCUOWC	272	X'00000110'
COMMIT MQCUOWC	256	X'00000100'
BACKOUT MQCUOWC	4352	X'00001100'

MQCXP_ * (Struttura parametro uscita canale)

Tabella 130. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQCXP	"CXP¬"
ID_STRUC_MQCXP_ARRAY	'C','X','P','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 131. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCXP_VERSION_1	1	X'00000001'
MQCXP_VERSION_2	2	X'00000002'
MQCXP_VERSION_3	3	X'00000003'
MQCXP_VERSION_4	4	X'00000004'
MQCXP_VERSION_5	5	X'00000005'
MQCXP_VERSION_6	6	X'00000006'
MQCXP_VERSION_7	7	X'00000007'
MQCXP_VERSION_8	8	X'00000008'
MQCXP_VERSION_9	9	X'00000009'
VERSIONE MQCXP_CURRENT_	9	X'00000009'

MQDC_ * (Classe di destinazione)

Tabella 132. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQDC_MANAGED	1	X'00000001'
MQDC_PROVIDED	2	X'00000002'

MQDCC_ * (Opzioni di conversione e maschere e fattori)

Opzioni di conversione

Tabella 133. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQDCC_DEFAULT_CONVERSION	1	X'00000001'
MQDCC_FILL_TARGET_BUFFER	2	X'00000002'
MQDCC_INT_DEFAULT_CONVERSION	4	X'00000004'
MQDCC_SOURCE_ENC_NATIVE	(value differs by platform or version)	(value differs by platform or version)

Tabella 133. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQDCC_SOURCE_EN_NORMAL	16	X'00000010'
MQDCC_SOURCE_ENC_REVERSED	32	X'00000020'
MQDCC_SOURCE_ENC_UNDEFINED	0	X'00000000'
MQDCC_TARGET_ENC_NATIVE	(value differs by platform or version)	(value differs by platform or version)
MQDCC_XX_ENCODE_CASE_ONE destinazione - NORMAL	256	X'00000100'
MQDCC_TARGET_ENC_REVERSED	512	X'00000200'
MQDCC_TARGET_ENC_UNDEFINED	0	X'00000000'
MQDCC_NONE	0	X'00000000'

Maschere e fattori di conversione

Tabella 134. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQDCC_SOURCE_ENC_MASK	240	X'000000F0'
MASK_EN_MQDCC_XX_ENCODE_CASE_ONE destinazione	3840	X'00000F00'
MQDCC_SOURCE_ENC_FACTOR	16	X'00000010'
MQDCC_XX_ENCODE_CASE_ONE file_di_destinazione	256	X'00000100'

MQDELO_ * (Opzioni di eliminazione di pubblicazione / sottoscrizione)

Tabella 135. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQDELO_NONE	0	X'00000000'
MQDELO_LOCALE	4	X'00000004'

MQDH_ * (Struttura intestazione di distribuzione)

Tabella 136. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQDH_	"DH¬¬"
ID_STRUC_MQDH_ARRAY	'D','H','¬','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 137. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQDH_VERSION_1	1	X'00000001'
VERSIONE MQDH_CURRENT_	1	X'00000001'

MQDHF_ * (Indicatore intestazione distribuzione)

Tabella 138. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
ID_MSG_NEW_MQDHF	1	X'00000001'

Tabella 138. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQDHF_NONE	0	X'00000000'

MQDISCONNECT_ * (Tipi di disconnessione formato comando)

Tabella 139. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQDISCONNECT_NORMALE	0	X'00000000'
MQDISCONNECT_IMPLICIT	1	X'00000001'
MQDISCONNECT_Q_MGR	2	X'00000002'

MQDL_ * (Elenchi di distribuzione)

Tabella 140. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQDL_SUPPORTED	1	X'00000001'
MQDL_NOT_SUPPORTED	0	X'00000000'

MQDLH_ * (Struttura intestazione non instradabile)

Tabella 141. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQDLH	"DLH¬"
ID_MQDLH_STRUC_ARRAY	'D','L','H','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

MQDLH_VERSION_1	1	X'00000001'
VERSIONE MQDLH_CURRENT_	1	X'00000001'

MQDLV_ * (Distribuzione messaggi persistenti/non persistenti)

Tabella 142. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQDLV_AS_PARENT	0	X'00000000'
MQDLV_ALL	1	X'00000001'
DUR_TUT_MQDLV	2	X'00000002'
MQDLV_ALL_AVAIL	3	X'00000003'

MQDMHO_ * (Elimina opzioni e struttura dell'handle del messaggio)

Struttura delle opzioni di gestione dei messaggi di eliminazione

Tabella 143. Strutture di costanti	
Nome	**Struttura**
ID MQDMHO_STRUC_ID	"DMHO"
ID_STRUC_MQDMHO_ARRAY	'D','M','H','O'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 144. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQDMHO_VERSION_1	1	X'00000001'
VERSIONE MQDMHO_CURRENT_	1	X'00000001'

Opzioni di eliminazione dell'handle del messaggio

Tabella 145. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQDMHO_NONE	0	X'00000000'

MQDMPO_ * (Elimina struttura e opzioni della proprietà del messaggio)

Elimina struttura delle opzioni delle proprietà del messaggio

Tabella 146. Strutture di costanti

Nome	Struttura
ID_STRUC_MQDMPO	"DMPO"
ID_STRUC_MQDMPO_ARRAY	'D','M','P','O'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 147. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQDMPO_VERSION_1	1	X'00000001'
VERSIONE MQDMPO_CURRENT_	1	X'00000001'

Opzioni di eliminazione proprietà messaggio

Tabella 148. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQDMPO_DEL_PRIMO	0	X'00000000'
MQDMPO_DEL_PROP_UNDER_CURSOR	1	X'00000001'
MQDMPO_NONE	0	X'00000000'

MQDNSWLM_ * (WLM DNS)

Tabella 149. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQDNSWLM_NO	0	X'00000000'
SÌ MQDNSWLM	1	X'00000001'

MQDT_ * (tipi di destinazione)

Tabella 150. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
APPL MQDT	1	X'00000001'

Tabella 150. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
BROKER_MQD	2	X'00000002'

MQDXP_ * (Struttura parametro di uscita conversione)

Tabella 151. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQDXP	"DXP¬"
MATRICE MQDXP_STRUC_ID_ARRAY	'D','X','P','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 152. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQDXP_VERSION_1	1	X'00000001'
MQDXP_VERSION_2	2	X'00000002'
VERSIONE MQDXP_CURRENT_	2	X'00000002'

MQEC_ * (Valori segnale)

Tabella 153. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQEC_MSG_ARRIVED	2	X'00000002'
MQEC_WAIT_INTERVAL_EXPIRED	3	X'00000003'
MQEC_WAIT_ANNULLATO	4	X'00000004'
MQEC_Q_MGR_QUIESCING	5	X'00000005'
MQEC_CONNECTION_QUIESCING	6	X'00000006'

MQEI_ * (Scadenza)

Tabella 154. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQEI_UNLIMITED	-1	X'FFFFFFFF'

MQENC_ * (Codifica)

MQENC_ * (Codifica)

Tabella 155. Valori delle costanti per piattaforma

Nome	Piattaforma	Valore decimale	Valore esadecimale
MQEN_NATIVE	IBM i	273	X'00000111'
	Linux	546	X'00000222'
	Linux su SPARC	273	X'00000111'
	Linux su x86	546	X'00000222'
	AIX and Linux	273	X'00000111'
	Windows	546	X'00000222'
	Micro Focus COBOL su Windows	17	X'00000011'
	z/OS	785	X'00000311'

MQENC_ * (Maschere di codifica)

Tabella 156. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MASCHERA_NUMERO_INTERO_MQENC_	15	X'0000000F'
MQEN_DECIM_MASK	240	X'000000F0'
MASCHERA_MQENC_MOBILE	3840	X'00000F00'
MASK MQENC_RESERVED_	-4096	X'FFFFF000'

MQENC_ * (Codifiche per numeri interi binari)

Tabella 157. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQEN_INTEGER_UNDEFINED	0	X'00000000'
MQENC_INTEGER_NORMAL	1	X'00000001'
MQENC_INTEGER_REVERSED	2	X'00000002'

MQENC_ * (Codifiche per numeri interi decimali compressi)

Tabella 158. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQEN_DECIM_UNDEFINED	0	X'00000000'
MQENC_DECIMAL_NORMAL	16	X'00000010'
MQENC_DECIMAL_REVERSED	32	X'00000020'

MQENC_ * (Codifiche per numeri a virgola mobile)

Tabella 159. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQEN_FLOAT_UNDEFINED	0	X'00000000'
MQENC_FLOAT_IEEE_NORMAL	256	X'00000100'

Tabella 159. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQENC_FLOAT_IEEE_REVERSED	512	X'00000200'
MQENC_FLOAT_S390	768	X'00000300'
MQEN_FLOAT_TNS	1024	X'00000400'

MQEPH_ * (Struttura intestazione del formato del comando integrato e indicatori)

Struttura dell'intestazione del formato del comando incorporato

Tabella 160. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQEPH	"EPH¬"
ARRAY - MQEPH_STRUC_ID_ARRAY	'E','P','H','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 161. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQEPH_STRUC_LENGTH_FIXED	68	X'00000044'
MQEPH_VERSION_1	1	X'00000001'
VERSIONE MQEPH_CURRENT_	1	X'00000001'

Indicatori dell'intestazione del formato del comando integrato

Tabella 162. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQEPH_NONE	0	X'00000000'
MQEPH_CCSID_EMBEDDED	1	X'00000001'

MQET_ * (Tipo di escape del formato del comando)

Tabella 163. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQET_MQSC	1	X'00000001'

MQEVO_ * (Formato del comando Origini evento)

Tabella 164. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQEVO_ALTRO	0	X'00000000'
CONSOLE MQEVO	1	X'00000001'
INIT_MQEVO	2	X'00000002'
MQEVO_MSG	3	X'00000003'
MQEVO_MQSET	4	X'00000004'
MQEVO_INTERNO	5	X'00000005'

Tabella 164. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQEVO_MQSUB	6	X'00000006'
MQEVO_CTLMSG	7	X'00000007'
REST MQEVO	8	X'00000008'

MQEVR_ * (Formato del comando Registrazione evento)

Tabella 165. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
DISABILITAZIONE_MQEVR	0	X'00000000'
MQEVR_ENABLED	1	X'00000001'
MQEVR_ECCEZIONE	2	X'00000002'
MQEVR_NO_DISPLAY	3	X'00000003'

MQEXPI_ * (Intervallo scansione scadenza)

Tabella 166. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQEXPI_OFF	0	X'00000000'

MQFB_ * (Valori feedback)

Tabella 167. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQFB_NONE	0	X'00000000'
MQFB_SYSTEM_FIRST	1	X'00000001'
MQFB_QUIT	256	X'00000100'
SCADENZA_MQFB	258	X'00000102'
COA MQFB	259	X'00000103'
COD MQFB	260	X'00000104'
MQFB_CHANNEL_COMPLETED	262	X'00000106'
MQFB_CHANNEL_FAIL_RETRY	263	X'00000107'
MQFB_CHANNEL_NON RIUSCITO	264	X'00000108'
MQFB_APPL_NON può essere avviato	265	X'00000109'
ERRORE MQFB_TM_	266	X'0000010A'
ERRORE TIPO_APPL_MQFB	267	X'0000010B'
MQFB_STOPPED_BY_MSG_EXIT	268	X'0000010C'
MQFB_ATTIVITÀ	269	X'0000010D'
MQFB_XMIT_Q_MSG_ERROR	271	X'0000010F'
PAN MQFB	275	X'00000113'
NAN_MQFB	276	X'00000114'
MQFB_STOPPED_BY_CHAD_EXIT	277	X'00000115'
MQFB_STOPPED_BY_PUBSUB_EXIT	279	X'00000117'
MQFB_NOT_A_REPOSITORY_MSG	280	X'00000118'

Tabella 167. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQFB_BIND_OPEN_CLUSRCVR_DEL	281	X'00000119'
MQFB_MAX_ATTIVITÀ	282	X'0000011A'
MQFB_NOT_FORWARDED	283	X'0000011B'
MQFB_NON_CONSEGNATO	284	X'0000011C'
MQFB_UNSUPPORT_FORWARDING	285	X'0000011D'
MQFB_UNSUPPORTED_DELIVERY	286	X'0000011E'
LENGTH_ZERO MQFB_DATA_	291	X'00000123'
MQFB_DATA_LENGTH_NEGATIVE	292	X'00000124'
LENGTH_TOO_BIG MQFB_DATA_BIG	293	X'00000125'
MQFB_BUFFER_OVERFLOW	294	X'00000126'
MQFB_LENGTH_OFF_BY_ONE	295	X'00000127'
ERRORE MQFB_IIH	296	X'00000128'
MQFB_NOT_AUTHORIZED_FOR_IMS	298	X'0000012A'
ERRORE MQFB_IMS	300	X'0000012C'
MQFB_IMS_FIRST	301	X'0000012D'
MQFB_IMS_LAST	399	X'0000018F'
ERRORE INTERNO MQFB_CICS_	401	X'00000191'
MQFB_CICS_NOT_AUTHORIZED	402	X'00000192'
MQFB_CICS_BRIDGE_FAILURE	403	X'00000193'
ERRORE MQFB_CICS_CORREL_ID_	404	X'00000194'
ERRORE CCSID MQFB_CICS_	405	X'00000195'
ERRORE MQFB_CICS_ENCODING_ERROR	406	X'00000196'
ERRORE MQFB_CICS_CIH	407	X'00000197'
ERRORE MQFB_CICS_UOW_	408	X'00000198'
ERRORE MQFB_CICS_COMMAREA_	409	X'00000199'
MQFB_CICS_APPL_NOT_STARTED	410	X'0000019A'
MQFB_CICS_APPL_ABENDED	411	X'0000019B'
ERRORE MQFB_CICS_DLQ	412	X'0000019C'
MQFB_CICS_UOW_BACKED_OUT	413	X'0000019D'
MQFB_PUBLICATIONS_ON_REQUEST	501	X'000001F5'
MQFB_SUBSCRIER_IS_PUBLISHER	502	X'000001F6'
MQFB_MSG_SCOPE_MISMATCH	503	X'000001F7'
MQFB_SELECTOR_MISMATCH	504	X'000001F8'
MQFB_IMS_NACK_1A_REASON_FIRST	600	X'00000258'
MQFB_IMS_NACK_1A_REASON_LAST	855	X'00000357'
MQFB_SYSTEM_LAST	65535	X'0000FFFF'
MQFB_APPL_FIRST	65536	X'00010000'
LAST APPL_MQFB	999999999	X'3B9AC9FF'

MQFC_ * (Opzioni di forzatura del formato del comando)

Tabella 168. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
SÌ MQFC	1	X'00000001'
N. MQFC	0	X'00000000'

MQFMT_ * (formati)

Tabella 169. Nomi e valori costanti	
Nome	**Valore**
MQFMT_NONE	"¬¬¬¬¬¬¬¬"
MMQFMT_ADMIN	"MQADMIN¬"
MQFMT_CHANNEL_COMPLETED	"MQCHCOM¬"
MQFMT_CICS	"MQCICS¬¬"
MQFMT_COMMAND_1	"MQCMD1¬¬"
MQFMT_COMMAND_2	"MQCMD2¬¬"
MQFMT_DEAD_LETTER_HEADER	"MQDEAD¬¬"
INTESTAZIONE_DIST_MQFM	"MQHDIST¬"
MQFMT_EMBEDDED_PCF	"MQHEPCF¬"
EVENTO MQFMT	"MQEVENT¬"
IMS MQFMT	"MQIMS¬¬¬"
MQFMT_IMS_VAR_STRING	"MQIMSVS¬"
MQFMT_MD_EXTENSIONE	"MQHMDE¬¬"
MQFMT_PCF	"MQPCF¬¬¬"
MQFMT_REF_MSG_HEADER	"MQHREF¬¬"
MQFMT_RF_HEADER	"MQHRF¬¬¬"
MQFMT_RF_HEADER_1	"MQHRF¬¬¬"
MQFMT_RF_HEADER_2	"MQHRF2¬¬"
MQFMT_STRING	"MQSTR¬¬¬"
MQFMT_TRIGGER	"MQTRIG¬¬"
Intestazione MQFMT_WORK_INFO_HEADER	"MQHWIH¬¬"
MQFMT_XMIT_Q_HEADER	"MQXMIT¬¬"
MQFMT_NONE_ARRAY	'¬','¬','¬','¬','¬','¬','¬','¬'
ARRAY MQFMT_ADMIN_	'M','Q','A','D','M','I','N','¬'
MQFMT_CHANNEL_COMPLETED_ARRAY	'M','Q','C','H','C','O','M','¬'
CICS_ARRAY MQFMT_	'M','Q','C','I','C','S','¬','¬'
MQFMT_COMMAND_1_ARRAY	'M','Q','C','M','D','1','¬','¬'
MQFMT_COMMAND_2_ARRAY	'M','Q','C','M','D','2','¬','¬'
MQFMT_DEAD_LETTER_HEADER_ARRAY	'M','Q','D','E','A','D','¬','¬'
MQFMT_DIST_HEADER_ARRAY	'M','Q','H','D','I','S','T','¬'
MQFMT_EMBEDDED_PCF_ARRAY	'M','Q','H','E','P','C','F','¬'
ARRA_EV_MQFMT	'M','Q','E','V','E','N','T','¬'

Tabella 169. Nomi e valori costanti (Continua)	
Nome	**Valore**
MQFMT_IMS_ARRAY	'M','Q','I','M','S','¬','¬','¬'
MQFMT_IMS_VAR_STRING_ARRAY	'M','Q','I','M','S','V','S','¬'
MQFMT_MD_EXTENSION_ARRAY	'M','Q','H','M','D','E','¬','¬'
MQFMT_PCF_ARRAY	'M','Q','P','C','F','¬','¬','¬'
MQFMT_REF_MSG_HEADER_ARRAY	'M','Q','H','R','E','F','¬','¬'
MQFMT_RF_HEADER_ARRAY	'M','Q','H','R','F','¬','¬','¬'
MQFMT_RF_HEADER_1_ARRAY	'M','Q','H','R','F','¬','¬','¬'
MQFMT_RF_HEADER_2_ARRAY	'M','Q','H','R','F','2','¬','¬'
MQFMT_STRING_ARRAY	'M','Q','S','T','R','¬','¬','¬'
MATRICE MQFMT_TRIGGER_ARRAY	'M','Q','T','R','I','G','¬','¬'
MQFMT_WORK_INFO_HEADER_ARRAY	'M','Q','H','W','I','H','¬','¬'
MQFMT_XMIT_Q_HEADER_ARRAY	'M','Q','X','M','I','T','¬','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

MQFUN_ * (Tipi di funzione dell'applicazione)

Tabella 170. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQFUN_TYPE_UNKNOWN	0	X'00000000'
JVM MQFUN_TYPE_	1	X'00000001'
MQFUN_TYPE_PROGRAM	2	X'00000002'
MQFUN_TYPE_PROCEDURE	3	X'00000003'
MQFUN_TYPE_USERDEF	4	X'00000004'
COMANDO MQFUN_TYPE_AND	5	X'00000005'

MQGA_ * (Selettori attributi gruppo)

Tabella 171. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQGA_FIRST	8001	X'00001F41'
MQGA_LAST	9000	X'00002328'

MQGACF_ * (Formato del comando Tipi di parametro gruppo)

Tabella 172. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQGACF_FIRST	8001	X'00001F41'
MQGACF_COMMAND_CONTEXT	8001	X'00001F41'
DATI COMMAND_MQGACF	8002	X'00001F42'
MQGACF_TRACE_ROUTE	8003	X'00001F43'
OPERAZIONE MQGACF	8004	X'00001F44'
ATTIVITÀ MQGACF	8005	X'00001F45'

Tabella 172. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQGACF_EMBEDDED_MQMD	8006	X'00001F46'
MESSAGE MQGACF	8007	X'00001F47'
MQGACF_MQM	8008	X'00001F48'
MQGACF_VALUE_NAMING	8009	X'00001F49'
DATI_CONTEGGIO_Q_MQGACF	8010	X'00001F4A'
MQGACF_Q_STATISTICS_DATA	8011	X'00001F4B'
MQGACF_CHL_STATISTICS_DATA	8012	X'00001F4C'
MQGACF_LAST_UTENTE	8012	X'00001F4C'

MQGI_ * (Identificatore gruppo)

Tabella 173. Nomi e valori costanti	
Nome	**Valore**
MQGI_NONE	X'00...00' (24 valori null)
MQGI_NON_ARRAY	'\0','\0',... (24 valori null)

MQGMO_ * (Richiama opzioni e struttura del messaggio)

Richiama la struttura delle opzioni del messaggio

Tabella 174. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQGMO	"GMO¬"
MATRICE MQGMO_STRUC_ID_ARRAY	'G','M','O','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 175. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQGMO_VERSION_1	1	X'00000001'
MQGMO_VERSION_2	2	X'00000002'
MQGMO_VERSION_3	3	X'00000003'
MQGMO_VERSION_4	4	X'00000004'
VERSIONE MQGMO_CURRENT_	4	X'00000004'

Acquisisci opzioni messaggio

Tabella 176. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQGMO_WAIT	1	X'00000001'
MQGMO_NO_WAIT	0	X'00000000'
MQGMO_SET_SIGNAL	8	X'00000008'
MQGMO_FAIL_IF_QUIESCING	8192	X'00002000'
SYNCPOINT MQGMO	2	X'00000002'

Tabella 176. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQGMO_SYNCPOINT_IF_PERSISTENT	4096	X'00001000'
MQGMO_NO_SYNCPOINT	4	X'00000004'
MQGMO_MARK_SKIP_BACKOUT	128	X'00000080'
MQGMO_BROWSE_FIRST	16	X'00000010'
MQGMO_BROWSE_SUCCESSIVO	32	X'00000020'
MQGMO_BROWSE_MSG_UNDER_CURSOR	2048	X'00000800'
MQGMO_BROWSE_HANDLE	17825808	X'01100010'
MQGMO_BROWSE_CO_OP	18874384	X'01200010'
MQGMO_MSG_UNDER_CURSOR	256	X'00000100'
LOCK_MQGMO	512	X'00000200'
MQGMO_UNLOCK	1024	X'00000400'
MQGMO_ACCEPT_TRUNCATED_MSG	64	X'00000040'
MQGMO_CONVERT	16384	X'00004000'
ORDER LOGICAL_MQGMO_	32768	X'00008000'
MQGMO_COMPLETE_MSG	65536	X'00010000'
MQGMO_ALL_MSGS_AVAILABLE	131072	X'00020000'
MQGMO_ALL_SEGMENTS_AVAILABLE	262144	X'00040000'
MQGMO_MARK_BROWSE_HANDLE	1048576	X'00100000'
MQGMO_MARK_BROWSE_CO_OP	2097152	X'00200000'
MQGMO_UNMARK_BROWSE_CO_OP	4194304	X'00400000'
MQGMO_UNMARK_BROWSE_HANDLE	8388608	X'00800000'
MQGMO_UNMARKED_BROWSE_MSG	16777216	X'01000000'
MQGMO_PROPERTIES_FORCE_MQRFH2	33554432	X'02000000'
MQGMO_NO_PROPERTIES	67108864	X'04000000'
MQGMO_PROPERTIES_IN_HANDLE	134217728	X'08000000'
MQGMO_PROPERTIES_COMPATIBILITY	268435456	X'10000000'
MQGMO_PROPERTIES_AS_Q_DEF	0	X'00000000'
MQGMO_NONE	0	X'00000000'

MQGS_ * (Stato gruppo)

Tabella 177. Nomi e valori costanti	
Nome	**Valore**
MQGS_NOT_IN_GROUP	'¬'
MQGS_MSG_IN_GROUP	'G'
MQGS_LAST_MSG_IN_GROUP	'L'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

MQHA_ * (Gestisci selettori)

Tabella 178. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQHA_FIRST	4001	X'00000FA1'
MQHA_BAG_HANDLE	4001	X'00000FA1'
MQHA_LAST_UTENTE	4001	X'00000FA1'
MQHA_LAST	6000	X'00001770'

MQHB_ * (Maniglie borsa)

Tabella 179. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQHB_UNUSABLE_HBAG	-1	X'FFFFFFFF'
MQHB_NONE	-2	X'FFFFFFFE'

MQHC_ * (Handle di connessione)

Tabella 180. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
DEF_MQH_HCONN	0	X'00000000'
MQH_UNUSABLE_HCONN	-1	X'FFFFFFFF'
MQHC_UNASSOCIAT_HCONN	-3	X'FFFFFFFD'

MQHM_ * (handle del messaggio)

Tabella 181. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQHM_UNUSABLE_HMSG	-1	X'FFFFFFFF'
MQHM_NONE	0	X'00000000'

MQHO_ * (handle oggetto)

Tabella 182. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQHO_UNUSABLE_HOBJ	-1	X'FFFFFFFF'
MQHO_NONE	0	X'00000000'

MQHSTATE_ * (Stati handle formato comando)

Tabella 183. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQHSTATE_INACTIVE	0	X'00000000'
MQHSTATE_ATTIVO	1	X'00000001'

MQIA_ * (Selettori di attributi interi)

Tabella 184. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQIA_ACCOUNTING_CONN_OVERRIDE	136	X'00000088'
MQIA_ACCOUNTING_INTERVAL	135	X'00000087'
MQIA_ACCOUNTING_MQI	133	X'00000085'
MQIA_ACCOUNTING_Q	134	X'00000086'
MQIA_ACTIVE_CHANNELS	100	X'00000064'
MQIA_ACTIVITY_CONN_OVERRIDE	239	X'000000EF'
MQIA_ACTIVITY_RECORDING	138	X'0000008A'
MQIA_ACTIVITY_TRACE	240	X'000000F0'
MQIA_ADOPTNEWMCA_CHECK	102	X'00000066'
MQIA_ADOPTNEWMCA_INTERVAL	104	X'00000068'
MQIA_ADOPTNEWMCA_TYPE	103	X'00000067'
MQIA_ADOPT_CONTEXT	260	X'00000104'
MQ Adv. VUE MQ Adv. VUE MQIA_ADVANCED_CAPABILITY	273	X'00000111'
MQIA_AMQP_CAPABILITY	265	X'00000109'
MQIA_APPL_TYPE	1	X'00000001'
MQIA_ARCHIVE	60	X'0000003C'
MQIA_AUTHENTICATION_FAIL_DELAY	259	X'00000103'
MQIA_AUTHENTICATION_METHOD	266	X'0000010A'
MQIA_AUTH_INFO_TYPE	66	X'00000042'
MQIA_AUTHORITY_EVENT	47	X'0000002F'
MQIA_AUTO_REORG_INTERVAL	174	X'000000AE'
MQIA_AUTO_REORGANIZATION	173	X'000000AD'
MQIA_BACKOUT_THRESHOLD	22	X'00000016'
MQIA_BASE_TYPE	193	X'000000C1'
MQIA_BATCH_INTERFACE_AUTO	86	X'00000056'
MQIA_BRIDGE_EVENT	74	X'0000004A'
MQIA_CF_LEVEL	70	X'00000046'
MQIA_CF_RECOVER	71	X'00000047'
MQIA_CHANNEL_AUTO_DEF	55	X'00000037'
MQIA_CHANNEL_AUTO_DEF_EVENT	56	X'00000038'
MQIA_CHANNEL_EVENT	73	X'00000049'
MQIA_CHECK_CLIENT_BINDING	258	X'00000102'
MQIA_CHECK_LOCAL_BINDING	257	X'00000101'
MQIA_CHINIT_ADAPTERS	101	X'00000065'
MQIA_CHINIT_CONTROL	119	X'00000077'
MQIA_CHINIT_DISPATCHERS	105	X'00000069'
MQIA_CHINIT_TRACE_AUTO_START	117	X'00000075'
MQIA_CHINIT_TRACE_TABLE_SIZE	118	X'00000076'

Tabella 184. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQIA_CLUSTER_OBJECT_STATE	256	X'00000100'
MQIA_CLUSTER_PUB_ROUTE	255	X'000000FF'
MQIA_CLUSTER_Q_TYPE	59	X'0000003B'
MQIA_CLUSTER_WORKLOAD_LENGTH	58	X'0000003A'
MQIA_CLWL_MRU_CHANNELS	97	X'00000061'
MQIA_CLWL_Q_RANK	95	X'0000005F'
MQIA_CLWL_Q_PRIORITY	96	X'00000060'
MQIA_CLWL_USEQ	98	X'00000062'
MQIA_CMD_SERVER_AUTO	87	X'00000057'
MQIA_CMD_SERVER_CONTROL	120	X'00000078'
MQIA_CMD_SERVER_CONVERT_MSG	88	X'00000058'
MQIA_CMD_SERVER_DLQ_MSG	89	X'00000059'
MQIA_CODED_CHAR_SET_ID	2	X'00000002'
MQIA_COMM_EVENT	232	X'000000E8'
MQIA_COMMAND_EVENT	99	X'00000063'
MQIA_COMMAND_LEVEL	31	X'0000001F'
MQIA_CONFIGURATION_EVENT	51	X'00000033'
MQIA_CPI_LEVEL	27	X'0000001B'
MQIA_CURRENT_Q_DEPTH	3	X'00000003'
MQIA_DEF_BIND	61	X'0000003D'
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	250	X'000000FA'
MQIA_DEF_INPUT_OPEN_OPTION	4	X'00000004'
MQIA_DEF_PERSISTENCE	5	X'00000005'
MQIA_DEF_PRIORITY	6	X'00000006'
MQIA_DEF_PUT_RESPONSE_TYPE	184	X'000000B8'
MQIA_DEF_READ_AHEAD	188	X'000000BC'
MQIA_DEFINITION_TYPE	7	X'00000007'
MQIA_DISPLAY_TYPE	262	X'00000106'
MQIA_DIST_LISTS	34	X'00000022'
MQIA_DNS_WLM	106	X'0000006A'
MQIA_DURABLE_SUB	175	X'000000AF'
MQIA_EXPIRY_INTERVAL	39	X'00000027'
MQIA_FIRST	1	X'00000001'
z/OS MQIA_GROUP_UR	221	X'000000DD'
MQIA_HARDEN_GET_BACKOUT	8	X'00000008'
MQIA_HIGH_Q_DEPTH	36	X'00000024'
MQIA_IGQ_PUT_AUTHORITY	65	X'00000041'
MQIA_INDEX_TYPE	57	X'00000039'
MQIA_INHIBIT_EVENT	48	X'00000030'

Tabella 184. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQIA_INHIBIT_GET	9	X'00000009'
MQIA_INHIBIT_PUB	181	X'000000B5'
MQIA_INHIBIT_PUT	10	X'0000000A'
MQIA_INHIBIT_SUB	182	X'000000B6'
z/OS MQIA_INTRA_GROUP_QUEUING	64	X'00000040'
MQIA_IP_ADDRESS_VERSION	93	X'0000005D'
MQIA_KEY_REUSE_COUNT	267	X'0000010B'
MQIA_LAST	2000	X'000007D0'
MQIA_LAST_USED	267	X'0000010B'
MQIA_LDAP_AUTHORMD	263	X'00000107'
MQIA_LDAP_NESTGRP	264	X'00000108'
MQIA_LDAP_SECURE_COMM	261	X'00000105'
MQIA_LISTENER_PORT_NUMBER	85	X'00000055'
MQIA_LISTENER_TIMER	107	X'0000006B'
MQIA_LOGGER_EVENT	94	X'0000005E'
MQIA_LU62_CHANNELS	108	X'0000006C'
MQIA_LOCAL_EVENT	49	X'00000031'
MQIA_MSG_MARK_BROWSE_INTERVAL	68	X'00000044'
MQIA_MAX_CHANNELS	109	X'0000006D'
MQIA_MAX_CLIENTS	172	X'000000AC'
MQIA_MAX_GLOBAL_LOCKS	83	X'00000053'
MQIA_MAX_HANDLES	11	X'0000000B'
MQIA_MAX_LOCAL_LOCKS	84	X'00000054'
MQIA_MAX_MSG_LENGTH	13	X'0000000D'
MQIA_MAX_OPEN_Q	80	X'00000050'
MQIA_MAX_PRIORITY	14	X'0000000E'
MQIA_MAX_PROPERTIES_LENGTH	192	X'000000C0'
MQIA_MAX_Q_DEPTH	15	X'0000000F'
MQIA_MAX_Q_TRIGGERS	90	X'0000005A'
MQIA_MAX_RECOVERY_TASKS	171	X'000000AB'
MQIA_MAX_UNCOMMITTED_MSGS	33	X'00000021'
MQIA_MCAST_BRIDGE	233	X'000000E9'
MQIA_MONITOR_INTERVAL	81	X'00000051'
MQIA_MONITORING_AUTO_CLUSSDR	124	X'0000007C'
MQIA_MONITORING_CHANNEL	122	X'0000007A'
MQIA_MONITORING_Q	123	X'0000007B'
MQIA_MSG_DELIVERY_SEQUENCE	16	X'00000010'
MQIA_MSG_DEQ_COUNT	38	X'00000026'
MQIA_MSG_ENQ_COUNT	37	X'00000025'

Tabella 184. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQIA_NAME_COUNT	19	X'00000013'
MQIA_NAMELIST_TYPE	72	X'00000048'
MQIA_NPM_CLASS	78	X'0000004E'
MQIA_NPM_DELIVERY	196	X'000000C4'
MQIA_OPEN_INPUT_COUNT	17	X'00000011'
MQIA_OPEN_OUTPUT_COUNT	18	X'00000012'
MQIA_OUTBOUND_PORT_MAX	140	X'0000008C'
MQIA_OUTBOUND_PORT_MIN	110	X'0000006E'
MQIA_PAGESET_ID	62	X'0000003E'
MQIA_PERFORMANCE_EVENT	53	X'00000035'
MQIA_PLATFORM	32	X'00000020'
MQIA_PM_DELIVERY	195	X'000000C3'
MQIA_PROPERTY_CONTROL	190	X'000000BE'
MQIA_PROT_POLICY_CAPABILITY	251	X'000000FB'
MQIA_PROXY_SUB	199	X'000000C7'
MQIA_PUB_COUNT	215	X'000000D7'
MQIA_PUB_SCOPE	219	X'000000DB'
MQIA_PUBSUB_CLUSTER	249	X'000000F9'
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	206	X'000000CE'
MQIA_PUBSUB_MODE	187	X'000000BB'
MQIA_PUBSUB_NP_MSG	203	X'000000CB'
MQIA_PUBSUB_NP_RESP	205	X'000000CD'
MQIA_PUBSUB_SYNC_PT	207	X'000000CF'
MQIA_Q_DEPTH_HIGH_EVENT	43	X'0000002B'
MQIA_Q_DEPTH_HIGH_LIMIT	40	X'00000028'
MQIA_Q_DEPTH_LOW_EVENT	44	X'0000002C'
MQIA_Q_DEPTH_LOW_LIMIT	41	X'00000029'
MQIA_Q_DEPTH_MAX_EVENT	42	X'0000002A'
MQIA_Q_SERVICE_INTERVAL	54	X'00000036'
MQIA_Q_SERVICE_INTERVAL_EVENT	46	X'0000002E'
MQIA_Q_TYPE	20	X'00000014'
MQIA_Q_USERS	82	X'00000052'
MQIA_QMGR_CFCONLOS	245	X'000000F5'
MQIA_QMOPT_CONS_COMMS_MSGS	155	X'0000009B'
MQIA_QMOPT_CONS_CRITICAL_MSGS	154	X'0000009A'
MQIA_QMOPT_CONS_ERROR_MSGS	153	X'00000099'
MQIA_QMOPT_CONS_INFO_MSGS	151	X'00000097'
MQIA_QMOPT_CONS_REORG_MSGS	156	X'0000009C'
MQIA_QMOPT_CONS_SYSTEM_MSGS	157	X'0000009D'

Tabella 184. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQIA_QMOPT_CONS_WARNING_MSGS	152	X'00000098'
MQIA_QMOPT_CSMT_ON_ERROR	150	X'00000096'
MQIA_QMOPT_INTERNAL_DUMP	170	X'000000AA'
MQIA_QMOPT_LOG_COMMS_MSGS	162	X'000000A2'
MQIA_QMOPT_LOG_CRITICAL_MSGS	161	X'000000A1'
MQIA_QMOPT_LOG_ERROR_MSGS	160	X'000000A0'
MQIA_QMOPT_LOG_INFO_MSGS	158	X'0000009E'
MQIA_QMOPT_LOG_REORG_MSGS	163	X'000000A3'
MQIA_QMOPT_LOG_SYSTEM_MSGS	164	X'000000A4'
MQIA_QMOPT_LOG_WARNING_MSGS	159	X'0000009F'
MQIA_QMOPT_TRACE_COMMS	166	X'000000A6'
MQIA_QMOPT_TRACE_CONVERSION	168	X'000000A8'
MQIA_QMOPT_TRACE_REORG	167	X'000000A7'
MQIA_QMOPT_TRACE_MQI_CALLS	165	X'000000A5'
MQIA_QMOPT_TRACE_SYSTEM	169	X'000000A9'
MQIA_QSG_DISP	63	X'0000003F'
MQIA_READ_AHEAD	189	X'000000BD'
MQIA_RECEIVE_TIMEOUT	111	X'0000006F'
MQIA_RECEIVE_TIMEOUT_MIN	113	X'00000071'
MQIA_RECEIVE_TIMEOUT_TYPE	112	X'00000070'
MQIA_REMOTE_EVENT	50	X'00000032'
MQIA_RETENTION_INTERVAL	21	X'00000015'
MQIA_REVERSE_DNS_LOOKUP	254	X'000000FE'
MQIA_SCOPE	45	X'0000002D'
MQIA_SECURITY_CASE	141	X'0000008D'
MQIA_SERVICE_CONTROL	139	X'0000008B'
MQIA_SERVICE_TYPE	121	X'00000079'
MQIA_SHAREABILITY	23	X'00000017'
MQIA_SHARED_Q_Q_MGR_NAME	77	X'0000004D'
MQIA_SSL_EVENT	75	X'0000004B'
MQIA_SSL_FIPS_REQUIRED	92	X'0000005C'
MQIA_SSL_RESET_COUNT	76	X'0000004C'
MQIA_SSL_TASKS	69	X'00000045'
MQIA_START_STOP_EVENT	52	X'00000034'
MQIA_STATISTICS_CHANNEL	129	X'00000081'
MQIA_STATISTICS_AUTO_CLUSSDR	130	X'00000082'
MQIA_STATISTICS_INTERVAL	131	X'00000083'
MQIA_STATISTICS_MQI	127	X'0000007F'
MQIA_STATISTICS_Q	128	X'00000080'

Tabella 184. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQIA_SUB_COUNT	204	X'000000CC'
MQIA_SUB_SCOPE	218	X'000000DA'
MQIA_SYNCPOINT	30	X'0000001E'
MQIA_TCP_CHANNELS	114	X'00000072'
MQIA_TCP_KEEP_ALIVE	115	X'00000073'
MQIA_TCP_STACK_TYPE	116	X'00000074'
MQIA_TIME_SINCE_RESET	35	X'00000023'
MQIA_TOPIC_DEF_PERSISTENCE	185	X'000000B9'
MQIA_TOPIC_NODE_COUNT	253	X'000000FD'
MQIA_TOPIC_TYPE	208	X'000000D0'
MQIA_TRACE_ROUTE_RECORDING	137	X'00000089'
MQIA_TREE_LIFE_TIME	183	X'000000B7'
MQIA_TRIGGER_CONTROL	24	X'00000018'
MQIA_TRIGGER_DEPTH	29	X'0000001D'
MQIA_TRIGGER_INTERVAL	25	X'00000019'
MQIA_TRIGGER_MSG_PRIORITY	26	X'0000001A'
MQIA_TRIGGER_TYPE	28	X'0000001C'
MQIA_TRIGGER_RESTART	91	X'0000005B'
MQIA_USAGE	12	X'0000000C'
MQIA_USE_DEAD_LETTER_Q	234	X'000000EA'
MQIA_USER_LIST	2000	X'000007D0'
MQIA_WILDCARD_OPERATION	216	X'000000D8'
MQIA_XR_CAPABILITY	243	X'000000F3'

MQIACF_ * (Tipo di parametro numero intero in formato comando)

Tabella 185. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIACF_FIRST	1001	X'000003E9'
MQIACF_Q_MGR_ATTRS	1001	X'000003E9'
MQIACF_Q_ATTRS	1002	X'000003EA'
MQIACF_PROCESS_ATTRS	1003	X'000003EB'
MQIACF_NAMELIST_ATTRS	1004	X'000003EC'
FORCE_MQIACF	1005	X'000003ED'
REPLACE MQIACF	1006	X'000003EE'
PURGE MQIACF	1007	X'000003EF'
QUIESCE MQIACF	1008	X'000003F0'
MOD_MQIACF	1008	X'000003F0'
TUTTE le MQIACF	1009	X'000003F1'
TIPO_APPL_EVENTI MQIACF	1010	X'000003F2'

Tabella 185. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
ORIGIN_EV_MQIACF	1011	X'000003F3'
ID_PARAMETER_MQIACF	1012	X'000003F4'
ID_ERRORE_MQIACF	1013	X'000003F5'
IDENTIFICATIVO_ERRORE_MQIACF_	1013	X'000003F5'
MQIACF_SELECTOR	1014	X'000003F6'
MQIACF_CHANNEL_ATTRS	1015	X'000003F7'
TIPO_OGGETTO_MQIAC	1016	X'000003F8'
TIPO_ESCAP_MQIACF	1017	X'000003F9'
ERRORE_MQIACF_OFFSET	1018	X'000003FA'
MQIACF_AUTH_INFO_ATTRS	1019	X'000003FB'
QUALIFICATORE_MOTIVO_MQIACF	1020	X'000003FC'
COMANDO MQIACF	1021	X'000003FD'
OPEN_MQIACF_OPZIONI	1022	X'000003FE'
TIPO_OPEN_MQIACF	1023	X'000003FF'
ID_PROCESSO MQIACF	1024	X'00000400'
ID_THREAD_MQIAC	1025	X'00000401'
MQIACF_Q_STATUS_ATTRS	1026	X'00000402'
MQIACF_UNCOMMITTED_MSGS	1027	X'00000403'
Stato_HANDLE_MQIACF	1028	X'00000404'
MQIACF_AUX_ERROR_DATA_INT_1	1070	X'0000042E'
MQIACF_AUX_ERROR_DATA_INT_2	1071	X'0000042F'
MQIACF_CONV_REASON_CODE	1072	X'00000430'
TIPO_BRIDGE_MQIACF	1073	X'00000431'
MQIACF_RICHIESTA	1074	X'00000432'
INTERVALLO_ATTESA_MQIACF	1075	X'00000433'
OPZIONI MQIACF	1076	X'00000434'
OPZIONI_BROKER_MQIACF_	1077	X'00000435'
TIPO_REFRESH_MQIACF	1078	X'00000436'
NUMERO_SEQUENZA_MQIACF_	1079	X'00000437'
DATI_INTEGER_MQIACF	1080	X'00000438'
OPZIONI MQIACF_REGISTRATION_	1081	X'00000439'
OPZIONI MQIACF_PUBLICATION_OPTIONS	1082	X'0000043A'
INFO CLUSTER_MQIACF	1083	X'0000043B'
MQIACF_Q_MGR_DEFINITION_TYPE	1084	X'0000043C'
TIPO_MGR_Q_MQIACF	1085	X'0000043D'
AZIONE MQIACF	1086	X'0000043E'
MQIACF_SOSPENSIONE	1087	X'0000043F'
CONTEGGIO_BROKER_MQIACF_	1088	X'00000440'
CONTO_APPL_MQIACF	1089	X'00000441'

Tabella 185. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
CONTEGGIO_ANONIMO_MQIACF	1090	X'00000442'
MQIACF_REG_REG_OPZIONI	1091	X'00000443'
MQIACF_DELETE_OPTIONS	1092	X'00000444'
MQIACF_CLUSTER_Q_MGR_ATTRS	1093	X'00000445'
INTERVALLO_AGGIORNATI_MQIACF_	1094	X'00000446'
MQIACF_REFRESH_REPOSITORY	1095	X'00000447'
MQIACF_REMOVE_QUEUES	1096	X'00000448'
TIPO_INPUT_OPEN_MQIACF	1098	X'0000044A'
MQIACF_OPEN_OUTPUT	1099	X'0000044B'
OPEN_MQIACF_SET	1100	X'0000044C'
MQIACF_OPEN_INQUIRE	1101	X'0000044D'
MQIACF_OPEN_BROWSE	1102	X'0000044E'
TIPO_STATO_Q_MQIACF	1103	X'0000044F'
MQIACF_Q_HANDLE	1104	X'00000450'
STATO coda MQIACF	1105	X'00000451'
TIPO_SECURITY_MQIACF	1106	X'00000452'
MQIACF_CONNECTION_ATTRS	1107	X'00000453'
OPZIONI MQIACF_CONNECT_	1108	X'00000454'
TIPO_INFO_CONN MQIACF_CONN	1110	X'00000456'
CONN MQIACF_CONN_INFO_CONN	1111	X'00000457'
GESTORE_INFO_CONN_MQIACF_	1112	X'00000458'
MQIACF_CONN_INFO_ALL	1113	X'00000459'
MQIACF_AUTH_PROFILE_ATTRS	1114	X'0000045A'
ELENCO_AUTORIZZAZIONI MQIACF	1115	X'0000045B'
AUTO_AUTO_MQIACF	1116	X'0000045C'
MQIACF_AUTH_REMOVE_AUTORI	1117	X'0000045D'
TIPO_ENTITA_MQIACF	1118	X'0000045E'
INFO_COMMAND_MQIACF	1120	X'00000460'
MQIACF_CMDSCOPE_Q_MGR_COUNT	1121	X'00000461'
MQIACF_Q_MGR_SISTEMA	1122	X'00000462'
MQIACF_Q_MGR_EVENT	1123	X'00000463'
MGR_DQM MQIACF_Q	1124	X'00000464'
MQIACF_Q_MGR_CLUSTER	1125	X'00000465'
DISPS QSG_MQIACF	1126	X'00000466'
UOW_MQIACF_STATO	1128	X'00000468'
MQIACF_SECURITY_ITEM	1129	X'00000469'
STRUC_CF_MQIACF_STATO	1130	X'0000046A'
TIPO_UOW_MQIACF	1132	X'0000046C'
MQIACF_CF_STRUC_ATTRS	1133	X'0000046D'

Tabella 185. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
INTERVALLO_ESCLUSO_MQIACF	1134	X'0000046E'
TIPO_STATO_CF_MQIACF	1135	X'0000046F'
MQIACF_CF_STATUS_SUMMARY	1136	X'00000470'
MQIACF_CF_STATUS_CONNECT	1137	X'00000471'
STATO_CF_MQIACF_BACKUP	1138	X'00000472'
TIPO_STRUC_CF_MQIACF	1139	X'00000473'
MQIACF_CF_STRUC_SIZE_MAX	1140	X'00000474'
MQIACF_CF_STRUC_SIZE_USED	1141	X'00000475'
MQIACF_CF_STRUC_ENTRIES_MAX	1142	X'00000476'
MQIACF_CF_STRUC_ENTRIES_USED	1143	X'00000477'
DIMENSIONE MQIACF_CF_STRUC_BACKUP_	1144	X'00000478'
TIPO_MOVE_MQIACF	1145	X'00000479'
MQIACF_MOVE_TYPE_MOVE	1146	X'0000047A'
MQIACF_MOVE_TYPE_ADD	1147	X'0000047B'
NUMERO MQIACF_Q_MGR_	1148	X'0000047C'
MGR_STATO Q_MQIACF	1149	X'0000047D'
MQIACF_DB2_CONN_STATUS	1150	X'0000047E'
MQIACF_SECURITY_ATTRS	1151	X'0000047F'
TIMEOUT MQIACF_SECURITY_	1152	X'00000480'
INTERVALLO_SICUREZZA_MQIACF	1153	X'00000481'
COMMUTAZIONE_SICUREZZA_MQIACF	1154	X'00000482'
IMPOSTAZIONI MQIACF_SECURITY_SETTING	1155	X'00000483'
MQIACF_STORAGE_CLASS_ATTRS	1156	X'00000484'
TIPO_USAGE_MQIACF	1157	X'00000485'
ID_POOL_BUFFER_MQIACF	1158	X'00000486'
MQIACF_USAGE_TOTAL_PAGES	1159	X'00000487'
MQIACF_USAGE_UNUSED_PAGES	1160	X'00000488'
PAGINE MQIACF_USAGE_PERSIST_PAGES	1161	X'00000489'
MQIACF_USAGE_NONPERSIST_PAGES	1162	X'0000048A'
MQIACF_USAGE_RESTART_EXTENTS	1163	X'0000048B'
MQIACF_USAGE_EXPAND_COUNT	1164	X'0000048C'
STATO PAGESET_MQIACF	1165	X'0000048D'
MQIACF_USAGE_TOTAL_BUFFERS	1166	X'0000048E'
TIPO_DATA_USAGE_MQIACF_SET	1167	X'0000048F'
MQIACF_USAGE_PAGESET	1168	X'00000490'
MQIACF_USAGE_DATA_SET	1169	X'00000491'
MQIACF_USAGE_BUFFER_POOL	1170	X'00000492'
MQIACF_MOVE_CONT	1171	X'00000493'
MQIACF_EXPIRY_Q_COUNT	1172	X'00000494'

Tabella 185. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
OGGETTI_CONFIGURAZIONE_MQIACF_	1173	X'00000495'
EVENTI_CONFIGURAZIONE_MQIACF_EVENTI	1174	X'00000496'
TIPO_SYSP_MQIACF	1175	X'00000497'
MQIACF_SYSP_DEALLOC_INTERVAL	1176	X'00000498'
MQIACF_SYSP_MAX_ARCHIVE	1177	X'00000499'
TAPES READ_MAX_MQIACF_SYSPES	1178	X'0000049A'
DIMENSIONE MQIACF_SYSP_IN_BUFFER_	1179	X'0000049B'
DIMENSIONE MQIACF_SYSP_OUT_BUFFER_	1180	X'0000049C'
CONTA.MQIACF_SYSP_OUT_BUFFER_COUNT	1181	X'0000049D'
ARCHIVIO_SYSP_MQIACF_ARCHIVE	1182	X'0000049E'
MQIACF_SYSP_ATTIVO	1183	X'0000049F'
ARCHIVE MQIACF_SYSP_DUAL_	1184	X'000004A0'
MQIACF_SYSP_DUAL_BSDS	1185	X'000004A1'
MQIACF_SYSP_MAX_CONNS	1186	X'000004A2'
MQIACF_SYSP_MAX_CONNS_FORE	1187	X'000004A3'
MQIACF_SYSP_MAX_CONNS_BACK	1188	X'000004A4'
MQIACF_SYSP_EXIT_INTERVAL	1189	X'000004A5'
MQIACF_SYSP_EXIT_TASKS	1190	X'000004A6'
CONTA.MQIACF_SYSP_CHKPOINT_COUNT	1191	X'000004A7'
MQIACF_SYSP_OTMA_INTERVAL	1192	X'000004A8'
MQIACF_SYSP_Q_INDEX_DEFER	1193	X'000004A9'
MQIACF_SYSP_DB2_TASKS	1194	X'000004AA'
MQIACF_SYSP_RESLEVEL_AUDIT	1195	X'000004AB'
CODICE RIGA MQIACF_SYSP_	1196	X'000004AC'
MQIACF_SYSP_SMF_ACCOUNTING	1197	X'000004AD'
STATO_SMF MQIACF_SYSP_	1198	X'000004AE'
MQIACF_SYSP_SMF_INTERVAL	1199	X'000004AF'
MQIACF_SYSP_TRACE_CLASS	1200	X'000004B0'
DIMENSIONE MQIACF_SYSP_TRACE_	1201	X'000004B1'
MQIACF_SYSP_WLM_INTERVAL	1202	X'000004B2'
MQIACF_SYSP_ALLOC_UNIT	1203	X'000004B3'
MQIACF_SYSP_ARCHIVE_RETAIN	1204	X'000004B4'
MQIACF_SYSP_ARCHIVE_WTOR	1205	X'000004B5'
DIMENSIONE_BLOCK_SYSP_MQIACF_	1206	X'000004B6'
CATALOGO_SISTEMA_MQIACF_SISTEMA	1207	X'000004B7'
SYSP_COMPACT MQIACF	1208	X'000004B8'
MQIACF_SYSP_ALLOC_PRIMARY	1209	X'000004B9'
MQIACF_SYSP_ALLOC_SECONDARY	1210	X'000004BA'
PROTECT MQIACF_SYSP_	1211	X'000004BB'

Tabella 185. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
INTERVALLO_QUIESCE_SISTEMA_MQIACF_SYSP_	1212	X'000004BC'
MQIACF_SYSP_TIMESTAMP	1213	X'000004BD'
ADDRESS MQIACF_SYSP_UNIT_	1214	X'000004BE'
STATO UNITA MQIACF_SYSP_	1215	X'000004BF'
COPIA MQIACF_SYSP_LOG	1216	X'000004C0'
MQIACF_SYSP_LOG_USED	1217	X'000004C1'
MQIACF_SYSP_LOG_SUSPEND	1218	X'000004C2'
STATO FSYSP_OFFLOAD MQIAC	1219	X'000004C3'
MQIACF_SYSP_TOTAL_LOGS	1220	X'000004C4'
MQIACF_SYSP_FULL_LOGS	1221	X'000004C5'
MQIACF_LISTENER_ATTRS	1222	X'000004C6'
MQIACF_LISTENER_STATUS_ATTRS	1223	X'000004C7'
MQIACF_SERVICE_ATTRS	1224	X'000004C8'
MQIACF_SERVICE_STATUS_ATTRS	1225	X'000004C9'
MQIACF_Q_TIME_INDICATOR	1226	X'000004CA'
MQIACF_OLDEST_MSG_AGE	1227	X'000004CB'
OPZIONI MQIACF_AUTH_	1228	X'000004CC'
MQIACF_Q_MGR_STATUS_ATTRS	1229	X'000004CD'
CONTEGGIO_MQIACF_CONNESSIONE	1230	X'000004CE'
MQIACF_Q_MGR_FACILITY	1231	X'000004CF'
STATO_CHINIT_MQIACF	1232	X'000004D0'
STATO SERVER_CMD_MQIACF	1233	X'000004D1'
MQIACF_ROUTE_DETAIL	1234	X'000004D2'
MQIACF_RECORDED_ATTIVITÀ	1235	X'000004D3'
ATTIVITA ' MQIACF_MAX_	1236	X'000004D4'
CONTEGGIO_DISCONTINUITÀ_MQIACF_	1237	X'000004D5'
MQIACF_ACCUMULAZIONE_INSTRADAMENTO	1238	X'000004D6'
MQIACF_ROUTE_DELIVERY	1239	X'000004D7'
TIPO_OPERAZIONE MQIACF	1240	X'000004D8'
CONTEGGIO_BACKOUT_MQIAC	1241	X'000004D9'
CODICE_COMP_MQIACF	1242	X'000004DA'
MQIACF_ENCODING	1243	X'000004DB'
MQIACF_SCADENZA	1244	X'000004DC'
MQIACF_FEEDBACK	1245	X'000004DD'
FLAGS_MSG_MQIACF	1247	X'000004DF'
MQIACF_MSG_LENGTH	1248	X'000004E0'
TIPO_MSG_MQIACF	1249	X'000004E1'
MQIACF_OFFSET	1250	X'000004E2'
LUNGHEZZA_ORIGINALE_MQIACF_	1251	X'000004E3'

Tabella 185. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQIACF_PERSISTENZA	1252	X'000004E4'
PRIORITà_MQIACF	1253	X'000004E5'
CODICE_REASON_MQIACF	1254	X'000004E6'
REPORT MQIACF	1255	X'000004E7'
VERSIONE MQIACF	1256	X'000004E8'
MQIACF_UNRECORDED_ATTIVITÀ	1257	X'000004E9'
MQIACF_MONITORAGGIO	1258	X'000004EA'
MQIACF_ROUTE_FORWARDING	1259	X'000004EB'
STATO SERVICE_MQIACF	1260	X'000004EC'
TIPO_Q_MQIACF	1261	X'000004ED'
SUPPO_ID_UTENTE_MQIACF	1262	X'000004EE'
MQIACF_INTERFACE_VERSIONE	1263	X'000004EF'
ATTRS MQIACF_AUTH_SERVICE_	1264	X'000004F0'
TIPO_MQIACF_USAGE_EXPAND_TIPO	1265	X'000004F1'
MQIACF_SYSP_CLUSTER_CACHE	1266	X'000004F2'
MQIACF_SYSP_DB2_BLOB_TASKS	1267	X'000004F3'
UNITS MQIACF_SYSP_WLM_INT_UNITS	1268	X'000004F4'
TOPIC_ATTRS MQIACF	1269	X'000004F5'
MQIACF_PUBSUB_PROPERTIES	1271	X'000004F7'
CLASSE_DESTINAZIONE_MQIACF	1273	X'000004F9'
MQIACF_DURABLE_SUBSCRIPTION	1274	X'000004FA'
MQIACF_SOTTOSCRIZIONE_AMBITO	1275	X'000004FB'
ID_UTENTE_VARIABILE_MQIACF_	1277	X'000004FD'
SOLO MQIACF_REQUEST_ONLY	1280	X'00000500'
MQIACF_PUB_PRIORITY	1283	X'00000503'
MQIACF_SUB_ATTRS	1287	X'00000507'
SCHEMA MQIACF_WILDCARD_SCHEMA	1288	X'00000508'
TIPO_SUB_MQIACF	1289	X'00000509'
MQIACF_MESSAGE_XX_ENCODE_CASE_ONE conteggio	1290	X'0000050A'
MQIACF_Q_MGR_PUBSUB	1291	X'0000050B'
MQIACF_Q_MGR_VERSIONE	1292	X'0000050C'
MQIACF_SUB_STATUS_ATTRS	1294	X'0000050E'
STATO TOPIC_MQIACF	1295	X'0000050F'
SUB TOPIC_MQIACF	1296	X'00000510'
MQIACF_TOPIC_PUB	1297	X'00000511'
MQIACF_RETAINED_PUBLICATION	1300	X'00000514'
MQIACF_TOPIC_STATUS_ATTRS	1301	X'00000515'
TIPO_STATO_TOPIC_MQIACF	1302	X'00000516'
MQIACF_SUB_OPZIONI	1303	X'00000517'

Tabella 185. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
CONTEGGIO_PUBBLICAZIONE_MQIACF	1304	X'00000518'
TIPO_CLEAR_MQIACF	1305	X'00000519'
Ambito MQIACF_CLEAR_SCOPE	1306	X'0000051A'
LIVELLO_SOTTO_MQIACF	1307	X'0000051B'
STATO_ASYNC_MQIACF	1308	X'0000051C'
RIEPILOGO MQIACF_SUB_SUMMARY	1309	X'0000051D'
MQIACF_OBSOLETE_MSGS	1310	X'0000051E'
STATO SOTTOSCRIZIONE PUBBLICA MQIACF	1311	X'0000051F'
TIPO_STATO_PS_MQIACF	1314	X'00000522'
MQIACF_PUBSUB_STATUS_ATTRS	1318	X'00000526'
TIPO_SELECTOR_MQIACF	1321	X'00000529'
MQIACF_MCAST_REL_INDICATOR	1351	X'00000547'
TIPO_CHLAUTO_MQIACF	1352	X'00000548'
TIPO MQXR_DIAGNOSTICHE	1354	X'0000054A'
MQIACF_CHLAUTH_ATTRS	1355	X'0000054B'
ID_OPERAZIONE MQIACF	1356	X'0000054C'
TIPO_CALLER_API_MQIACF	1357	X'0000054D'
AMBIENTE API MQIACF_AMBIENTE	1358	X'0000054E'
DETTAGLI MQIACF_TRACE_DETAIL	1359	X'0000054F'
HOBJ MQIACF	1360	X'00000550'
TIPO_CALL_MQIACF	1361	X'00000551'
OPERAZIONE MQIACF_MQCB	1362	X'00000552'
TIPO_MQCB_MQIACF	1363	X'00000553'
MQIACF_MQCB_OPZIONI	1364	X'00000554'
OPZIONI_CHIUDI_MQIACF	1365	X'00000555'
OPERAZIONE MQIACF_CTL	1366	X'00000556'
OPZIONI MQIACF_GET	1367	X'00000557'
MQIACF_RECS_PRESENTE	1368	X'00000558'
CONTEGGIO_DEST_CONOSCIUTO_MQIACF_	1369	X'00000559'
MQIACF_UNKNOWN_DEST_COUNT	1370	X'0000055A'
MQIACF_INVALID_DEST_COUNT	1371	X'0000055B'
TIPO_RESOLV_MQIACF	1372	X'0000055C'
OPZIONI MQIACF_PUT_	1373	X'0000055D'
LUNGHEZZA_BUFFER_MQIACF_	1374	X'0000055E'
LUNGHEZZA_DATI_TRACCIA_MQIACF	1375	X'0000055F'
MQIACF_SMDS_EXPANDST	1376	X'00000560'
LUNGHEZZA_STRUTTURA_MQIACF	1377	X'00000561'
CONTEGGIO_ELEMENTO_MQIACF	1378	X'00000562'
ORA_MQIACF_EXPIRY_TIME	1379	X'00000563'

Tabella 185. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
TEMPO MQIACF_CONNECT_TIME	1380	X'00000564'
ORA MQIACF_DISCONNECT_TIME	1381	X'00000565'
HSUB MQIACF	1382	X'00000566'
MQIACF_SUBRQ_OPZIONI	1383	X'00000567'
IDRM_XA_MQIACF	1384	X'00000568'
MQIACF_XA_FLAGS	1385	X'00000569'
CODICE_XA_MQIACF	1386	X'0000056A'
MQIACF_XA_HANDLE	1387	X'0000056B'
XA_RETVAL MQIACF_AL	1388	X'0000056C'
TIPO_STATO_MQIACF	1389	X'0000056D'
CONTO_XA_MQIACF	1390	X'0000056E'
MQIACF_SELECTOR_COUNT	1391	X'0000056F'
SELECTORS MQIACF	1392	X'00000570'
Conteggio_INTATTR_MQIACF_	1393	X'00000571'
INTATTRS MQIACF	1394	X'00000572'
AZIONE MQIACF_SUBRQ_	1395	X'00000573'
NUM_PUBS MQIACF	1396	X'00000574'
DIMENSIONE_POINTER_MQIACF	1397	X'00000575'
MQIACF_REMOVE_AUTOREC	1398	X'00000576'
MQIACF_XR_ATTRS	1399	X'00000577'
TIPO_FUNZIONE_APPL_MQIACF_	1400	X'00000578'
MQIACF_AMQP_ATTRS	1401	X'00000579'
TIPO_EXPORT_MQIACF	1402	X'0000057A'
MQIACF_EXPORT_ATTRS	1403	X'0000057B'
MQIACF_SYSTEM_OBJECTS	1404	X'0000057C'
MQIACF_SWAP_CONNESSIONE	1405	X'0000057D'
TIPO MQIACF_AMQP_DIAGNOSTICS_TYPE	1406	X'0000057E'
MQIACF_BUFFER_POOL_LOCATION	1408	X'00000580'
STATO DI CONNESSIONE MQIACF_LDAP_CONNECTION_DI	1409	X'00000581'
POOL DI MQIACF_SYSP_MAX_ACE_POOL	1410	X'00000582'
PAGECLAS MQIACF	1411	X'00000583'
TIPO_AUTO_MQIACF	1412	X'00000584'
MQIACF_SYSP_MAX_CONC_OFFLOADS	1413	X'00000585'
MQIACF_SYSP_ZHYPERWRITE	1414	X'00000586'
MQIACF_Q_MGR_STATUS_LOG	1415	X'00000587'
SIZE MQIACF_ARCHIVE_LOG_	1416	X'00000588'
MQIACF_MEDIA_LOG_SIZE	1417	X'00000589'
DIMENSIONE LOG MQIACF_RESTART_	1418	X'0000058A'
MQIACF_REUSABLE_LOG_SIZE	1419	X'0000058B'

Tabella 185. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQIACF_LOG_IN_USE	1420	X'0000058C'
UTILIZZO_LOG_MQIACF_	1421	X'0000058D'
MQIACF_IGNORE_STATO	1423	X'0000058F'
MQIACF_MOVABLE_APPL_COUNT	1424	X'00000590'
ATTR_INFO_APPL_MQIACF_	1425	X'00000591'
MQIACF_APPL_MOVABLE	1426	X'00000592'
MQIACF_REMOTE_QMGR_ACTIVE	1427	X'00000593'
TIPO_INFO_APPL_MQIACF	1428	X'00000594'
APPL_INFO_MQIACF_APPL	1429	X'00000595'
MQIACF_APPL_INFO_QMGR	1430	X'00000596'
MQIACF_APPL_INFO_LOCAL	1431	X'00000597'
MQIACF_APPL_IMMOVABLE_COUNT	1432	X'00000598'
MQIACF_BALANCED	1433	X'00000599'
BALSTATO MQIACF	1434	X'0000059A'
MQIACF_APPL_IMMOVABLE_REASON	1435	X'0000059B'
MQIACF_DS_ENCRYPTED	1436	X'0000059C'
DIMENSIONE MQIACF_CUR_Q_FILE_	1437	X'0000059D'
DIMENSIONE MQIACF_CUR_MAX_FILE_	1438	X'0000059E'
TIPO_BALANCING_MQIACF	1439	X'0000059F'
OPZIONI_BILANCIAMENTO_MQIACF_OPZIONI	1440	X'000005A0'
TIMEOUT balancing_mqiacf_	1441	X'000005A1'
MQIACF_SYSP_SMF_STAT_TIME_SECS	1442	X'000005A2'
MQIACF_SYSP_SMF_ACCT_TIME_MINS	1443	X'000005A3'
MQIACF_SYSP_SMF_ACCT_TIME_SECS	1444	X'000005A4'
MQIACF_LAST_UTENTE	1444	X'000005A4'

MQIACH_ * (Formato del comando Tipi di canali a numero intero)

Tabella 186. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIACH_FIRST	1501	X'000005DD'
MQIACH_XMIT_PROTOCOL_TYPE	1501	X'000005DD'
MQIACH_BATCH_SIZE	1502	X'000005DE'
MQIACH_DISC_INTERVAL	1503	X'000005DF'
MQIACH_SHORT_TIMER	1504	X'000005E0'
MQIACH_SHORT_RETRY	1505	X'000005E1'
MQIACH_LONG_TIMER	1506	X'000005E2'
MQIACH_LONG_RETRY	1507	X'000005E3'
MQIACH_PUT_AUTHORITY	1508	X'000005E4'
MQIACH_SEQUENCE_NUMBER_WRAP	1509	X'000005E5'

Tabella 186. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQIACH_MAX_MSG_LENGTH	1510	X'000005E6'
MQIACH_CHANNEL_TYPE	1511	X'000005E7'
MQIACH_DATA_COUNT	1512	X'000005E8'
MQIACH_NAME_COUNT	1513	X'000005E9'
MQIACH_MSG_SEQUENCE_NUMBER	1514	X'000005EA'
MQIACH_DATA_CONVERSION	1515	X'000005EB'
MQIACH_IN_DOUBT	1516	X'000005EC'
MQIACH_MCA_TYPE	1517	X'000005ED'
MQIACH_SESSION_COUNT	1518	X'000005EE'
MQIACH_ADAPTER	1519	X'000005EF'
MQIACH_COMMAND_COUNT	1520	X'000005F0'
MQIACH_SOCKET	1521	X'000005F1'
MQIACH_PORT	1522	X'000005F2'
MQIACH_CHANNEL_INSTANCE_TYPE	1523	X'000005F3'
MQIACH_CHANNEL_INSTANCE_ATTRS	1524	X'000005F4'
MQIACH_CHANNEL_ERROR_DATA	1525	X'000005F5'
MQIACH_CHANNEL_TABLE	1526	X'000005F6'
MQIACH_CHANNEL_STATUS	1527	X'000005F7'
MQIACH_INDOUBT_STATUS	1528	X'000005F8'
MQIACH_LAST_SEQ_NUMBER	1529	X'000005F9'
MQIACH_LAST_SEQUENCE_NUMBER	1529	X'000005F9'
MQIACH_CURRENT_MSGS	1531	X'000005FB'
MQIACH_CURRENT_SEQ_NUMBER	1532	X'000005FC'
MQIACH_CURRENT_SEQUENCE_NUMBER	1532	X'000005FC'
MQIACH_SSL_RETURN_CODE	1533	X'000005FD'
MQIACH_MSGS	1534	X'000005FE'
MQIACH_BYTES_SENT	1535	X'000005FF'
MQIACH_BYTES_RCVD	1536	X'00000600'
MQIACH_BYTES_RECEIVED	1536	X'00000600'
MQIACH_BATCHES	1537	X'00000601'
MQIACH_BUFFERS_SENT	1538	X'00000602'
MQIACH_BUFFERS_RCVD	1539	X'00000603'
MQIACH_BUFFERS_RECEIVED	1539	X'00000603'
MQIACH_LONG_RETRIES_LEFT	1540	X'00000604'
MQIACH_SHORT_RETRIES_LEFT	1541	X'00000605'
MQIACH_MCA_STATUS	1542	X'00000606'
MQIACH_STOP_REQUESTED	1543	X'00000607'
MQIACH_MR_COUNT	1544	X'00000608'
MQIACH_MR_INTERVAL	1545	X'00000609'

Tabella 186. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQIACH_NPM_SPEED	1562	X'0000061A'
MQIACH_HB_INTERVAL	1563	X'0000061B'
MQIACH_BATCH_INTERVAL	1564	X'0000061C'
MQIACH_NETWORK_PRIORITY	1565	X'0000061D'
MQIACH_KEEP_ALIVE_INTERVAL	1566	X'0000061E'
MQIACH_BATCH_HB	1567	X'0000061F'
MQIACH_SSL_CLIENT_AUTH	1568	X'00000620'
MQIACH_ALLOC_RETRY	1570	X'00000622'
MQIACH_ALLOC_FAST_TIMER	1571	X'00000623'
MQIACH_ALLOC_SLOW_TIMER	1572	X'00000624'
MQIACH_DISC_RETRY	1573	X'00000625'
MQIACH_PORT_NUMBER	1574	X'00000626'
MQIACH_HDR_COMPRESSION	1575	X'00000627'
MQIACH_MSG_COMPRESSION	1576	X'00000628'
MQIACH_CLWL_CHANNEL_RANK	1577	X'00000629'
MQIACH_CLWL_CHANNEL_PRIORITY	1578	X'0000062A'
MQIACH_CLWL_CHANNEL_WEIGHT	1579	X'0000062B'
MQIACH_CHANNEL_DISP	1580	X'0000062C'
MQIACH_INBOUND_DISP	1581	X'0000062D'
MQIACH_CHANNEL_TYPES	1582	X'0000062E'
MQIACH_ADAPS_STARTED	1583	X'0000062F'
MQIACH_ADAPS_MAX	1584	X'00000630'
MQIACH_DISPS_STARTED	1585	X'00000631'
MQIACH_DISPS_MAX	1586	X'00000632'
MQIACH_SSLTASKS_STARTED	1587	X'00000633'
MQIACH_SSLTASKS_MAX	1588	X'00000634'
MQIACH_CURRENT_CHL	1589	X'00000635'
MQIACH_CURRENT_CHL_MAX	1590	X'00000636'
MQIACH_CURRENT_CHL_TCP	1591	X'00000637'
MQIACH_CURRENT_CHL_LU62	1592	X'00000638'
MQIACH_ACTIVE_CHL	1593	X'00000639'
MQIACH_ACTIVE_CHL_MAX	1594	X'0000063A'
MQIACH_ACTIVE_CHL_PAUSED	1595	X'0000063B'
MQIACH_ACTIVE_CHL_STARTED	1596	X'0000063C'
MQIACH_ACTIVE_CHL_STOPPED	1597	X'0000063D'
MQIACH_ACTIVE_CHL_RETRY	1598	X'0000063E'
MQIACH_LISTENER_STATUS	1599	X'0000063F'
MQIACH_SHARED_CHL_RESTART	1600	X'00000640'
MQIACH_LISTENER_CONTROL	1601	X'00000641'

Tabella 186. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQIACH_BACKLOG	1602	X'00000642'
MQIACH_XMITQ_TIME_INDICATOR	1604	X'00000644'
MQIACH_NETWORK_TIME_INDICATOR	1605	X'00000645'
MQIACH_EXIT_TIME_INDICATOR	1606	X'00000646'
MQIACH_BATCH_SIZE_INDICATOR	1607	X'00000647'
MQIACH_XMITQ_MSGS_AVAILABLE	1608	X'00000648'
MQIACH_CHANNEL_SUBSTATE	1609	X'00000649'
MQIACH_SSL_KEY_RESETS	1610	X'0000064A'
MQIACH_COMPRESSION_RATE	1611	X'0000064B'
MQIACH_COMPRESSION_TIME	1612	X'0000064C'
MQIACH_MAX_XMIT_SIZE	1613	X'0000064D'
MQIACH_DEF_CHANNEL_DISP	1614	X'0000064E'
MQIACH_SHARING_CONVERSATIONS	1615	X'0000064F'
MQIACH_MAX_SHARING_CONVS	1616	X'00000650'
MQIACH_CURRENT_SHARING_CONVS	1617	X'00000651'
MQIACH_MAX_INSTANCES	1618	X'00000652'
MQIACH_MAX_INSTS_PER_CLIENT	1619	X'00000653'
MQIACH_CLIENT_CHANNEL_WEIGHT	1620	X'00000654'
MQIACH_CONNECTION_AFFINITY	1621	X'00000655'
MQIACH_AUTH_INFO_TYPES	1622	X'00000656'
MQIACH_RESET_REQUESTED	1623	X'00000657'
MQIACH_BATCH_DATA_LIMIT	1624	X'00000658'
MQIACH_MSG_HISTORY	1625	X'00000659'
MQIACH_MULTICAST_PROPERTIES	1626	X'0000065A'
MQIACH_NEW_SUBSCRIBER_HISTORY	1627	X'0000065B'
MQIACH_MC_HB_INTERVAL	1628	X'0000065C'
MQIACH_USE_CLIENT_ID	1629	X'0000065D'
MQIACH_MQTT_KEEP_ALIVE	1630	X'0000065E'
MQIACH_IN_DOUBT_IN	1631	X'0000065F'
MQIACH_IN_DOUBT_OUT	1632	X'00000660'
MQIACH_MSGS_SENT<	1633	X'00000661'
MQIACH_MSGS_RECEIVED	1634	X'00000662'
MQIACH_MSGS_RCVD	1634	X'00000662'
MQIACH_PENDING_OUT	1635	X'00000663'
MQIACH_AVAILABLE_CIPHERSPECS	1636	X'00000664'
MQIACH_MATCH	1637	X'00000665'
MQIACH_USER_SOURCE	1638	X'00000666'
MQIACH_WARNING	1639	X'00000667'
MQIACH_DEF_RECONNECT	1640	X'00000668'

Tabella 186. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQIACH_CHANNEL_SUMMARY_ATTRS	1642	X'0000066A'
MQIACH_PROTOCOL	1643	X'0000066B'
MQIACH_AMQPKEEPALIVE	1644	X'0000066C'
MQIACH_SECURITY_PROTOCOL	1645	X'0000066D'
z/OS MQIACH_SPL_PROTECTION	1646	X'0000066E'
MQIACH_LAST_USED	1646	X'0000066E'

MQIAMO_ * (Tipo di parametro monitoraggio numero intero formato comando)

Tabella 187. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIAMO_PRIMO	701	X'000002BD'
MQIAMO_AVG_BATCH_SIZE	702	X'000002BE'
MQIAMO_AVG_Q_TIME	703	X'000002BF'
BACKOUTS MQIAMO	704	X'000002C0'
MQIAMO_BROWSES	705	X'000002C1'
MQIAMO_BROWSE_MAX_BYTES	706	X'000002C2'
MQIAMO_BROWSE_MIN_BYTES	707	X'000002C3'
MQIAMO_BROWSES_FAILED	708	X'000002C4'
MQIAMO_CHIUSO	709	X'000002C5'
COMMIT MQIAMO	710	X'000002C6'
MQIAMO_COMMITS_FAILED	711	X'000002C7'
CONNS MQIAMO	712	X'000002C8'
MQIAMO_CONNS_MAX	713	X'000002C9'
DISCO MQIAMO	714	X'000002CA'
MQIAMO_DISCS_IMPLICIT	715	X'000002CB'
TIPO_DISC_MQIAMO	716	X'000002CC'
MQIAMO_EXIT_TIME_AVG	717	X'000002CD'
MQIAMO_EXIT_TIME_MAX	718	X'000002CE'
MIN MQIAMO_EXIT_TIME_	719	X'000002CF'
MQIAMO_BATCHES	720	X'000002D0'
MQIAMO_GENERATED_MSGS	721	X'000002D1'
GET_MQIAMO	722	X'000002D2'
MQIAMO_GET_MAX_BYTES	723	X'000002D3'
MQIAMO_GET_MIN_BYTES	724	X'000002D4'
MQIAMO_GETS_NON RIUSCITO	725	X'000002D5'
MQIAMO_INCOMPLETE_BATCH	726	X'000002D6'
INQS MQIAMO	727	X'000002D7'
MSGS_MQIAMO	728	X'000002D8'

Tabella 187. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQIAMO_NET_TIME_AVG	729	X'000002D9'
MQIAMO_NET_TIME_MAX	730	X'000002DA'
MQIAMO_NET_TIME_MIN	731	X'000002DB'
CONTEGGIO_OGGETTI_MQIAMO_	732	X'000002DC'
MQIAMO_APERTURE	733	X'000002DD'
MQIAMO_PUT1S	734	X'000002DE'
MQIAMO_PUTS	735	X'000002DF'
MQIAMO_PUT_MAX_BYTES	736	X'000002E0'
MQIAMO_PUT_MIN_BYTES	737	X'000002E1'
MQIAMO_PUT_RETRIES	738	X'000002E2'
MQIAMO_Q_MAX_DEPTH	739	X'000002E3'
MQIAMO_Q_MIN_DEPTH	740	X'000002E4'
MQIAMO_Q_TIME_AVG	741	X'000002E5'
MQIAMO_Q_TIME_MAX	742	X'000002E6'
MQIAMO_Q_TIME_MIN	743	X'000002E7'
SETS MQIAMO	744	X'000002E8'
MQIAMO_CONNS_NON RIUSCITO	749	X'000002ED'
MQIAMO_OPENS_FAILED	751	X'000002EF'
MQIAMO_INQS_NON RIUSCITO	752	X'000002F0'
MQIAMO_SETS_NON RIUSCITO	753	X'000002F1'
MQIAMO_PUTS_FAILED	754	X'000002F2'
MQIAMO_PUT1S_FAILED	755	X'000002F3'
MQIAMO_CLOSES_FAILED	757	X'000002F5'
MQIAMO_MSGS_EXPIRED	758	X'000002F6'
MQIAMO_MSGS_NOT_QUEUED	759	X'000002F7'
MQIAMO_MSGS_PURGED	760	X'000002F8'
MQIAMO_SUBS_DUR	764	X'000002FC'
MQIAMO_SUBS_NDUR	765	X'000002FD'
MQIAMO_SUBS_NON RIUSCITO	766	X'000002FE'
MQIAMO_SUBRQS	767	X'000002FF'
MQIAMO_SUBRQS_NON RIUSCITO	768	X'00000300'
Cbs MQIAMO	769	X'00000301'
MQIAMO_CBS_NON RIUSCITO	770	X'00000302'
CTLS MQIAMO	771	X'00000303'
MQIAMO_CTLS_NON RIUSCITO	772	X'00000304'
STATO_MQIAMO	773	X'00000305'
MQIAMO_STATS_NON RIUSCITO	774	X'00000306'
MQIAMO_SUB_DUR_HIGHWATER	775	X'00000307'
MQIAMO_SUB_DUR_LOWWATER	776	X'00000308'

Tabella 187. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQIAMO_SUB_NDUR_HIGHWATER	777	X'00000309'
MQIAMO_SUB_NDUR_LOWWATER	778	X'0000030A'
MQIAMO_TOPIC_PUTS	779	X'0000030B'
MQIAMO_TOPIC_PUTS_FAILED	780	X'0000030C'
MQIAMO_TOPIC_PUT1S	781	X'0000030D'
MQIAMO_TOPIC_PUT1S_FAILED	782	X'0000030E'
MQIAMO_PUBLISH_MSG_COUNT	784	X'00000310'
MQIAMO_UNSUBS_DUR	786	X'00000312'
MQIAMO_UNSUBS_NDUR	787	X'00000313'
MQIAMO_UNSUBS_NON RIUSCITO	788	X'00000314'
INTERVALLO_MQIAMO_	789	X'00000315'
MQIAMO_MSGS_SENT	790	X'00000316'
MQIAMO_BYTE_INVIATI	791	X'00000317'
MQIAMO_REPAIR_BYTES	792	X'00000318'
MODALITÀ_FEEDBACK_MQIAMO_	793	X'00000319'
TIPO_RELIABILITA_MQIAMO_RELIABILITY_TYPE	794	X'0000031A'
MQIAMO_LATE_JOIN_MARK	795	X'0000031B'
NACK_MQIAMO RCVD	796	X'0000031C'
MQIAMO_REPAIR_PKT	797	X'0000031D'
MQIAMO_HISTORY_PKT	798	X'0000031E'
MQIAMO_PENDING_PKTS	799	X'0000031F'
MQIAMO_PKT_RATE	800	X'00000320'
MQIAMO_MCAST_XMIT_RATE	801	X'00000321'
BATCH_TIME MQIAMO_MCAST_	802	X'00000322'
MQIAMO_MCAST_HEARTBEAT	803	X'00000323'
PORTA MQIAMO_DEST_DATA_PORT	804	X'00000324'
PORTA_RIPARAZIONE_DEST_MQIAMO_	805	X'00000325'
MQIAMO_ACKS_RCVD	806	X'00000326'
RICONOSCIMENTI MQIAMO_ACTIVE_ACKERS	807	X'00000327'
MQIAMO_PKTS_SENT	808	X'00000328'
MQIAMO_TOTAL_REPAIR_PKTS	809	X'00000329'
MQIAMO_TOTAL_PKTS_SENT	810	X'0000032A'
MQIAMO_TOTAL_MSGS_SENT	811	X'0000032B'
MQIAMO_TOTALE_BYTE_INVIATI	812	X'0000032C'
MQIAMO_NUM_FLUSSI	813	X'0000032D'
MQIAMO_ACK_FEEDBACK	814	X'0000032E'
MQIAMO_NACK_FEEDBACK	815	X'0000032F'
MQIAMO_PKTS_LOST	816	X'00000330'
MQIAMO_MSGS_RCVD	817	X'00000331'

Tabella 187. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQIAMO_MSG_BYTES_RCVD	818	X'00000332'
MSG_MQIAMO_CONSEGNATO	819	X'00000333'
MQIAMO_PKTS_PROCESSED	820	X'00000334'
MQIAMO_PKTS_DLVD	821	X'00000335'
MQIAMO_PKTS_DROP	822	X'00000336'
MQIAMO_PKTS_DUPLICATO	823	X'00000337'
MQIAMO_NACKS_CREATO	824	X'00000338'
MQIAMO_NACK_PKTS_SENT	825	X'00000339'
MQIAMO_REPAIR_PKTS_RQSTD	826	X'0000033A'
MQIAMO_REPAIR_PKTS_RCVD	827	X'0000033B'
MQIAMO_PKTS_REPAIRED	828	X'0000033C'
MQIAMO_TOTAL_MSGS_RCVD	829	X'0000033D'
MQIAMO_TOTAL_MSGS_BYTES_RCVD	830	X'0000033E'
MQIAMO_TOTAL_REPAIR_PKTS_RCVD	831	X'0000033F'
MQIAMO_TOTAL_REPAIR_PKTS_RQSTD	832	X'00000340'
MSG_TOTALE_MQIAMO_ELABORATI	833	X'00000341'
MQIAMO_TOTAL_MSGS_SELECTED	834	X'00000342'
MQIAMO_TOTAL_MSGS_EXPIRED	835	X'00000343'
MQIAMO_TOTAL_MSGS_CONSEGNATI	836	X'00000344'
MQIAMO_TOTAL_MSGS_RESTITUITI	837	X'00000345'
MQIAMO_LAST_UTENTE	837	X'00000345'

MQIAMO64_* (Formato del comando Tipi di parametro di monitoraggio numero intero a 64 bit)

Tabella 188. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIAMO64_AVG_Q_TIME	703	X'000002BF'
MQIAMO64_Q_TIME_AVG	741	X'000002E5'
MQIAMO64_Q_TIME_MAX	742	X'000002E6'
MQIAMO64_Q_TIME_MIN	743	X'000002E7'
MQIAMO64_BROWSE_BYTES	745	X'000002E9'
MQIAMO64_BYTES	746	X'000002EA'
MQIAMO64_GET_BYTES	747	X'000002EB'
MQIAMO64_PUT_BYTES	748	X'000002EC'
MQIAMO64_TOPIC_PUT_BYTES	783	X'0000030F'
MQIAMO64_PUBLISH_MSG_BYTES	785	X'00000311'

MQIASY_ * (Selettori di sistema interi)

Tabella 189. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIASY_FIRST	-1	X'FFFFFFFF'
ID_MQIASY_CODED_CHAR_SET_ID	-1	X'FFFFFFFF'
TIPO_MQIAS_	-2	X'FFFFFFFE'
COMANDO MQIAS_AND	-3	X'FFFFFFFD'
NUMERO MQIASY_MSG_SEQ_	-4	X'FFFFFFFC'
MQIASY_CONTROL	-5	X'FFFFFFFB'
CODICE_DI_MQIAS_SOCIETÀ	-6	X'FFFFFFFA'
MQIASY_REASON	-7	X'FFFFFFF9'
MQIASY_BAG_OPZIONI	-8	X'FFFFFFF8'
VERSIONE MQIASY_	-9	X'FFFFFFF7'
MQIASY_LAST_UTENTE	-9	X'FFFFFFF7'
MQIASY_LAST	-2000	X'FFFFF830'

MQIAUT_ * (Autenticatore intestazione informazioni IMS)

Tabella 190. Nomi e valori costanti	
Nome	**Valore**
MQIAUT_NONE	"¬¬¬¬¬¬¬¬"
MQIAUT_NON_ARRAY	'¬','¬','¬','¬','¬','¬','¬','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

MQIAV_ * (Valori attributo numero intero)

Tabella 191. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIAV_NOT_APPLICABLE	-1	X'FFFFFFFF'
MQIAV_NON DEFINITO	-2	X'FFFFFFFE'

MQICM_ * (Modalità di commit intestazione informazioni IMS)

Tabella 192. Nomi e valori costanti	
Nome	**Valore**
MQICM_COMMIT_THEN_SEND	'0'
MQICM_SEND_THEN_COMMIT	'1'

MQIDO_ * (Opzioni in dubbio formato comando)

Tabella 193. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
COMMIT MQIDO	1	X'00000001'
BACKOUT MQIDO	2	X'00000002'

MQIEP_ * (Punti di ingresso interfaccia)

Struttura dei parametri di sicurezza della connessione

Tabella 194. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQIEP	"IEP¬"
MATRICE MQIEP_STRUC_ID_ARRAY	'I','E','P','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 195. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIEP_VERSION_1	1	X'00000001'
VERSIONE MQDXP_CURRENT_	1	X'00000001'

MQIGQ_ * (In coda all'interno del gruppo)

Tabella 196. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIGQ_DISABLED	0	X'00000000'
MQIGQ_ENABLED	1	X'00000001'

MQIGQPA_ * (Autorizzazione immissione accodamento all'interno del gruppo)

Tabella 197. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIGQPA_PREDEFINITO	1	X'00000001'
CONTEXT MQIGQPA	2	X'00000002'
MQIGQPA_ONLY_IGQ	3	X'00000003'
MQIGQPA_ALTERNATE_OR_IGQ	4	X'00000004'

MQIIH_ * (Indicatori e struttura dell'intestazione delle informazioni IMS)

Struttura dell'intestazione delle informazioni IMS

Tabella 198. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQIIH	"IIH¬"
ARRAY ID_STRUC_MQIIH	'I','I','H','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 199. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIIH_VERSION_1	1	X'00000001'
VERSIONE MQIIH_CURRENT_	1	X'00000001'

Tabella 199. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQIIH_LENGTH_1	84	X'00000054'

Indicatori intestazione informazioni IMS

Tabella 200. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIIH_NONE	0	X'00000000'
MQIIH_PASS_ESPIRAZIONE	1	X'00000001'
MQIIH_UNLIMIT_EXPIRATION	0	X'00000000'
MQIIH_REPLY_FORMAT_NONE	8	X'00000008'
MQIIH_IGNORE_PURG	16	X'00000010'
MQIIH_CM0_REQUEST_RESPONSE	32	X'00000020'

MQIMPO_ * (Richiedi opzioni e struttura della proprietà del messaggio)

Interroga la struttura delle opzioni della proprietà del messaggio

Tabella 201. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQIMPO	"IMPO"
MATRICE MQIMPO_STRUC_ID_ARRAY	'I','M','P','O'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 202. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIMPO_VERSION_1	1	X'00000001'
VERSIONE MQIMPO_CURRENT_	1	X'00000001'

Opzioni di interrogazione proprietà messaggio

Tabella 203. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
TIPO_CONVER_MQIMPO	2	X'00000002'
QUERY MQIMPO_LENGTH	4	X'00000004'
MQIMPO_INQ_FIRST	0	X'00000000'
MQIMPO_INQ_NEXT	8	X'00000008'
MQIMPO_INQ_PROP_UNDER_CURSOR	16	X'00000010'
VALORE MQIMPO_CONVERT_	32	X'00000020'
MQIMPO_NONE	0	X'00000000'

MQINBD_ * (Formato del comando Disposizioni in entrata)

Tabella 204. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQINBD_Q_MGR	0	X'00000000'
Gruppo_MQIN	3	X'00000003'

MQIND_ * (Valori indice speciali)

Tabella 205. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIND_NONE	-1	X'FFFFFFFF'
MQIND_ALL	-2	X'FFFFFFFE'

MQIPADDR_ * (Versioni indirizzo IP)

Tabella 206. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIPADDR_IPV4	0	X'00000000'
MQIPADDR_IPV6	1	X'00000001'

MQISS_ * (Ambiti di sicurezza dell'intestazione di informazioni IMS)

Tabella 207. Nomi e valori costanti	
Nome	**Valore**
CHECK MQISS	'C'
MQISS_FULL	'F'

MQIT_ * (Tipi di indice)

Tabella 208. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQIT_NONE	0	X'00000000'
ID_MSG_MQIT	1	X'00000001'
ID CORREL_MQIT	2	X'00000002'
MQIT_MSG_TOKEN	4	X'00000004'
ID_GROUP_MQIT	5	X'00000005'

MQITEM_ * (Tipo di elemento per mqInquireItemInfo)

Tabella 209. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
NUMERO_NUMERO_INTERO	1	X'00000001'
MQITEM_STRING	2	X'00000002'
MQITEM_BAG	3	X'00000003'
MQITEM_BYTE_STRING	4	X'00000004'
FILTRO MQITEM_INTEGER_FILTER	5	X'00000005'

Tabella 209. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
FILTRO DI STRINGA MQITEM_	6	X'00000006'
MQITEM_INTEGER64	7	X'00000007'
FILTRO DI STRINGA MQITEM_BYTE	8	X'00000008'

MQITII_ * (Identificativo istanza transazione intestazione informazioni IMS)

Tabella 210. Nomi e valori costanti	
Nome	**Valore**
MQITII_NONE	X'00...00' (16 valori null)
MQITIA_NON_ARRAY	'\0','\0',... (16 valori null)

MQITS_ * (Stati transazione intestazione informazioni IMS)

Tabella 211. Nomi e valori costanti	
Nome	**Valore**
MQITS_IN_CONVERSAZIONE	'C'
MQITS_NON_IN_CONVERSAZIONE	'¬'
MQITS_ARCHITECTED	'A'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

MQKAI_ * (IntervalloKeepAlive)

Tabella 212. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
AUTO MQKAI	-1	X'FFFFFFFF'

MQMASTER_ * (Amministrazione principale)

Tabella 213. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQMASTER_NO	0	X'00000000'
MQMASTER_SÌ	1	X'00000001'

MQMCAS_ * (Formato del comando Stato agent canale messaggi)

Tabella 214. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQCAS_STOPPED	0	X'00000000'
MQMCAS_RUNNING	3	X'00000003'

MQMCAT_ * (Tipi MCA)

Tabella 215. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
PROCESSO MQMCAT	1	X'00000001'

Tabella 215. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQMCAT_THREAD	2	X'00000002'

MQMCD_ * (Informazioni tag opzioni di pubblicazione / sottoscrizione)

Opzioni di pubblicazione / sottoscrizione Tag Message Content Descriptor (mcd) Tag

Tabella 216. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
VERSIONE MQMCD_FOLDER_	1	X'00000001'

Nomi tag opzioni di pubblicazione / sottoscrizione

Tabella 217. Nomi e valori costanti	
Nome	**Valore**
DOMINIO_MSG_MQMCD	"Msd"
MQMCD_MSG_SET	"Set"
TIPO_MSG_MQMCD	"Type"
MQMCD_MSG_FORMAT	"Fmt"

Opzioni di pubblicazione / sottoscrizione Tag nomi tag XML

Tabella 218. Nomi e valori costanti	
Nome	**Valore**
DOMAIN_MQMCD_MSG_B	"<Msd>"
DOMINIO_MSG_MQMCD_E	"</Msd>"
MQMCD_MSG_SET_B	"<Set>"
MQMCD_MSG_SET_E	"</Set>"
MQMCD_MSG_TYPE_B	"<Type>"
MQMCD_MSG_TYPE_E	"</Type>"
MQMCD_MSG_FORMAT_B	"<Fmt>"
MQMCD_MSG_FORMAT_E	"</Fmt>"

Valori tag tag opzioni di pubblicazione / sottoscrizione

Tabella 219. Nomi e valori costanti	
Nome	**Valore**
MQMCD_DOMAIN_NONE	"none"
NUOVO_DOMINIO_MQMCD	"neon"
MQMCD_DOMAIN_MRM	"mrm"
MQMCD_DOMAIN_JMS_NONE	"jms_none"
MQMCD_DOMAIN_JMS_TEXT	"jms_text"
MQMCD_DOMAIN_JMS_OBJECT	"jms_object"
MQMCD_DOMAIN_JMS_MAP	"jms_map"

Tabella 219. Nomi e valori costanti (Continua)	
Nome	**Valore**
MQMCD_DOMAIN_JMS_STREAM	"jms_stream"
JMS_BYTES MQMCD_DOMAIN_	"jms_bytes"

MQMD_ * (Struttura descrittore messaggio)

Tabella 220. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQMD	"MD¬¬"
ID_MQMD_STRUC_ARRAY	'M','D','¬','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 221. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQMD_VERSION_1	1	X'00000001'
MQMD_VERSION_2	2	X'00000002'
VERSIONE MQMD_CURRENT_	2	X'00000002'

MQMDE_ * (Struttura estensione descrittore del messaggio)

Tabella 222. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQMDE	"MDE¬"
MATRICE MQMDE_STRUC_ID_ARRAY	'M','D','E','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 223. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQMDE_VERSION_2	2	X'00000002'
VERSIONE MQMDE_CURRENT_	2	X'00000002'
MQMDE_LENGTH_2	72	X'00000048'

MQMDEF_ * (Indicatori di estensione del descrittore del messaggio)

Tabella 224. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQMDEF_NONE	0	X'00000000'

MQMDS_ * (Sequenza di consegna messaggi)

Tabella 225. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
PRIORITà_MQMDS	0	X'00000000'
FIFO MQMDS	1	X'00000001'

MQMF_ * (Indicatori messaggio)

Tabella 226. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQMF_SEGMENTATION_INIBITO	0	X'00000000'
MQMF_SEGMENTAZIONE_CONSENTITA	1	X'00000001'
MQMF_MSG_IN_GROUP	8	X'00000008'
MQMF_LAST_MSG_IN_GROUP	16	X'00000010'
ISCRIZIONE MQMF_SE	2	X'00000002'
MQMF_LAST_SEGMENT	4	X'00000004'
MQMF_NONE	0	X'00000000'

MQMHBO_ * (handle del messaggio per le opzioni e la struttura del buffer)

Gestione dei messaggi per la struttura delle opzioni di buffer

Tabella 227. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQMHBO_	"MHBO"
MATRICE MQMHBO_STRUC_ID_ARRAY	'M','H','B','O'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 228. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQMHBO_VERSION_1	1	X'00000001'
VERSIONE MQMHBO_CURRENT_	1	X'00000001'

Gestione messaggi per opzioni buffer

Tabella 229. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQMHBO_PROPERTIES_IN_MQRFH2	1	X'00000001'
MQMHBO_DELETE_PROPERTIES	2	X'00000002'
MQMHBO_NONE	0	X'00000000'

MQMI_ * (Identificatore messaggio)

Tabella 230. Nomi e valori costanti	
Nome	**Valore**
MQMI_NONE	X'00...00' (24 valori null)
MQMI_NONE_ARRAY	'\0','\0',... (24 valori null)

MQMMBI_ * (Contrassegno messaggio - Intervallo di ricerca)

Tabella 231. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQMMBI_UNLIMITED	-1	X'FFFFFFFF'

MQMO_ * (Opzioni di corrispondenza)

Tabella 232. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
ID MQMO_MATCH_MSG_ID	1	X'00000001'
ID CORREL_MQMO_MATCH_	2	X'00000002'
ID_GROUP_MATCH_MQMO	4	X'00000004'
NUMERO SEQ MQMO_MATCH_MSG_	8	X'00000008'
MQMO_MATCH_OFFSET	16	X'00000010'
MQMO_MATCH_MSG_TOKEN	32	X'00000020'
MQMO_NONE	0	X'00000000'

MQMODE_ * (Opzioni modalità formato comando)

Tabella 233. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
FORCE_MQMODE	0	X'00000000'
MQMODE_QUIESCE	1	X'00000001'
MQMODE_TERMINATE	2	X'00000002'

MQMON_ * (Monitoraggio valori)

Tabella 234. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQMON_NON_DISPONIBILE	-1	X'FFFFFFFF'
MQMON_NONE	-1	X'FFFFFFFF'
MGR MQMON_Q	-3	X'FFFFFFFD'
MQMON_DISATTIVO	0	X'00000000'
MMON_UN	1	X'00000001'
DISABILITAZIONE_MQMON_	0	X'00000000'
MMON_ENABLED	1	X'00000001'
MMON_LOW	17	X'00000011'
MQMON_MEDIO	33	X'00000021'
MQMON_HIGH	65	X'00000041'

MQMT_ * (Tipi di messaggio)

Tabella 235. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQMT_SYSTEM_FIRST	1	X'00000001'
MQMT_REQUEST	1	X'00000001'
MQMT_REPLY	2	X'00000002'
MQMT_DATAGRAM	8	X'00000008'
REPORT MQMT	4	X'00000004'
MQMT_MQ_FIELDS_FROM_MQE	112	X'00000070'

Tabella 235. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQMT_MQ_FIELDS	113	X'00000071'
SYSTEM_MQMT_LAST	65535	X'0000FFFF'
MQMT_APPL_FIRST	65536	X'00010000'
APPL_MQMT_LAST	999999999	X'3B9AC9FF'

MQMTOK_ * (Token messaggio)

Tabella 236. Nomi e valori costanti	
Nome	**Valore**
MQMTOK_NONE	X'00...00' (16 valori null)
MQMTOK_NON_ARRAY	'\0','\0',... (16 valori null)

MQNC_ * (Conteggio nomi)

Tabella 237. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQNC_MAX_NAMELIST_NAME_COUNT	256	X'00000100'

MQNPM_ * (Classe messaggio non persistente)

Tabella 238. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQNPM_CLASS_NORMAL	0	X'00000000'
MQNPM_CLASS_HIGH	10	X'0000000A'

MQNPMS_ * (NonPersistent- Velocità messaggi)

Tabella 239. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQNPMS_NORMAL	1	X'00000001'
MQNPMS_FAST	2	X'00000002'

MQNT_ * (Tipi elenco nomi)

Tabella 240. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQNT_NONE	0	X'00000000'
MQNT_Q	1	X'00000001'
MQNT_CLUSTER	2	X'00000002'
INFO AUTORE MQNT	4	X'00000004'
MQNT_ALL	1001	X'000003E9'

MQNVS_ * (Nomi per stringa nome / valore)

Tabella 241. Nomi e valori costanti	
Nome	**Valore**
TIPO_APPL_MQNV	"OPT_APP_GRP¬"
TIPO_MSG_MQNVS	"OPT_MSG_TYPE¬"

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

MQOA_ * (Limiti per selettori per attributi oggetto)

Tabella 242. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQOA_PRIMO	1	X'00000001'
MQOA_LAST	9000	X'00002328'

MQOD_ * (Struttura descrittore oggetto)

Tabella 243. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQOD	"OD¬¬"
MATRICE MQOD_STRUC_ID_ARRAY	'O','D','¬','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 244. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQOD_VERSION_1	1	X'00000001'
MQOD_VERSION_2	2	X'00000002'
MQOD_VERSION_3	3	X'00000003'
MQOD_VERSION_4	4	X'00000004'
VERSIONE MQOD_CURRENT_	4	X'00000004'
MQOD_XX_ENCODE_CASE_ONE lo_corrente	(value differs by platform or version)	(value differs by platform or version)

MQOII_* (Identificativo istanza oggetto)

Tabella 245. Nomi e valori costanti	
Nome	**Valore**
MQOII_NONE	X'00...00' (24 valori null)
MQOII_NONE_ARRAY	'\0','\0',... (24 valori null)

MQOL_ * (Lunghezza originale)

Tabella 246. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQOL_NON DEFINITO	-1	X'FFFFFFFF'

MQOM_ * (Opzioni di messaggi Db2 obsoleti sul gruppo di interrogazione)

Tabella 247. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQOM_NO	0	X'00000000'
SÌ MQOM	1	X'00000001'

MQOO_ * (Opzioni di apertura)

Tabella 248. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQOO_BIND_AS_Q_DEF	0	X'00000000'
MQOO_READ_AHEAD_AS_Q_DEF	0	X'00000000'
MQOO_INPUT_AS_Q_DEF	1	X'00000001'
MQOO_INPUT_SHARED	2	X'00000002'
MQOO_INPUT_EXCLUSIVE	4	X'00000004'
MQOO_SFOGLIA	8	X'00000008'
OUTPUT MQOO	16	X'00000010'
MQOO_INQUIRE	32	X'00000020'
SET MQOO	64	X'00000040'
MQOO_SAVE_ALL_CONTEXT	128	X'00000080'
MQOO_PASS_IDENTITY_CONTEXT	256	X'00000100'
MQOO_PASS_ALL_CONTEXT	512	X'00000200'
MQOO_SET_IDENTITY_CONTEXT	1024	X'00000400'
MQOO_SET_ALL_CONTEXT	2048	X'00000800'
MQOO_ALTERNATE_USER_AUTHORITY	4096	X'00001000'
MQOO_FAIL_IF_QUIESCING	8192	X'00002000'
MQOO_BIND_ON_OPEN	16384	X'00004000'
MQOO_BIND_NON_FISSO	32768	X'00008000'
CO_MQOO	131072	X'00020000'
MQOO_RESOLVE_LOCAL_TOPIC	262144	X'00040000'
MQOO_NO_READ_AHEAD	524288	X'00080000'
MQOO_READ_AHEAD	1048576	X'00100000'
Gruppo_BIND_MQOO	4194304	X'00400000'

*MQOO_ * (Di seguito utilizzato solo in C + +)*

Tabella 249. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQOO_RESOLVE_NAMES	65536	X'00010000'
MQOO_RESOLVE_LOCAL_Q	262144	X'00040000'

MQOP_ * (Codici di funzionamento per MQCTL e MQCB)

Codici di operazione per MQCTL

Tabella 250. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
INIZIO_MQOP	1	X'00000001'
MQOP_START_WAIT	2	X'00000002'
MQOP_STOP	4	X'00000004'

Codici di operazione per MQCB

Tabella 251. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
REGISTER MQOP	256	X'00000100'
MQOP_DEREGISTER	512	X'00000200'

Codici di operazione per MQCTL e MQCB

Tabella 252. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQOP_SOSPENSIONE	65536	X'00010000'
MQOP_RESUME	131072	X'00020000'

MQOPEN_ * (Valori correlati alla struttura MQOPEN_PRIV)

Tabella 253. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQOPEN_PRIV_VERSION_1	1	X'00000001'
VERSIONE MQOPEN_PRIV_CURRENT_	1	X'00000001'

MQOPER_ * (Operazioni attività)

Tabella 254. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQOPER_SYSTEM_FIRST	0	X'00000000'
MQOPER_SCONOSCIUTO	0	X'00000000'
MQOPER_BROWSE	1	X'00000001'
DISCARD MQOPERAZIONE	2	X'00000002'
GET MQOPERAZIONE	3	X'00000003'
MQOPERA_PUT	4	X'00000004'
MQOPER_PUT_REPLY	5	X'00000005'
REPORT MQOPER_PUT_REPORT	6	X'00000006'
MQOPER_RECEIVE	7	X'00000007'
MQOPER_INVIA	8	X'00000008'
MQOPER_TRANSFORM	9	X'00000009'

Tabella 254. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQOPER_PUBBLICA	10	X'0000000A'
MQOPER_EXCLUDED_PUBLISH	11	X'0000000B'
MQOPER_DISCARDED_PUBLISH	12	X'0000000C'
LAST SYSTEM_MQOPER	65535	X'0000FFFF'
MQOPER_APPL_PRIMO	65536	X'00010000'
LAST APPL_MQOPER	999999999	X'3B9AC9FF'

MQOT_ * (Tipi di oggetti e tipi di oggetti estesi)

Tipi di oggetto

Tabella 255. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQOT_NONE	0	X'00000000'
MQOT_Q	1	X'00000001'
ELENCO NOMI MQOTT	2	X'00000002'
PROCESSO MQOT_	3	X'00000003'
MQOT_STORAGE_CLASSE	4	X'00000004'
Gestore code MQOT_GR	5	X'00000005'
CANALIZZATA MQOT_	6	X'00000006'
INFO MQOT_AUTH_O	7	X'00000007'
TOPIC MQOT_T	8	X'00000008'
MQOT_CF_STRUC	10	X'0000000A'
LISTENER MQOT_	11	X'0000000B'
SERVIZIO_MQT	12	X'0000000C'
MQOT_RESERVED_1	999	X'000003E7'

Tipi di oggetto estesi

Tabella 256. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQOT_ALL	1001	X'000003E9'
MQOT_ALIAS_Q	1002	X'000003EA'
MQOT_MODEL_Q	1003	X'000003EB'
Coda MQOT_LOCAL_Q	1004	X'000003EC'
MQOT_REMOTE_Q	1005	X'000003ED'
MQOT_SENDER_CHANNEL	1007	X'000003EF'
CANALE_SERVER_MQOT_	1008	X'000003F0'
MQOT_REQUESTER_XX_ENCODE_CASE_ONE canalizzata	1009	X'000003F1'
CANALE DI RICETTE MQOT_T	1010	X'000003F2'
MQOT_XX_ENCODE_CASE_ONE canale_corrente	1011	X'000003F3'
MQOT_SAVED_CHALLEGATO	1012	X'000003F4'

Tabella 256. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQOT_SVRCONN_CHALLEGATO	1013	X'000003F5'
MQOT_CLNTCONN_CHALLEGATO	1014	X'000003F6'
MQOT_CANALE_BREVE	1015	X'000003F7'
CHLAUTH MQOT_	1016	X'000003F8'
MQOT_REMOTE_Q_MGR_NAME	1017	X'000003F9'
PROT_POLICY MQOT_	1019	X'000003FB'
CANALIZZATA MQOT_TT_T	1020	X'000003FC'
MQOT_AMQP_XX_ENCODE_CASE_ONE canale	1021	X'000003FD'
REC AUTORE MQOT_	1022	X'000003FE'

MQPA_ * (Autorizzazione Put)

Tabella 257. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPA_PREDEFINITO	1	X'00000001'
CONTEXT MQPA	2	X'00000002'
MQPA_ONLY_MCA	3	X'00000003'
MQPA_ALTERNATE_OR_MCA	4	X'00000004'

MQPD_ * (Descrittore proprietà, supporto e contesto)

Struttura descrittore proprietà

Tabella 258. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQPD	"PD¬¬"
ID_MQPD_STRUC_ARRAY	'P','D','¬','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 259. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPD_VERSION_1	1	X'00000001'
VERSIONE MQPD_CURRENT_	1	X'00000001'

Opzioni descrittore proprietà

Tabella 260. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPD_NONE	0	X'00000000'

Opzioni di supporto proprietà

Tabella 261. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPD_SUPPORT_OPZIONALE	1	X'00000001'
MQPD_SUPPORT_XX_ENCODE_CASE_ONE obbligatorio	1048576	X'00100000'
MQPD_SUPPORT_REQUIRED_IF_LOCAL	1024	X'00000400'
MQPD_REJECT_UNSUP_MASK	-1048576	X'FFF00000'
MQPD_ACCEPT_UNSUP_IF_XMIT_MASK	1047552	X'000FFC00'
MQPD_ACCEPT_UNSUP_MASK	1023	X'000003FF'

Contesto proprietà

Tabella 262. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPD_NO_CONTEXT	0	X'00000000'
CONTEXT MQPD_USER_	1	X'00000001'

MQPER_ * (Valori persistenza)

Tabella 263. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPER_PERSISTENCE_AS_PARENT	-1	X'FFFFFFFF'
MQPER_NOT_PERSISTENT	0	X'00000000'
PERSISTORA_MQPER_	1	X'00000001'
MQPER_PERSISTENCE_AS_Q_DEF	2	X'00000002'
MQPER_PERSISTENCE_AS_TOPIC_DEF	2	X'00000002'

MQPL_ * (Piattaforme)

Tabella 264. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MVS MQPL	1	X'00000001'
MQPL_OS390	1	X'00000001'
ZOS MQPL	1	X'00000001'
MQPL_OS2	2	X'00000002'
AIX MQPL	3	X'00000003'
MQPL_UNIX	3	X'00000003'
MQPL_OS400	4	X'00000004'
WINDOWS MQPL	5	X'00000005'
MQPL_WINDOWS_NT	11	X'0000000B'
VMS MQPL	12	X'0000000C'
NSK MQPL	13	X'0000000D'
MQPL_OPEN_TP1	15	X'0000000F'
VM MQPL	18	X'00000012'

Tabella 264. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
TPF MQPL	23	X'00000017'
VSE MQPL	27	X'0000001B'
APPLICAZIONE_MQPL	28	X'0000001C'
MQPL_NATIVE	1	X'00000001'

MQPMO_ * (Inserire le opzioni del messaggio e la struttura per la maschera di pubblicazione)

Struttura delle opzioni del messaggio di inserimento

Tabella 265. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQPMO	"PMO¬"
MATRICE MQPMO_STRUC_ID_ARRAY	'P','M','O','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 266. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPMO_VERSION_1	1	X'00000001'
MQPMO_VERSION_2	2	X'00000002'
MQPMO_VERSION_3	3	X'00000003'
VERSIONE MQPMO_CURRENT_	3	X'00000003'
MQPMO_CURRENT_LENGTH	(value differs by platform or version)	(value differs by platform or version)

Opzioni di inserimento messaggio

Tabella 267. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPMO_SYNCPOINT	2	X'00000002'
MQPMO_NO_SYNCPOINT	4	X'00000004'
MQPMO_DEFAULT_CONTEXT	32	X'00000020'
ID_MQPMO_NEW_MSG_	64	X'00000040'
ID_CORREL_NEW_MQPMO_	128	X'00000080'
MQPMO_PASS_IDENTITY_CONTEXT	256	X'00000100'
MQPMO_PASS_ALL_CONTEXT	512	X'00000200'
MQPMO_SET_IDENTITY_CONTEXT	1024	X'00000400'
MQPMO_SET_ALL_CONTEXT	2048	X'00000800'
MQPMO_ALTERNATE_USER_AUTHORITY	4096	X'00001000'
MQPMO_FAIL_IF_QUIESCING	8192	X'00002000'
MQPMO_NO_CONTEXT	16384	X'00004000'
ORDER MQPMO_LOGICAL_	32768	X'00008000'

Tabella 267. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQPMO_ASYNC_RESPONSE	65536	X'00010000'
MQPMO_SYNC_RESPONSE	131072	X'00020000'
MQPMO_RESOLVE_LOCAL_Q	262144	X'00040000'
MQPMO_RETAIN	2097152	X'00200000'
MQPMO_MD_FOR_OUTPUT_ONLY	8388608	X'00800000'
MQPMO_SCOPE_QMGR	67108864	X'04000000'
MQPMO_SUPPRESS_REPLYTO	134217728	X'08000000'
MQPMO_NOT_OWN_SUBS	268435456	X'10000000'
MQPMO_RESPONSE_AS_Q_DEF	0	X'00000000'
MQPMO_RESPONSE_AS_TOPIC_DEF	0	X'00000000'
MQPMO_NONE	0	X'00000000'

Opzioni di inserimento messaggio per la maschera di pubblicazione

Tabella 268. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPMO_PUB_OPTIONS_MASK	2097152	X'00200000'

MQPMRF_ * (Inserisci campi record messaggio)

Tabella 269. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
ID_MSG_MQPMRF	1	X'00000001'
ID CORREL_MQPMRF	2	X'00000002'
ID_GROUP_MQPMRF	4	X'00000004'
MQPMRF_FEEDBACK	8	X'00000008'
MQPMRF_ACCOUNTING_TOKEN	16	X'00000010'
MQPMRF_NONE	0	X'00000000'

MQPO_ * (Formato del comando Opzioni di eliminazione)

Tabella 270. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPO_SÌ	1	X'00000001'
MQPO_NO	0	X'00000000'

MQPRI_ * (Priorità)

Tabella 271. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPRI_PRIORITY_AS_Q_DEF	-1	X'FFFFFFFF'
MQPRI_PRIORITY_AS_PARENT	-2	X'FFFFFFFE'
MQPRI_PRIORITY_AS_PUBLISHED	-3	X'FFFFFFFD'

Tabella 271. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQPRI_PRIORITY_AS_TOPIC_DEF	-1	X'FFFFFFFF'

MQPROP_ * (Valori di controllo proprietà coda e canale e lunghezza massima proprietà)

Valori di controllo proprietà coda e canale

Tabella 272. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
COMPATIBILITÀ_MQPROP_	0	X'00000000'
MQPROP_NONE	1	X'00000001'
TUTTE le MQPROP	2	X'00000002'
MQPROP_FORCE_MQRFH2	3	X'00000003'

Lunghezza massima proprietà

Tabella 273. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPROP_UNRESTRICTED_LENGTH	-1	X'FFFFFFFF'

MQPRT_ * (Inserisci valori risposta)

Tabella 274. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPRT_RESPONSE_AS_PARENT	0	X'00000000'
RISPOSTA MQPRT_SYNC_RESPONSE	1	X'00000001'
MQPRT_ASYNC_RESPONSE	2	X'00000002'

MQPS_ * (Pubblicazione / Sottoscrizione)

Formato comando Stato pubblicazione / sottoscrizione

Tabella 275. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPS_STATUS_INACTIVE	0	X'00000000'
MQPS_STATUS_STARTING	1	X'00000001'
MQPS_STATUS_STOPPING	2	X'00000002'
MQPS_STATUS_ATTIVO	3	X'00000003'
COMPAT_STATO_MQPS	4	X'00000004'
ERRORE MQPS_STATUS	5	X'00000005'
STATO_MQPS_RIFIUTATO	6	X'00000006'

Pubblica / sottoscrivi tag come stringhe

COMANDO MQPS	"MQPSCommand"

CODICE_COMP_MQPS	"MQPSCompCode"
ID_CORREL_MQPS	"MQPSCorrelId"
MQPS_DELETE_OPTIONS	"MQPSDelOpts"
ID_ERRORE_MQPS	"MQPSErrorId"
POS ERRORE MQPS	"MQPSErrorPos"
DATI_IN_MQPS	"MQPSIntData"
ID_PARAMETER_MQPS	"MQPSParmId"
OPZIONI MQPS_PUBLICATION_OPTIONS	"MQPSPubOpts"
DATAORA_PUBBLICAZIONE_MQPS	"MQPSPubTime"
MQPS_Q_MGR_NAME	"MQPSQMgrName"
MQPS_Q_NAME	"MQPSQName"
REASON_MQPS	"MQPSReason"
TESTO_MOTIVO_MQPS	"MQPSReasonText"
OPZIONI MQPS_REGISTRATION_	"MQPSRegOpts"
NUMERO_SEQUENZA_MQPS	"MQPSSeqNum"
MQPS_NOME_FLUSSO	"MQPSStreamName"
DATI DI STRINGA MQPS	"MQPSStringData"
IDENTITÀ_SOTTOSCRIZIONE_MQPS	"MQPSSubIdentity"
MQPS_NOME_SOTTOSCRIZIONE	"MQPSSubName"
MQPS_SUBSCRIPTION_USER_DATA	"MQPSSubUserData"
TOPIC MQPS	"MQPSTopic"
ID_USER_MQPS	"MQPSUserId"

Pubblica / Sottoscrivi tag come stringhe vuote

MQPS_COMMAND_B	"¬MQPSCommand¬"
MQPS_COMP_CODE_B	"¬MQPSCompCode¬"
ID_CORREL_MQPS_B	"¬MQPSCorrelId¬"
MQPS_DELETE_OPTIONS_B	"¬MQPSDelOpts¬"
ID_ERRORE_MQPS_B	"¬MQPSErrorId¬"
MQPS_ERRORE_POS_B	"¬MQPSErrorPos¬"
MQPS_INTEGER_DATA_B	"¬MQPSIntData¬"
ID_PARAMETER_MQPS_B	"¬MQPSParmId¬"
MQPS_PUBLICATION_OPTIONS_B	"¬MQPSPubOpts¬"
MQPS_PUBLISH_TIMESTAMP_B	"¬MQPSPubTime¬"
MQPS_Q_MGR_NAME_B	"¬MQPSQMgrName¬"
MQPS_Q_NAME_B	"¬MQPSQName¬"

MQPS_REASON_B	"¬MQPSReason¬"
MQPS_REASON_TEXT_B	"¬MQPSReasonText¬"
MQPS_REGISTRATION_OPTIONS_B	"¬MQPSRegOpts¬"
MQPS_SEQUENCE_NUMBER_B	"¬MQPSSeqNum¬"
MQPS_STREAM_NAME_B	"¬MQPSStreamName¬"
MQPS_STRING_DATA_B	"¬MQPSStringData¬"
MQPS_SOTTOSCRIZIONE_IDENTITÀ_B	"¬MQPSSubIdentity¬"
NOME_SOTTOSCRIZIONE_MQPS_B	"¬MQPSSubName¬"
MQPS_SUBSCRIPTION_USER_DATA_B	"¬MQPSSubUserData¬"
TOPIC_MQPS B	"¬MQPSTopic¬"
ID_USER_MQPS_B	"¬MQPSUserId¬"

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Pubblica / sottoscrivi valori tag comando come stringhe

MQPS_ELIMINA_PUBBLICAZIONE	"DeletePub"
MQPS_DEREGISTER_PUBLISHER	"DeregPub"
MQPS_DEREGISTER_SUBSCRIBER	"DeregSub"
MQPS_PUBBLICA	"Publish"
MQPS_REGISTER_PUBLISHER	"RegPub"
SOTTOSCRIZIONE_REGISTER_MQPS	"RegSub"
AGGIORNA_RICHIESTA_MQPS	"ReqUpdate"

Pubblica / sottoscrivi valori tag comando come stringhe vuote

MQPS_DELETE_PUBLICATION_B	"¬DeletePub¬"
MQPS_DEREGISTER_PUBLISHER_B	"¬DeregPub¬"
MQPS_DEREGISTER_SUBSCRIER_B	"¬DeregSub¬"
MQPS_PUBBLICA_B	"¬Publish¬"
MQPS_REGISTER_PUBLISHER_B	"¬RegPub¬"
MQPS_REGISTER_SUBSCRIER_B	"¬RegSub¬"
MQPS_REQUEST_UPDATE_B	"¬ReqUpdate¬"

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Opzioni di pubblicazione / sottoscrizione dei valori tag come stringhe

NOME_INDIRIZZO_MQPS	"AddName"
MQPS_ANONIMO	"Anon"
MQPS_CORREL_ID_AS_IDENTITY	"CorrelAsId"

MQPS_DEREGISTER_ALL	"DeregAll"
MQPS_DIRECT_REQUESTS	"DirectReq"
MQPS_DUPLICATES_OK	"DupsOK"
MQPS_FULL_XX_ENCODE_CASE_ONE risposta	"FullResp"
MQPS_INCLUDE_STREAM_NAME	"InclStreamName"
MQPS_INFORM_IF_CONSERVATI	"InformIfRet"
MQPS_IS_RETAINED_PUBLICAZIONE	"IsRetainedPub"
MQPS_JOIN_EXCLUSIVE	"JoinExcl"
INDIRIZZO_JOIN MQPS	"JoinShared"
SOLO MQPS_LEAVE_	"LeaveOnly"
LOCALE MQPS	"Local"
LOCKED MQPS	"Locked"
MQPS_NUOVA_PUBBLICAZIONI_SOLO	"NewPubsOnly"
MQPS_NO_ALTERAZIONE	"NoAlter"
MQPS_NO_REGISTRATION	"NoReg"
MQPS_NON_PERSISTENTE	"NonPers"
MQPS_NONE	"None"
MQPS_OTHER_SUBSCRIERS_ONLY	"OtherSubsOnly"
MQPS_PERSISTENTE	"Pers"
MQPS_PERSISTENT_AS_PUBLISH	"PersAsPub"
MQPS_PERSIST_AS_Q	"PersAsQueue"
MQPS_PUBLISH_ON_REQUEST_ONLY	"PubOnReqOnly"
MQPS_RETAIN_PUBBLICAZIONE	"RetainPub"
ID_UTENTE_VARIABILE_MQPS	"VariableUserId"

Pubblica / sottoscrivi opzioni Opzioni Tag Valori come stringhe vuote

MQPS_ADD_NAME_B	"¬AddName¬"
MQPS_ANONIMO_B	"¬Anon¬"
ID_CORREL_MQPS_AS_IDENTITY_B	"¬CorrelAsId¬"
MQPS_DEREGISTER_ALL_B	"¬DeregAll¬"
MQPS_DIRECT_REQUESTS_B	"¬DirectReq¬"
MQPS_DUPLICATES_OK_B	"¬DupsOK¬"
MQPS_FULL_XX_ENCODE_CASE_ONE rispo_b	"¬FullResp¬"
MQPS_INCLUDE_NOME_FLUSSO_B	"¬InclStreamName¬"
MQPS_INFORM_IF_RETAINED_B	"¬InformIfRet¬"
MQPS_IS_RETAINED_PUBLICATION_B	"¬IsRetainedPub¬"

MQPS_JOIN_EXCLUSIVE_B	"¬JoinExcl¬"
MQPS_JOIN_SHARED_B	"¬JoinShared¬"
MQPS_LEAVE_ONLY_B	"¬LeaveOnly¬"
BLOC_MQPS	"¬Local¬"
MQPS_LOCKED_B	"¬Locked¬"
MQPS_NEW_PUBLICATIONS_ONLY_B	"¬NewPubsOnly¬"
MQPS_NO_ALTERAZIONE B	"¬NoAlter¬"
MQPS_NO_REGISTRATION_B	"¬NoReg¬"
MQPS_NON_PERSIST_B	"¬NonPers¬"
MQPS_NONE_B	"¬None¬"
MQPS_OTHER_SUBSCRIBERS_ONLY_B	"¬OtherSubsOnly¬"
MQPS_PERSIST_B	"¬Pers¬"
MQPS_PERSISTENT_AS_PUBBLICA_B	"¬PersAsPub¬"
MQPS_PERSISTENT_AS_Q_B	"¬PersAsQueue¬"
MQPS_PUBLISH_ON_REQUEST_ONLY_B	"¬PubOnReqOnly¬"
MQPS_RETAIN_PUBLICATION_B	"¬RetainPub¬"
MQPS_VARIABLE_USER_ID_B	"¬VariableUserId¬"

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

MQPSC_ * (Cartella comandi di pubblicazione / sottoscrizione tag di opzioni di pubblicazione / sottoscrizione (psc) Tag)

Tabella 276. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
VERSIONE MQPSC_FOLDER_	1	X'00000001'

*MQPSC_ * (Nome tag tag opzioni di pubblicazione / sottoscrizione)*

COMANDO MQPSC	"Command"
MQPSC_REGISTRATION_OPTION	"RegOpt"
OPZIONE MQPSC_PUBLICATION_OPTION	"PubOpt"
MQPSC_DELETE_OPTION	"DelOpt"
TOPIC MQPSC	"Topic"
PUNTO_SOTTOSCRIZIONE_MQPSC_	"SubPoint"
FILTRO MQPSC	"Filter"
MQPSC_Q_MGR_NAME	"QMgrName"
MQPSC_Q_NAME	"QName"
DATAORA_PUBBLICAZIONE_MQP	"PubTime"
NUMERO_SEQUENZA_MQPSC_	"SeqNum"

MQPSC_SUBSCRIPTION_NAME	"SubName"
IDENTITÀ_SOTTOSCRIZIONE_MQPSC_	"SubIdentity"
DATI_UTENTE_SOTTOSCRIZIONE_MQPSC_	"SubUserData"
ID CORREL_MQPSC	"CorrelId"

*MQPSC_ * (Nome tag XML tag opzioni di pubblicazione / sottoscrizione)*

COMMAND_MQPSC_B	"<Command>"
MQPSC_COMMAND_E	"</Command>"
MQPSC_REGISTRATION_OPTION_B	"<RegOpt>"
MQPSC_REGISTRATION_OPTION_E	"</RegOpt>"
MQPSC_PUBLICATION_OPTION_B	"<PubOpt>"
MQPSC_PUBLICATION_OPTION_E	"</PubOpt>"
MQPSC_DELETE_OPTION_B	"<DelOpt>"
MQPSC_DELETE_OPTION_E	"</DelOpt>"
TOPIC_MQPSC_B	"<Topic>"
TOPIC_MQPSC_E	"</Topic>"
PUNTO_SOTTOSCRIZIONE_MQPSC_B	"<SubPoint>"
MQPSC_SUBSCRIPTION_POINT_E	"</SubPoint>"
FILTER_MQPSC_B	"<Filter>"
FILTER_MQPSC_E	"</Filter>"
MQPSC_Q_MGR_NAME_B	"<QMgrName>"
MQPSC_Q_MGR_NAME_E	"</QMgrName>"
MQPSC_Q_NAME_B	"<QName>"
MQPSC_Q_NAME_E	"</QName>"
MQPSC_PUBLISH_TIMESTAMP_B	"<PubTime>"
MQPSC_PUBLISH_TIMESTAMP_E	"</PubTime>"
MQPSC_SEQUENCE_NUMBER_B	"<SeqNum>"
MQPSC_SEQUENCE_NUMBER_E	"</SeqNum>"
NOME_SOTTOSCRIZIONE_MQPSC_B	"<SubName>"
NOME_SOTTOSCRIZIONE_MQPSC_E	"</SubName>"
IDENTITÀ_SOTTOSCRIZIONE_MQPSC_B	"<SubIdentity>"
IDENTITÀ_SOTTOSCRIZIONE_MQPSC_E	"</SubIdentity>"
MQPSC_SUBSCRIPTION_USER_DATA_B	"<SubUserData>"
MQPSC_SUBSCRIPTION_DATA_UTENTE	"</SubUserData>"
ID_CORREL_MQPSC_B	"<CorrelId>"
ID_CORREL_MQPSC_E	"</CorrelId>"

*MQPSC_ * (Opzioni di pubblicazione / sottoscrizione Tag Valori publisher come stringhe)*

PUBBLICAZIONE_ELIMINAZIONE_MQPSC_	"DeletePub"
MQPSC_DEREGISTER_SUBSCRIBER	"DeregSub"
MQPSC_PUBBLICA	"Publish"
SOTTOSCRITTORE_REGISTRA_MQPSC_SUBSCRIBER	"RegSub"
AGGIORNAMENTO MQPSC_REQUEST_UPDATE	"ReqUpdate"

*MQPSC_ * (Opzioni di pubblicazione / sottoscrizione Valori nome tag come stringhe)*

NOME_INDIRIZZO_MQPSC	"AddName"
MQPSC_CORREL_ID_AS_IDENTITÀ	"CorrelAsId"
TUTTE LE MQPSC_DEREGISTER_	"DeregAll"
MQPSC_DUPLICATES_OK	"DupsOK"
MQPSC_FULL_XX_ENCODE_CASE_ONE risposta	"FullResp"
MQPSC_INFOR_SE_CONSERVATA	"InformIfRet"
MQPSC_IS_RETAINED_PUB	"IsRetainedPub"
MQPSC_JOIN_SHARED	"JoinShared"
MQPSC_JOIN_ESCLUSIVO	"JoinExcl"
SOLO MQPSC_LEAVE_	"LeaveOnly"
LOCALE MQPSC	"Local"
LOCKED MQPSC	"Locked"
SOLO MQPSC_NEW_PUBS_	"NewPubsOnly"
MQPSC_NO_ALTERAZIONE	"NoAlter"
MQPSC_NON_PERSISTENT	"NonPers"
MQPSC_OTHER_SUBS_SOLO	"OtherSubsOnly"
MQPSC_PERSISTENTE	"Pers"
MQPSC_PERSISTENT_AS_PUBLISH	"PersAsPub"
MQPSC_PERSIST_AS_Q	"PersAsQueue"
MQPSC_NONE	"None"
MQPSC_PUB_ON_REQUEST_ONLY	"PubOnReqOnly"
MQPSC_RETAIN_PUB	"RetainPub"
ID_UTENTE_VARIABILE_MQPSC_	"VariableUserId"

MQPSCR_ * (Opzioni di pubblicazione / sottoscrizione)

Tag delle opzioni di pubblicazione / sottoscrizione Cartella di risposta di pubblicazione / sottoscrizione (pscr) Tag

Tabella 277. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
VERSIONE MQPSCR_FOLDER_	1	X'00000001'

Nomi tag opzioni di pubblicazione / sottoscrizione

MQPSCR_COMPLEZIONE	"Completion"
MQPSCR_XX_ENCODE_CASE_ONE risposta	"Response"
MQPSCR_REASON	"Reason"

Opzioni di pubblicazione / sottoscrizione Tag nomi tag XML

MQPSCR_COMPLETION_B	"<Completion>"
MQPSCR_COMPLETION_E	"</Completion>"
RISPOSTA MQPSCR_B	"<Response>"
RISPOSTA MQPSCR_E	"</Response>"
MQPSCR_REASON_B	"<Reason>"
MQPSCR_REASON_E	"</Reason>"

Valori tag tag opzioni di pubblicazione / sottoscrizione

MQPSCR_OK	"ok"
AVVERTENZA MQPSCR_WARNING	"warning"
ERRORE MQPSCR	"error"

MQPSM_ * (modalità Pub / Sub)

Tabella 278. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
DISABILITAZIONE_MQPSM	0	X'00000000'
COMPAT_MQPSM	1	X'00000001'
MQPSM_ENABLED	2	X'00000002'

MQPSPROP_ * (Proprietà messaggio Pub / Sot)

Tabella 279. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPSPROP_NONE	0	X'00000000'
COMPAT_MQPSPROP	1	X'00000001'
MQPSPROP_RFH2	2	X'00000002'
MQPSPROP_MSGPROP	3	X'00000003'

MQPSST_ * (Tipo stato pubblicazione / sottoscrizione formato comando)

Tabella 280. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
TUTTE le MQPSST	0	X'00000000'
LOCALE MQPSST	1	X'00000001'
PARENTE_MQPSST	2	X'00000002'
CHILD_MQPSST	3	X'00000003'

MQPUBO_ * (Opzioni di pubblicazione / sottoscrizione)

Tabella 281. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPUBO_NONE	0	X'00000000'
MQPUBO_CORREL_ID_AS_IDENTITY	1	X'00000001'
PUBBLICAZIONE_RETAIN_MQPUBO_PUBBLICAZIONE	2	X'00000002'
MQPUBO_OTHER_SUBSCRIERS_ONLY	4	X'00000004'
MQPUBO_NO_REGISTRAZIONE	8	X'00000008'
MQPUBO_IS_RETAINED_PUBLICATION	16	X'00000010'

MQPXP_ * (Struttura parametro uscita di instradamento di pubblicazione / sottoscrizione)

Tabella 282. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQPXP	"PXP¬"
ARRAY MQPXP_STRUC_ID_ARRAY	'P','X','P','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 283. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQPXP_VERSION_1	1	X'00000001'
VERSIONE MQPXP_CURRENT_	1	X'00000001'

MQQA_ * (Attributi coda)

Impedisci acquisizione valori

Tabella 284. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQQA_GET_INIBITO	1	X'00000001'
MQQA_GET_ALLOWED	0	X'00000000'

Impedisci valori di inserimento

Tabella 285. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQQA_PUT_INIBITO	1	X'00000001'
MQQA_PUT_CONSENTITO	0	X'00000000'

Condivisione coda

Tabella 286. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQQA_XX_ENCODE_CASE_ONE abilitazione	1	X'00000001'
MQQA_NOT_SHAREABLE	0	X'00000000'

Indurimento back-out

Tabella 287. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQQA_BACKOUT_HARDENED	1	X'00000001'
MQQA_BACKOUT_NOT_HARDENED	0	X'00000000'

MQQDT_ * (Tipi di definizione coda)

Tabella 288. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQQDT_PREDEFINED	1	X'00000001'
MQQDT_PERMANENT_DYNAMIC	2	X'00000002'
MQQDT_TEMPORARY_DYNAMIC	3	X'00000003'
MQQDT_SHARED_DYNAMIC	4	X'00000004'

MQQF_ * (Indicatori coda)

Tabella 289. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
Coda MQQF_LOCAL	1	X'00000001'
MQQF_CLWL_USEQ_ANY	64	X'00000040'
MQQF_CLWL_USEQ_LOCALE	128	X'00000080'

MQQMDT_ * (Formato del comando Tipi di definizione del gestore code)

Tabella 290. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQQMDT_EXPLICIT_CLUSTER_SENDER	1	X'00000001'
MQQMDT_AUTO_CLUSTER_SENDER	2	X'00000002'
MQQMDT_AUTO_EXP_CLUSTER_SENDER	4	X'00000004'
RICEVENTE MQQMDT_CLUSTER_DESTINATARIO	3	X'00000003'

MQQMF_ * (Indicatori gestore code)

Tabella 291. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQQMF_REPOSITORY_Q_MGR	2	X'00000002'
MQQMF_CLUSSDR_USER_DEFINED	8	X'00000008'
MQQMF_CLUSSDR_AUTO_DEFINED	16	X'00000010'
MQQMF_AVAILABLE	32	X'00000020'

MQQMFAC_ * (Formato del comando Funzione gestore code)

Tabella 292. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
BRIDGE mqqmfac_ims_bridge	1	X'00000001'
MQQMFAC_DB2	2	X'00000002'

MQQMSTA_ * (Formato del comando Stato gestore code)

Tabella 293. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQQMSTA_STARTING	1	X'00000001'
MQQMSTA_RUNNING	2	X'00000002'
MQQMSTA_QUIESCING	3	X'00000003'

MQQMT_ * (Formato del comando Tipi di gestore code)

Tabella 294. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
NORMAL MQQMT	0	X'00000000'
MQQMT_REPOSITORY	1	X'00000001'

MQQO_ * (Opzioni sospensione formato comando)

Tabella 295. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
SÌ MQQO	1	X'00000001'
MQQO_NO	0	X'00000000'

MQQSGD_ * (Disposizioni del gruppo di condivisione code)

Tabella 296. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQQSGD_ALL	-1	X'FFFFFFFF'
MQQSGD_Q_MGR	0	X'00000000'
MQQSGD_XX_ENCODE_CASE_ONE copia	1	X'00000001'
MQQSGD_SHARED	2	X'00000002'
GRUPPO_QGS	3	X'00000003'

Tabella 296. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQQSGD_PRIVATE	4	X'00000004'
MQQSGD_LIVE	6	X'00000006'

MQQSGS_ * (Stato del gruppo di condivisione code in formato comando)

Tabella 297. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQQSGS_UNKNOWN	0	X'00000000'
MQQSGS_CREATO	1	X'00000001'
MQQSGS_ATTIVO	2	X'00000002'
MQQSGS_INATTIVO	3	X'00000003'
MQQSGS_NON RIUSCITO	4	X'00000004'
MQQSGS_XX_ENCODE_CASE_ONE fine	5	X'00000005'

MQQSIE_ * (Formato del comando Servizio coda - Eventi intervallo)

Tabella 298. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQQSIE_NONE	0	X'00000000'
MQQSIE_HIGH	1	X'00000001'
MQQSIE_OK	2	X'00000002'

MQQSO_ * (Formato del comando Opzioni di apertura stato coda per SET, BROWSE, INPUT)

Tabella 299. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQQSO_NO	0	X'00000000'
SÌ MQQSO	1	X'00000001'
MQQSO_SHARED	1	X'00000001'
MQQSO_ESCLUSIVO	2	X'00000002'

MQQSOT_ * (Formato del comando Tipi di apertura stato coda)

Tabella 300. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQQSOT_ALL	1	X'00000001'
INPUT MQQSO	2	X'00000002'
OUTPUT MQQSOT	3	X'00000003'

MQQSUM_ * (Formato del comando Messaggi senza commit dello stato della coda)

Tabella 301. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQQSUM_SÌ	1	X'00000001'
MQQSUM_NO	0	X'00000000'

MQQT_ * (Tipi di coda e tipi di coda estesi)

Tipi di coda

Tabella 302. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
LOCALE MQQT	1	X'00000001'
MODEL MQQT	2	X'00000002'
ALIAS MQQT	3	X'00000003'
REMOTE MQQT	6	X'00000006'
CLUSTER MQQT_	7	X'00000007'

Tipi di coda estesa

Tabella 303. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQQT_ALL	1001	X'000003E9'

MQRC_ * (codici motivo)

Tabella 304. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQRC_NONE	0	X'00000000'
MQRC_APPL_PRIMO	900	X'00000384'
LAST MQRC_APPL	999	X'000003E7'
ERRORE TIPO_Q_ALIAS_MQRC	2001	X'000007D1'
MQRC_ALREADY_CONNECTED	2002	X'000007D2'
MQRC_BACK_OUT	2003	X'000007D3'
ERRORE MQRC_BUFFER_	2004	X'000007D4'
ERRORE MQRC_BUFFER_LENGTH	2005	X'000007D5'
MQRC_CHAR_ATTR_LENGTH_ERROR	2006	X'000007D6'
ERRORE MQRC_CHAR_ATTRS_	2007	X'000007D7'
MQRC_CHAR_ATTRS_TOO_SHORT	2008	X'000007D8'
MQRC_CONNECTION_BROKEN	2009	X'000007D9'
ERRORE MQRC_DATA_LENGTH	2010	X'000007DA'
MQRC_DYNAMIC_Q_NAME_ERROR	2011	X'000007DB'
ERRORE MQRC_ENVIRONMENT_ERROR	2012	X'000007DC'

Tabella 304. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
ERRORE DI MQRC_EXPIRY_	2013	X'000007DD'
ERRORE MQRC_FEEDBACK	2014	X'000007DE'
MQRC_GET_INHIBITED	2016	X'000007E0'
MQRC_HANDLE_NON_DISPONIBILE	2017	X'000007E1'
ERRORE MQRC_HCONN	2018	X'000007E2'
ERRORE MQRC_HOBJ_R	2019	X'000007E3'
ERRORE MQRC_INHIBIT_VALUE_	2020	X'000007E4'
ERRORE MQRC_INT_ATTR_COUNT_	2021	X'000007E5'
MQRC_INT_ATTR_COUNT_TOO_SMALL	2022	X'000007E6'
ERRORE - MQRC_INT_ATTRS_ARRAY_ERROR	2023	X'000007E7'
MQRC_SYNCPOINT_LIMITE_RAGGIUNTO	2024	X'000007E8'
MQRC_MAX_CONNS_LIMIT_REACHED	2025	X'000007E9'
ERRORE MQRC_MD	2026	X'000007EA'
MQRC_MISSING_REPLY_TO_Q	2027	X'000007EB'
ERRORE MQRC_MSG_TYPE_	2029	X'000007ED'
MQRC_MSG_TOO_BIG_FOR_Q	2030	X'000007EE'
MQRC_MSG_TOO_BIG_FOR_Q_MGR	2031	X'000007EF'
MQRC_NO_MSG_AVAILABLE	2033	X'000007F1'
MQRC_NO_MSG_UNDER_CURSOR	2034	X'000007F2'
MQRC_NOT_AUTHORIZED	2035	X'000007F3'
MQRC_NOT_OPEN_FOR_BROWSE	2036	X'000007F4'
MQRC_NOT_OPEN_FOR_INPUT	2037	X'000007F5'
MQRC_NOT_OPEN_FOR_INQUIRE	2038	X'000007F6'
MQRC_NOT_OPEN_FOR_OUTPUT	2039	X'000007F7'
MQRC_NOT_OPEN_FOR_SET	2040	X'000007F8'
MQRC_OBJECT_CHANGED	2041	X'000007F9'
MQRC_OBJECT_IN_USE	2042	X'000007FA'
ERRORE TIPO_OGGETTO_MQRC	2043	X'000007FB'
ERRORE MQRC_O	2044	X'000007FC'
MQRC_OPTION_NOT_VALID_FOR_TYPE	2045	X'000007FD'
ERRORE MQRC_OPTIONS_	2046	X'000007FE'
ERRORE MQRC_PERSISTENCE	2047	X'000007FF'
MQRC_PERSIST_NOT_ALLOWED	2048	X'00000800'
MQRC_PRIORITY_EXCEEDS_XX_ENCODE_CASE_ONE massimo	2049	X'00000801'
ERRORE MQRC_PRIORITY_ERROR	2050	X'00000802'
MQRC_PUT_INIBITO	2051	X'00000803'
MQRC_Q_XX_ENCODE_CASE_ONE eliminato	2052	X'00000804'
MQRC_Q_FULL	2053	X'00000805'
MQR_Q_NO_EMPTY	2055	X'00000807'

Tabella 304. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQRC_Q_SPACE_NOT_AVAILABLE	2056	X'00000808'
ERRORE MQRC_Q_TYPE_	2057	X'00000809'
ERRORE MQRC_Q_MGR_NAME_	2058	X'0000080A'
MQRC_Q_MGR_NOT_AVAILABLE	2059	X'0000080B'
ERRORE MQRC_REPORT_OPTIONS_	2061	X'0000080D'
MQRC_SECOND_MARK_NON_CONSENTITO	2062	X'0000080E'
ERRORE MQRC_SECURITY_ERROR	2063	X'0000080F'
ERRORE MQRC_SELECTOR_COUNT	2065	X'00000811'
MQRC_SELECTOR_LIMIT_EXCEEDED	2066	X'00000812'
ERRORE DI MQRC_SELECTOR_ERROR	2067	X'00000813'
MQRC_SELECTOR_NOT_FOR_TIPO	2068	X'00000814'
MQRC_SIGNAL_OUTSTANDING	2069	X'00000815'
MQRC_SIGNAL_REQUEST_ACCEPTED	2070	X'00000816'
MQRC_STORAGE_NON_DISPONIBILE	2071	X'00000817'
MQRC_SYNCPOINT_NOT_AVAILABLE	2072	X'00000818'
ERRORE MQRC_TRIGGER_CONTROL_	2075	X'0000081B'
ERRORE MQRC_TRIGGER_DEPTH_ERROR	2076	X'0000081C'
MQRC_TRIGGER_MSG_PRIORITY_ERR	2077	X'0000081D'
ERRORE MQRC_TRIGGER_TIPO	2078	X'0000081E'
MQRC_TRUNCATED_MSG_ACCEPTED	2079	X'0000081F'
MQRC_TRUNCATED_MSG_NON RIUSCITO	2080	X'00000820'
MQRC_UNKNOWN_ALIAS_BASE_Q	2082	X'00000822'
MQRC_UNKNOWN_OBJECT_NAME	2085	X'00000825'
MQRC_UNKNOWN_OBJECT_Q_MGR	2086	X'00000826'
MQRC_UNKNOWN_REMOTE_Q_MGR	2087	X'00000827'
ERRORE INTERVAL_WAIT_MQRC	2090	X'0000082A'
MQRC_XMIT_Q_TYPE_ERROR	2091	X'0000082B'
MQRC_XMIT_Q_USAGE_ERRORE	2092	X'0000082C'
MQRC_NOT_OPEN_FOR_PASS_ALL	2093	X'0000082D'
MQRC_NOT_OPEN_FOR_PASS_IDENT	2094	X'0000082E'
MQRC_NOT_OPEN_FOR_SET_ALL	2095	X'0000082F'
MQRC_NOT_OPEN_FOR_SET_IDENT	2096	X'00000830'
ERRORE MQRC_CONTEXT_HANDLE_	2097	X'00000831'
MQRC_CONTEXT_NOT_AVAILABLE	2098	X'00000832'
MQRC_SIGNAL1_ERROR	2099	X'00000833'
MQRC_OBJECT_ALREADY_EXISTS	2100	X'00000834'
MQRC_OBJECT_DAMAGED	2101	X'00000835'
PROBLEMA_RISORSA_MQRC_	2102	X'00000836'
MQRC_ANOTHER_Q_MGR_CONNECTED	2103	X'00000837'

Tabella 304. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQRC_UNKNOWN_REPORT_OPZIONE	2104	X'00000838'
ERRORE MQRC_STORAGE_CLASS_	2105	X'00000839'
MQRC_COD_NOT_VALID_FOR_XCF_Q	2106	X'0000083A'
MQRC_XWAIT_CANCELED	2107	X'0000083B'
ERRORE MQRC_XWAIT	2108	X'0000083C'
MQRC_SUPPRESSED_BY_EXIT	2109	X'0000083D'
ERRORE MQRC_FORMAZIONE	2110	X'0000083E'
ERRORE CCSID DI MQRC_SOURCE_	2111	X'0000083F'
ERRORE DI INIZIALIZZAZIONE MQRC_SOURCE_INTEGER_	2112	X'00000840'
ERRORE DI RIMOZIONE MQRC_SOURCE_DECIMAL_ENC_	2113	X'00000841'
ERRORE_ERRORE_ORIGINE_RISORSE MQRC	2114	X'00000842'
ERRORE MQRC_TARGET_CCSID_	2115	X'00000843'
ERRORE MQRC_TARGET_INTEGER_ENC	2116	X'00000844'
ERRORE DI RETE MQRC_TARGET_DECIMAL_ENC_ERROR	2117	X'00000845'
ERRORE DI RETE MQRC_TARGET_FLOAT_	2118	X'00000846'
MQRC_NOT_CONVERTED	2119	X'00000847'
MQRC_CONVERTED_MSG_TOO_BIG	2120	X'00000848'
MQRC_TRUNCATED	2120	X'00000848'
MQRC_NO_EXTERNAL_PARTECIPANTI	2121	X'00000849'
MQRC_PARTICIPANT_NON_DISPONIBILE	2122	X'0000084A'
MQRC_OUTCOME_MIXED	2123	X'0000084B'
MQRC_OUTCOME_PENDING	2124	X'0000084C'
MQRC_BRIDGE_STARTED	2125	X'0000084D'
MQRC_BRIDGE_STOPPED	2126	X'0000084E'
SCARSE_ARCHIVIAZIONE_ADATTATORE_MQRC_	2127	X'0000084F'
MQRC_UOW_IN_PROVERDE	2128	X'00000850'
ERRORE CARICAMENTO MQRC_ADAPTER_CONN_LOAD	2129	X'00000851'
MQRC_ADAPTER_SERV_LOAD_ERROR	2130	X'00000852'
ERRORE DEFS MQRC_ADAPTER_	2131	X'00000853'
ERRORE CARICAMENTO MQRC_ADAPTER_DEFS_	2132	X'00000854'
ERRORE CARICAMENTO MQRC_ADAPTER_CONV	2133	X'00000855'
ERRORE BO_MQRC	2134	X'00000856'
ERRORE MQRC_DH_	2135	X'00000857'
MQRC_MULTIPLE_MOTIVI	2136	X'00000858'
MQRC_OPEN_NON RIUSCITO	2137	X'00000859'
ERRORE CARICAMENTO MQRC_ADAPTER_DISC	2138	X'0000085A'
ERRORE_ERRORE_MQRC	2139	X'0000085B'
MQRC_CICS_WAIT_NON RIUSCITO	2140	X'0000085C'
ERRORE MQRC_DLH	2141	X'0000085D'

Tabella 304. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
ERRORE MQRC_HEADER_	2142	X'0000085E'
ERRORE_ORIGINE_RISORSE MQRC	2143	X'0000085F'
ERRORE MQRC_TARGET_LENGTH	2144	X'00000860'
ERRORE_ORIGINE_RISORSE MQRC	2145	X'00000861'
ERRORE - BUFFER_MQRC_TARGET_	2146	X'00000862'
ERRORE MQRC_IIH	2148	X'00000864'
ERRORE MQRC_PCF	2149	X'00000865'
ERRORE MQRC_DBCS	2150	X'00000866'
ERRORE MQRC_OBJECT_NAME_ERROR	2152	X'00000868'
MQRC_OBJECT_Q_MGR_NAME_ERROR	2153	X'00000869'
ERRORE MQRC_RECS_PRESENT_	2154	X'0000086A'
ERRORE MQRC_OBJECT_RECORDS	2155	X'0000086B'
ERRORE DI RISPOSTA MQR_RESPONSE_RECORDS	2156	X'0000086C'
MQRC_ASID_MISMATCH	2157	X'0000086D'
ERRORE MQRC_PMO_RECORD_FLAGS_ERROR	2158	X'0000086E'
ERRORE MQRC_PUT_MSG_RECORDS_	2159	X'0000086F'
MQRC_CONN_ID_IN_USE	2160	X'00000870'
MQRC_Q_MGR_QUIESCING	2161	X'00000871'
MQRC_Q_MGR_STOPPING	2162	X'00000872'
MQRC_DUPLICATE_RECOV_COORD	2163	X'00000873'
ERRORE PMO_MQRC	2173	X'0000087D'
MQRC_API_EXIT_NOT_FOUND	2182	X'00000886'
ERRORE USCITA MQRC_API	2183	X'00000887'
MQRC_REMOTE_Q_NAME_ERROR	2184	X'00000888'
MQRC_INCONSIST_PERSISTENZA	2185	X'00000889'
ERRORE MQRC_GMO	2186	X'0000088A'
MQRC_CICS_BRIDGE_RESTRICTION	2187	X'0000088B'
MQRC_STOPPED_BY_CLUSTER_EXIT	2188	X'0000088C'
MQRC_CLUSTER_RESOLUTION_ERRORE	2189	X'0000088D'
MQRC_CONVERTED_STRING_TOO_BIG	2190	X'0000088E'
ERRORE MQRC_TMC_	2191	X'0000088F'
MQRC_PAGESET_FULL	2192	X'00000890'
MQRC_STORAGE_MEDIUM_FULL	2192	X'00000890'
ERRORE MQRC_PAGESET_	2193	X'00000891'
MQRC_NAME_NOT_VALID_FOR_TYPE	2194	X'00000892'
ERRORE MQRC_UNEXPECTED_	2195	X'00000893'
MQRC_UNKNOWN_XMIT_Q	2196	X'00000894'
MQRC_UNKNOWN_DEF_XMIT_Q	2197	X'00000895'
MQRC_DEF_XMIT_Q_TYPE_ERROR	2198	X'00000896'

Tabella 304. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQRC_DEF_XMIT_Q_USAGE_ERROR	2199	X'00000897'
MQRC_MSG_MARKED_BROWSE_CO_OP	2200	X'00000898'
NAME_MQRC_IN_USE	2201	X'00000899'
MQRC_CONNECTION_QUIESCING	2202	X'0000089A'
MQRC_CONNECTION_STOPPING	2203	X'0000089B'
MQRC_ADAPTER_NON_DISPONIBILE	2204	X'0000089C'
ERRORE ID MQRC_MSG_	2206	X'0000089E'
ERRORE ID CORREL_MQR_	2207	X'0000089F'
ERRORE MQRC_FILE_SYSTEM	2208	X'000008A0'
MQRC_NO_MSG_LOCKED	2209	X'000008A1'
ERRORE MQRC_SOAP_DOTNET_	2210	X'000008A2'
MQRC_SOAP_AXIS_ERRORE	2211	X'000008A3'
ERRORE MQRC_SOAP_URL_	2212	X'000008A4'
MQRC_FILE_NOT_AUDITED	2216	X'000008A8'
MQRC_CONNECTION_NOT_AUTHORIZED	2217	X'000008A9'
MQRC_MSG_TOO_BIG_FOR_CHALLEGATO	2218	X'000008AA'
MQRC_CALL_IN_PROVERDE	2219	X'000008AB'
ERRORE RMH_MQRC	2220	X'000008AC'
MQRC_Q_MGR_ATTIVO	2222	X'000008AE'
MQRC_Q_MGR_NOT_ATTIVO	2223	X'000008AF'
Q_MQR_DEPTH_HIGH	2224	X'000008B0'
MQRC_Q_DEPTH_LOW	2225	X'000008B1'
MQRC_Q_INTERVALLO_SERVIZIO_ELEVATO	2226	X'000008B2'
MQRC_Q_SERVICE_INTERVAL_OK	2227	X'000008B3'
ERRORE MQRC_RFH_HEADER_FIELD_ERRORE	2228	X'000008B4'
ERRORE MQRC_RAS_PROPERTY_ERROR	2229	X'000008B5'
MQRC_UNIT_OF_WORK_NOT_STARTED	2232	X'000008B8'
MQRC_CHANNEL_AUTO_DEF_OK	2233	X'000008B9'
ERRORE ERRORE_DI_AUTOMAZIONE MQRC_CHANNEL_	2234	X'000008BA'
ERRORE MQRC_CFH	2235	X'000008BB'
ERRORE FILTRO MQRC	2236	X'000008BC'
ERRORE MQRC_CFIN	2237	X'000008BD'
ERRORE MQRC_CFSL	2238	X'000008BE'
ERRORE MQRC_CFST	2239	X'000008BF'
GRUPPO_INCOMPLE_MQRC	2241	X'000008C1'
MQRC_INCOMPLETE_MSG	2242	X'000008C2'
CCSIDS INCONSIST_MQRC	2243	X'000008C3'
MQRC_INCONSISTENT_ENCODINGS	2244	X'000008C4'
UOW MQRC_INCONSISTENT_	2245	X'000008C5'

Tabella 304. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQRC_INVALID_MSG_UNDER_CURSOR	2246	X'000008C6'
ERRORE MQRC_MATCH_OPTIONS_	2247	X'000008C7'
ERRORE MQRC_MDE	2248	X'000008C8'
ERRORE MQRC_MSG_FLAGS_	2249	X'000008C9'
ERRORE MQRC_MSG_SEQ_NUMBER_	2250	X'000008CA'
ERRORE MQRC_OFFSET_	2251	X'000008CB'
MQRC_ORIGINAL_LENGTH_ERRORE	2252	X'000008CC'
MQRC_SEGMENT_LENGTH_ZERO	2253	X'000008CD'
MQRC_UOW_NON_DISPONIBILE	2255	X'000008CF'
MQRC_WRONG_GMO_VERSIONE	2256	X'000008D0'
MQRC_WRONG_MD_VERSIONE	2257	X'000008D1'
ERRORE MQRC_GROUP_ID_	2258	X'000008D2'
BROWSE INCONSIST_MQRC	2259	X'000008D3'
ERRORE MQRC_XQH	2260	X'000008D4'
MQRC_SRC_ENV_ERRORE	2261	X'000008D5'
ERRORE MQRC_SR_NAME_	2262	X'000008D6'
ERRORE AMBIENTE MQRC_DEST	2263	X'000008D7'
ERRORE NAME_DEST_MQRC_	2264	X'000008D8'
ERRORE TM_MQRC	2265	X'000008D9'
ERRORE DI USCITA MQRC_CLUSTER_	2266	X'000008DA'
ERRORE DI CARICAMENTO MQRC_CLUSTER_EXIT_LOAD	2267	X'000008DB'
MQRC_CLUSTER_PUT_INIBITO	2268	X'000008DC'
MQRC_CLUSTER_RESOURCE_ERROR	2269	X'000008DD'
MQRC_NO_DESTINATIONS_AVAILABLE	2270	X'000008DE'
MQRC_CONN_TAG_IN_USO	2271	X'000008DF'
MQRC_PARTIALLY_CONVERTED	2272	X'000008E0'
ERRORE MQRC_CONNECTION_	2273	X'000008E1'
MQRC_OPTION_ENVIRONMENT_ERROR	2274	X'000008E2'
ERRORE MQRC_CD	2277	X'000008E5'
ERRORE MQRC_CLIENT_CONN_	2278	X'000008E6'
MQRC_CHANNEL_STOPPED_BY_USER	2279	X'000008E7'
ERRORE MQRC_HCONFIG	2280	X'000008E8'
ERRORE MQRC_FUNCTION_	2281	X'000008E9'
MQRC_CHANNEL_STARTED	2282	X'000008EA'
MQRC_CHANNEL_STOPPED	2283	X'000008EB'
MQRC_CHANNEL_CONV_ERROR	2284	X'000008EC'
MQRC_SERVICE_NOT_AVAILABLE	2285	X'000008ED'
MQRC_INITIALIZATION_FAILED	2286	X'000008EE'
MQRC_TERMINATION_FAILED	2287	X'000008EF'

Tabella 304. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQRC_UNKNOWN_Q_NAME	2288	X'000008F0'
ERRORE_SERVIZIO_MQRC	2289	X'000008F1'
MQRC_Q_ALREADY_EXISTS	2290	X'000008F2'
MQRC_USER_ID_NON_DISPONIBILE	2291	X'000008F3'
MQRC_UNKNOWN_ENTITY	2292	X'000008F4'
MQRC_UNKNOWN_AUTH_ENTITY	2293	X'000008F5'
MQRC_UNKNOWN_REF_OBJECT	2294	X'000008F6'
MQRC_CHANNEL_ACTIVATED	2295	X'000008F7'
MQRC_CHANNEL_NOT_ACTIVATED	2296	X'000008F8'
MQRC_UOW_CANCELED	2297	X'000008F9'
MQRC_FUNZIONE_NON_SUPPORTATA	2298	X'000008FA'
ERRORE TIPO_SELETTORE MQRC_	2299	X'000008FB'
ERRORE MQRC_COMMAND_TYPE_	2300	X'000008FC'
ERRORE_ISTANZA_MULTIPLE_MQRC_	2301	X'000008FD'
MQRC_SYSTEM_ITEM_NOT_ALTERABLE	2302	X'000008FE'
ERRORE MQRC_BAG_CONVERSION_	2303	X'000008FF'
MQRC_SELECTOR_OUT_OF_RANGE	2304	X'00000900'
MQRC_SELECTOR_NOT_UNIQUE	2305	X'00000901'
MQRC_INDEX_NOT_PRESENT	2306	X'00000902'
ERRORE MQRC_STRING_	2307	X'00000903'
MQRC_ENCODING_NOT_SUPPORTED	2308	X'00000904'
MQRC_SELECTOR_NOT_PRESENTE	2309	X'00000905'
ERRORE MQRC_OUT_SELECTOR_ERROR	2310	X'00000906'
MQRC_STRING_TRUNCATED	2311	X'00000907'
TIPO_WRONG_SELECTOR_MQRC_	2312	X'00000908'
TIPO_ITEM_INCONSIST_MQRC_	2313	X'00000909'
ERRORE MQRC_INDEX	2314	X'0000090A'
MQRC_SYSTEM_BAG_NOT_ALTERABLE	2315	X'0000090B'
ERRORE CONTEGGIO_ERRORI MQRC_IT	2316	X'0000090C'
MQRC_FORMAT_NOT_SUPPORTED	2317	X'0000090D'
MQRC_SELECTOR_NOT_SUPORTED	2318	X'0000090E'
ERRORE MQRC_ITEM_VALUE_	2319	X'0000090F'
ERRORE MQRC_HBAG_	2320	X'00000910'
MQRC_PARAMETER_MISSING	2321	X'00000911'
MQRC_CMD_SERVER_NOT_AVAILABLE	2322	X'00000912'
ERRORE MQRC_STRING_LENGTH	2323	X'00000913'
ERRORE MQRC_INQUIRY_COMMAND_	2324	X'00000914'
MQRC_NESTED_BAG_NOT_SUPORTED	2325	X'00000915'
TIPO_MQRC_BAG_WRONG_	2326	X'00000916'

Tabella 304. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
ERRORE TIPO_MQRC_ITEM_TYPE_	2327	X'00000917'
MQRC_SYSTEM_BAG_NOT_DELETABLE	2328	X'00000918'
MQRC_SYSTEM_ITEM_NOT_DELETABLE	2329	X'00000919'
ERRORE MQRC_CODED_CHAR_SET_ID_	2330	X'0000091A'
ERRORE MQRC_MSG_TOKEN_	2331	X'0000091B'
MQRC_MISSING_WIH	2332	X'0000091C'
ERRORE MQRC_WIH	2333	X'0000091D'
ERRORE MQRC_RFH	2334	X'0000091E'
ERRORE MQRC_RFH_STRING_	2335	X'0000091F'
MQRC_RFH_COMMAND_ERROR	2336	X'00000920'
ERRORE MQRC_RFH_PARM	2337	X'00000921'
MQRC_RFH_DUPLICATE_PARM	2338	X'00000922'
MQRC_RFH_PARM_MISSING	2339	X'00000923'
ERRORE MQRC_CHAR_CONVERSION_	2340	X'00000924'
MQRC_UCS2_CONVERSION_ERROR	2341	X'00000925'
MQRC_DB2_NOT_AVAILABLE	2342	X'00000926'
MQRC_OBJECT_NOT_UNIQUE	2343	X'00000927'
MQRC_CONN_TAG_NOT_RELEASED	2344	X'00000928'
MQRC_CF_NOT_AVAILABLE	2345	X'00000929'
MQRC_CF_STRUC_IN_USO	2346	X'0000092A'
MQRC_CF_STRUC_LIST_HDR_IN_USE	2347	X'0000092B'
MQRC_CF_STRUC_AUTH_NON RIUSCITO	2348	X'0000092C'
ERRORE MQRC_CF_STRUC_	2349	X'0000092D'
MQRC_CONN_TAG_NO_USABLE	2350	X'0000092E'
MQRC_GLOBAL_UOW_CONFLICT	2351	X'0000092F'
MQRC_LOCAL_UOW_CONFLICT	2352	X'00000930'
MQRC_HANDLE_IN_USE_FOR_UOW	2353	X'00000931'
ERRORE MQRC_UOW_ENLISTMENT_ERROR	2354	X'00000932'
MQRC_UOW_MIX_NON_SUPPORTATO	2355	X'00000933'
ERRORE MQRC_WXP	2356	X'00000934'
ERRORE MQRC_CURRENT_RECORD_ERRORE	2357	X'00000935'
ERRORE MQRC_NEXT_OFFSET_	2358	X'00000936'
MQRC_NO_RECORD_AVAILABLE	2359	X'00000937'
MQRC_OBJECT_LEVEL_INCOMPATIBILE	2360	X'00000938'
ERRORE MQRC_NEXT_RECORD_ERRORE	2361	X'00000939'
MQRC_BACKOUT_THRESHOLD_REACHED	2362	X'0000093A'
MQRC_MSG_NOT_MATCHED	2363	X'0000093B'
ERRORE MQRC_JMS_FORMATO	2364	X'0000093C'
MQRC_SEGMENTS_NOT_SUPPORTED	2365	X'0000093D'

Tabella 304. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQRC_WRONG_CF_LEVEL	2366	X'0000093E'
MQRC_CONFIG_CREA_OBJECT	2367	X'0000093F'
MQRC_CONFIG_CHANGE_OBJECT	2368	X'00000940'
MQRC_CONFIG_DELETE_OBJECT	2369	X'00000941'
MQRC_CONFIG_REFRESH_OBJECT	2370	X'00000942'
MQRC_CHANNEL_SSL_ERROR	2371	X'00000943'
MQRC_DEFINE IPANT_NOT_DEFINED	2372	X'00000944'
MQRC_CF_STRUC_NON RIUSCITO	2373	X'00000945'
ERRORE USCITA MQRC_API	2374	X'00000946'
MQRC_API_EXIT_INIT_ERROR	2375	X'00000947'
ERRORE USCITA MQRC_API	2376	X'00000948'
ERRORE REASON_MQRC_EXIT_	2377	X'00000949'
ERRORE MQRC_RESERVED_VALUE_	2378	X'0000094A'
MQRC_NO_DATA_AVAILABLE	2379	X'0000094B'
ERRORE MQRC_SCO	2380	X'0000094C'
MQRC_KEY_REPOSITORY_ERROR	2381	X'0000094D'
MQRC_CRYPTO_HARDWARE_ERROR	2382	X'0000094E'
ERRORE MQRC_AUTH_INFO_REC_COUNT_	2383	X'0000094F'
ERRORE MQRC_AUTH_INFO_REC_	2384	X'00000950'
ERRORE MQRC_AIR	2385	X'00000951'
MQRC_AUTH_INFO_TYPE_ERRORE	2386	X'00000952'
ERRORE MQRC_AUTH_INFO_CONN_NAME_ERROR	2387	X'00000953'
ERRORE MQRC_LDAP_USER_NAME_ERROR	2388	X'00000954'
MQRC_LDAP_USER_NAME_LENGTH_ERR	2389	X'00000955'
ERRORE MQRC_LDAP_PASSWORD_ERRORE	2390	X'00000956'
MQRC_SSL_ALREADY_INITIALIZED	2391	X'00000957'
ERRORE MQRC_SSL_CONFIG	2392	X'00000958'
ERRORE MQRC_SSL_INITIALIZATION_ERROR	2393	X'00000959'
MQRC_Q_INDEX_TYPE_ERROR	2394	X'0000095A'
MQRC_CFBS_ERRORE	2395	X'0000095B'
MQRC_SSL_NOT_ALLOWED	2396	X'0000095C'
ERRORE MQRC_JSSE	2397	X'0000095D'
MQRC_SSL_PEER_NAME_MISMATCH	2398	X'0000095E'
ERRORE MQRC_SSL_PEER_NAME_ERROR	2399	X'0000095F'
MQRC_UNSUPPORT_CIPHER_SUITE	2400	X'00000960'
MQRC_SSL_CERTIFICATE_REVOKED	2401	X'00000961'
ERRORE - CERT_STORE_MQRC_SSL_CERT_	2402	X'00000962'
ERRORE CARICAMENTO MQRC_CLIENT_EXIT_LOAD	2406	X'00000966'
ERRORE MQRC_CLIENT_EXIT_ERROR	2407	X'00000967'

Tabella 304. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQRC_UOW_XX_ENCODE_CASE_ONE sincronizzato	2408	X'00000968'
ERRORE MQRC_SSL_KEY_RESET_ERROR	2409	X'00000969'
MQRC_UNKNOWN_COMPONENT_NAME	2410	X'0000096A'
STATO LOGGER_MQRC	2411	X'0000096B'
MQRC_COMMAND_MQSC	2412	X'0000096C'
PCF COMMAND_MQRC	2413	X'0000096D'
ERRORE MQRC_CFIF	2414	X'0000096E'
ERRORE MQRC_CFF	2415	X'0000096F'
ERRORE MQRC_CFGR	2416	X'00000970'
MQRC_MSG_NOT_ALLOWED_IN_XX_ENCODE_CASE_ONE gruppo	2417	X'00000971'
ERRORE FUNZIONAMENTO FILTRO MQRC	2418	X'00000972'
ERRORE MQRC_NESTED_SELECTOR_ERROR	2419	X'00000973'
ERRORE MQRC_EPH	2420	X'00000974'
ERRORE MQRC_RFH_FORMATO	2421	X'00000975'
ERRORE MQRC_CFF	2422	X'00000976'
MQRC_CLIENT_CHANNEL_CONFLICT	2423	X'00000977'
ERRORE MQRC_S	2424	X'00000978'
ERRORE STRINGA MQRC_TOPIC_	2425	X'00000979'
ERRORE MQRC_STS_	2426	X'0000097A'
MQRC_NO_SUBSCRIPZIONE	2428	X'0000097C'
MQRC_SUBSCRIPTION_IN_USE	2429	X'0000097D'
ERRORE TIPO_STATO_MQRC	2430	X'0000097E'
ERRORE MQRC_SUB_USER_DATA_	2431	X'0000097F'
MQRC_SUB_ALREADY_EXISTS	2432	X'00000980'
MQRC_IDENTITY_MISMATCH	2434	X'00000982'
ERRORE SUB_ALTER_MQRC_	2435	X'00000983'
MQRC_DURABILITY_NOT_ALLOWED	2436	X'00000984'
MQRC_NO_RETAINED_MSG	2437	X'00000985'
ERRORE_ERRORE_MQRC	2438	X'00000986'
ERRORE MQRC_SUB_NAME_	2440	X'00000988'
ERRORE STRINGA MQRC_OBJECT_	2441	X'00000989'
ERRORE MQRC_PROPERTY_NAME_	2442	X'0000098A'
SEGMENTAZIONE_MQRC_NON_CONSENTITA	2443	X'0000098B'
ERRORE MQRC_CBD	2444	X'0000098C'
ERRORE MQRC_CTLO_	2445	X'0000098D'
MQRC_NO_CALLBACKS_ATTIVO	2446	X'0000098E'
MQRC_CALLBACK_NON_REGISTRATO	2448	X'00000990'
MQRC_OPTIONS_CHANGED	2457	X'00000999'
MQRC_READ_AHEAD_MSGS	2458	X'0000099A'

Tabella 304. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
ERRORE MQRC_SELECTOR_SYNTAX_ERROR	2459	X'0000099B'
ERRORE MQRC_HMSG_	2460	X'0000099C'
ERRORE MQRC_CMHO_	2461	X'0000099D'
ERRORE DI MQRC_DMHO_	2462	X'0000099E'
ERRORE MQRC_SMPO	2463	X'0000099F'
ERRORE MQRC_IMPO	2464	X'000009A0'
MQRC_PROPERTY_NAME_TOO_BIG	2465	X'000009A1'
PROP_MQRC_XX_ENCODE_CASE_ONE valore_non_CONVERTED	2466	X'000009A2'
MQRC_PROP_TYPE_NON_SUPPORTATO	2467	X'000009A3'
MQRC_PROPERTY_VALUE_TOO_BIG	2469	X'000009A5'
MQRC_PROP_CONV_NON supportato	2470	X'000009A6'
PROPRIETÀ_MQRC_NON_DISPONIBILE	2471	X'000009A7'
MQRC_PROP_NUMBER_FORMAT_ERRORE	2472	X'000009A8'
ERRORE TIPO_PROFILO_XX_ENCODE_CASE_CAPS_LOCK_OFF MQRC_	2473	X'000009A9'
MQRC_PROPERTIES_TOO_BIG	2478	X'000009AE'
MQRC_PUT_NON_CONSERVATO	2479	X'000009AF'
MQRC_ALIAS_TARGTYPE_CHANGED	2480	X'000009B0'
ERRORE DMPO_MQRC_	2481	X'000009B1'
ERRORE MQRC_PD	2482	X'000009B2'
ERRORE MQRC_CALLBACK_TIPO	2483	X'000009B3'
ERRORE MQRC_CBD_OPTIONS_	2484	X'000009B4'
ERRORE MQRC_MAX_MSG_LENGTH	2485	X'000009B5'
ERRORE DI MQRC_CALLBACK_ROUTINE_ERROR	2486	X'000009B6'
ERRORE MQRC_CALLBACK_LINK	2487	X'000009B7'
ERRORE OPERAZIONE MQRC	2488	X'000009B8'
ERRORE MQRC_BMHO_	2489	X'000009B9'
MQRC_UNSUPPORT_PROPERTY	2490	X'000009BA'
MQRC_PROP_NAME_NON_CONVERTITO	2492	X'000009BC'
MQRC_GET_ENABLED	2494	X'000009BE'
MQRC_MODULE_NOT_FOUND	2495	X'000009BF'
ID_NON_VALIDO MQRC_MODULE	2496	X'000009C0'
MQRC_MODULE_ENTRY_NOT_FOUND	2497	X'000009C1'
MQRC_MIXED_CONTENT_NOT_ALLOWED	2498	X'000009C2'
MQRC_MSG_HANDLE_IN_USE	2499	X'000009C3'
MQRC_HCONN_ASYNC_ATTIVO	2500	X'000009C4'
ERRORE MQRC_MHBO_	2501	X'000009C5'
MQRC_PUBLICATION_FAILURE	2502	X'000009C6'
MQRC_SUB_INHIBITED	2503	X'000009C7'
MQRC_SELECTOR_ALWAYS_FALSE	2504	X'000009C8'

Tabella 304. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
ERRORE MQRC_XEPO_	2507	X'000009CB'
MQRC_DURABILITY_NOT_ALTERABLE	2509	X'000009CD'
MQRC_TOPIC_NOT_ALTERABLE	2510	X'000009CE'
MQRC_SUBLEVEL_NOT_ALTERABLE	2512	X'000009D0'
NOME PROFILO MQRC_PROPERTY_LENGTH_ERR	2513	X'000009D1'
MQRC_DUPLICATE_GROUP_SUB	2514	X'000009D2'
MQR_GROUPING_NOT_ALTERABLE	2515	X'000009D3'
MQRC_SELECTOR_INVALID_FOR_TYPE	2516	X'000009D4'
MQRC_HOBJ_QUIESCED	2517	X'000009D5'
MQRC_HOBJ_QUIESCED_NO_MSGS	2518	X'000009D6'
ERRORE MQRC_SELECTION_STRING_	2519	X'000009D7'
ERRORE STRINGA MQRC_RES_OBJECT_	2520	X'000009D8'
MQRC_CONNECTION_SUSPENDED	2521	X'000009D9'
DESTINAZIONE_NON_VALIDA	2522	X'000009DA'
MQRC_SOTTOSCRIZIONE_NONVALIDA	2523	X'000009DB'
MQRC_SELECTOR_NOT_ALTERABLE	2524	X'000009DC'
MQRC_RETAINED_MSG_Q_ERROR	2525	X'000009DD'
MQRC_RETAINED_NOT_DELIVERED	2526	X'000009DE'
MQRC_RFH_RESTRICTED_FORMAT_ERR	2527	X'000009DF'
MQRC_CONNECTION_STOPPED	2528	X'000009E0'
MQRC_ASYNC_UOW_CONFLICT	2529	X'000009E1'
MQRC_ASYNC_XA_CONFLICT	2530	X'000009E2'
MQRC_PUBSUB_INHIBITED	2531	X'000009E3'
MQRC_MSG_HANDLE_COPY_FAILURE	2532	X'000009E4'
MQRC_DEST_CLASS_NOT_ALTERABLE	2533	X'000009E5'
MQRC_OPERAZIONE NOT_ALLOWED	2534	X'000009E6'
ERRORE MQRC_AZIONE	2535	X'000009E7'
MQRC_CHANNEL_NOT_AVAILABLE	2537	X'000009E9'
MQRC_HOST_NON_DISPONIBILE	2538	X'000009EA'
ERRORE MQRC_CHANNEL_CONFIG	2539	X'000009EB'
MQRC_UNKNOWN_CHANNEL_NAME	2540	X'000009EC'
LICENZA DI PUBBLICAZIONE MQRC_LOOPING_PUBLICATION	2541	X'000009ED'
MQRC_ALREADY_JOINED	2542	X'000009EE'
MQRC_CHANNEL_SSL_XX_ENCODE_CASE_ONE avvertenza	2552	X'000009F8'
ERRORE URL OCSP_MQRC	2553	X'000009F9'
MQRC_CIPHER_SPEC_NOT_SUITE_B	2591	X'00000A1F'
ERRORE MQRC_SUITE_B_R	2592	X'00000A20'
ERRORE MQRC_PASSWORD_PROTECTION_ERROR	2594	X'00000A22'
ERRORE_INPUT_REOPEN_MQRC_REOPEN_EXCL_	6100	X'000017D4'

Tabella 304. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQRC_REOPEN_INQUIRE_ERROR	6101	X'000017D5'
MQRC_REOPEN_SAVED_CONTEXT_ERR	6102	X'000017D6'
ERRORE CODA MQRC_REOPEN_TEMPORARY_Q	6103	X'000017D7'
MQRC_ATTRIBUTE_LOCKED	6104	X'000017D8'
MQRC_CURSOR_NO_VALID	6105	X'000017D9'
ERRORE MQRC_ENCODING_	6106	X'000017DA'
ERRORE MQRC_STRUC_ID	6107	X'000017DB'
NULL_POINTER MQRC	6108	X'000017DC'
RIFERIMENTO MQRC_NO_CONNECTION_DI	6109	X'000017DD'
MQRC_NO_BUFFER	6110	X'000017DE'
MQRC_BINARY_DATA_LENGTH_ERROR	6111	X'000017DF'
MQRC_BUFFER_NOT_AUTOMATIC	6112	X'000017E0'
MQRC_BUFFER insufficiente	6113	X'000017E1'
DATI MQRC_INSUFFICIENT_	6114	X'000017E2'
DATA_TRUNCATED MQRC_	6115	X'000017E3'
LENGTH - ZERO_MQRC	6116	X'000017E4'
MQRC_NEGATIVE_LENGTH	6117	X'000017E5'
MQRC_NEGATIVE_OFFSET	6118	X'000017E6'
MQRC_INCONSISTENT_FORMAT	6119	X'000017E7'
MQRC_INCONSISTENT_OBJECT_STATE	6120	X'000017E8'
MQRC_CONTEXT_OBJECT_NOT_VALID	6121	X'000017E9'
ERRORE APERTURA CONTENUTO MQRC_	6122	X'000017EA'
ERRORE MQRC_STRUC_LENGTH	6123	X'000017EB'
MQRC_NOT_CONNECTED	6124	X'000017EC'
MQRC_NOT_OPEN	6125	X'000017ED'
MQRC_DISTRIBUTION_LIST_EMPTY	6126	X'000017EE'
OPZIONI MQRC_INCONSISTENT_OPEN_OPTIONS	6127	X'000017EF'
VERSIONE MQRC_WRONG_	6128	X'000017F0'
ERRORE MQRC_REFERENCE_ERROR	6129	X'000017F1'

MQRCCF_ * (Codici motivo intestazione formato comando)

Per ulteriori informazioni sulla risposta del programmatore, consultare Codici di errore PCF.

Tabella 305. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQRCCF_CFH_TYPE_ERROR	3001	X'00000BB9'
MQRCCF_CFH_LENGTH_ERROR	3002	X'00000BBA'
ERRORE MQRCCF_CFH_VERSION_	3003	X'00000BBB'
NUMBER_ERR MQRCCF_CFH_MSG_SEQ_	3004	X'00000BBC'
ERRORE MQRCCF_CFH_CONTROL_	3005	X'00000BBD'

Tabella 305. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
ERRORE MQRCCF_CFH_PARM_COUNT_	3006	X'00000BBE'
ERRORE MQRCCF_CFH_COMMAND_	3007	X'00000BBF'
MQRCCF_COMMAND_FAILED	3008	X'00000BC0'
MQRCCF_CFIN_LENGTH_ERROR	3009	X'00000BC1'
MQRCCF_CFST_LENGTH_ERROR	3010	X'00000BC2'
MQRCCF_CFST_STRING_LENGTH_ERR	3011	X'00000BC3'
ERRORE MQRCCF_FORCE_VALUE_	3012	X'00000BC4'
ERRORE TIPO DI STRUTTURA MQRCCF_	3013	X'00000BC5'
ERRORE MQRCCF_CFIN_PARM_ID	3014	X'00000BC6'
MQRCCF_CFST_PARM_ID_ERROR	3015	X'00000BC7'
ERRORE MQRCCF_MSG_LENGTH	3016	X'00000BC8'
MQRCCF_CFIN_DUPLICATE_PARM	3017	X'00000BC9'
MQRCCF_CFST_DUPLICATE_PARM	3018	X'00000BCA'
MQRCCF_PARM_COUNT_TOO_SMALL	3019	X'00000BCB'
MQRCCF_PARM_COUNT_TOO_BIG	3020	X'00000BCC'
MQRCCF_Q_ALREADY_IN_CELL	3021	X'00000BCD'
MQRCCF_Q_TYPE_ERROR	3022	X'00000BCE'
MQRCCF_MD_FORMAT_ERROR	3023	X'00000BCF'
MQRCCF_CFSL_LENGTH_ERROR	3024	X'00000BD0'
ERRORE MQRCCF_REPLACE_VALUE_	3025	X'00000BD1'
MQRCCF_CFIL_DUPLICATE_VALUE	3026	X'00000BD2'
ERRORE MQRCCF_CFIL_COUNT	3027	X'00000BD3'
MQRCCF_CFIL_LENGTH_ERROR	3028	X'00000BD4'
ERRORE MQRCCF_QUIESCE_VALUE_	3029	X'00000BD5'
ERRORE MQRCCF_MODE_VALUE_	3029	X'00000BD5'
ERRORE MQRCCF_MSG_SEQ_NUMBER_	3030	X'00000BD6'
ERRORE CONTEGGIO_DATA_PING_MQRCCF	3031	X'00000BD7'
MQRCCF_PING_DATA_COMPARE_ERROR	3032	X'00000BD8'
MQRCCF_CFSL_PARM_ID_ERROR	3033	X'00000BD9'
ERRORE MQRCCF_CHANNEL_TYPE_ERROR	3034	X'00000BDA'
ERRORE MQRCCF_PARM_SEQUENCE	3035	X'00000BDB'
MQRCCF_XMIT_PROTOCOL_TYPE_ERR	3036	X'00000BDC'
ERRORE MQRCCF_BATCH_SIZ	3037	X'00000BDD'
ERRORE MQRCCF_DISC_INT_	3038	X'00000BDE'
ERRORE RETRA_MQRCCF_SHORT_	3039	X'00000BDF'
ERRORE MQRCCF_SHORT_TIMER	3040	X'00000BE0'
ERRORE MQRCCF_LONG_RETRY_ERROR	3041	X'00000BE1'
MQRCCF_LONG_TIMER_ERROR	3042	X'00000BE2'
ERRORE MQRCCF_SEQ_NUMBER_WRAP	3043	X'00000BE3'

Tabella 305. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQRCCF_MAX_MSG_LENGTH_ERROR	3044	X'00000BE4'
ERRORE UT_AUTORE MQRCCF_PUT_	3045	X'00000BE5'
ERRORE MQRCCF_PURGE_VALUE_	3046	X'00000BE6'
MQRCCF_CFIL_PARM_ID_ERROR	3047	X'00000BE7'
MQRCCF_MSG_TRUNCATED	3048	X'00000BE8'
ERRORE CCSID MQRCCF	3049	X'00000BE9'
ERRORE DI MIGRAZIONE MQRCCF_ENCODING_ERROR	3050	X'00000BEA'
ERRORE MQRCCF_QUEUES_VALUE_	3051	X'00000BEB'
ERRORE MQRCCF_DATA_CONV_VALUE_	3052	X'00000BEC'
ERRORE MQRCCF_INDOUBT_VALUE_	3053	X'00000BED'
ERRORE MQRCCF_ESCAPE_TYPE_ERROR	3054	X'00000BEE'
ERRORE MQRCCF_REPOS_VALUE_	3055	X'00000BEF'
ERRORE MQRCCF_CHANNEL_TABELLA	3062	X'00000BF6'
MQRCCF_MCA_TYPE_ERROR	3063	X'00000BF7'
ERRORE MQRCCF_CHL_INST_TYPE_ERROR	3064	X'00000BF8'
MQRCCF_CHL_STATUS_NOT_FOUND	3065	X'00000BF9'
MQRCCF_CFSL_DUPLICATE_PARM	3066	X'00000BFA'
MQRCCF_CFSL_TOTAL_LENGTH_ERROR	3067	X'00000BFB'
ERRORE MQRCCF_CFSL_COUNT	3068	X'00000BFC'
MQRCCF_CFSL_STRING_LENGTH_ERR	3069	X'00000BFD'
MQRCCF_BROKER_DELETED	3070	X'00000BFE'
ERRORE DI TRANSAZIONE MQRCCF_STREAM_ERROR	3071	X'00000BFF'
MQRCCF_TOPIC_ERROR	3072	X'00000C00'
MQRCCF_NOT_REGISTERED	3073	X'00000C01'
MQRCCF_Q_MGR_NAME_ERROR	3074	X'00000C02'
MQRCCF_FLUSSO non corretto	3075	X'00000C03'
MQRCCF_Q_NAME_ERROR	3076	X'00000C04'
MQRCCF_NO_RETAINED_MSG	3077	X'00000C05'
MQRCCF_DUPLICATE_IDENTITY	3078	X'00000C06'
MQRCCF_INCORRECT_Q	3079	X'00000C07'
MQRCCF_CORREL_ID_ERROR	3080	X'00000C08'
MQRCCF_NOT_AUTHORIZED	3081	X'00000C09'
MQRCCF_UNKNOWN_STREAM	3082	X'00000C0A'
MQRCCF_REG_OPTIONS_ERROR	3083	X'00000C0B'
MQRCCF_PUB_OPTIONS_ERROR	3084	X'00000C0C'
MQRCCF_UNKNOWN_BROKER	3085	X'00000C0D'
ERRORE MQRCCF_Q_MGR_CCSID	3086	X'00000C0E'
MQRCCF_DEL_OPTIONS_ERROR	3087	X'00000C0F'
MQRCCF_CLUSTER_NOME_CONFLITTO	3088	X'00000C10'

Tabella 305. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQRCCF_REPOS_NAME_CONFLICT	3089	X'00000C11'
MQRCCF_CLUSTER_Q_USAGE_ERROR	3090	X'00000C12'
ERRORE MQRCCF_ACTION_VALUE_	3091	X'00000C13'
ERRORE MQRCCF_COMMS_LIBRARY_ERROR	3092	X'00000C14'
ERRORE MQRCCF_NETBIOS_NAME_NET	3093	X'00000C15'
MQRCCF_BROKER_COMMAND_FAILED	3094	X'00000C16'
MQRCCF_CFST_CONFLICTING_PARM	3095	X'00000C17'
MQRCCF_PATH_NOT_VALID	3096	X'00000C18'
ERRORE MQRCCF_PARM_SYNTAX_ERROR	3097	X'00000C19'
ERRORE MQRCCF_PWD_LENGTH	3098	X'00000C1A'
MQRCCF_FILTER_ERROR	3150	X'00000C4E'
MQRCCF_WRONG_USER	3151	X'00000C4F'
MQRCCF_DUPLICATE_SUBSCRIPTION	3152	X'00000C50'
MQRCCF_SUB_NAME_ERROR	3153	X'00000C51'
MQRCCF_SUB_IDENTITY_ERROR	3154	X'00000C52'
MQRCCF_SUBSCRIPTION_IN_USE	3155	X'00000C53'
MQRCCF_SUBSCRIPTION_LOCKED	3156	X'00000C54'
MQRCCF_ALREADY_JOINED	3157	X'00000C55'
MQRCCF_OBJECT_IN_USE	3160	X'00000C58'
MQRCCF_UNKNOWN_FILE_NAME	3161	X'00000C59'
MQRCCF_FILE_NON_DISPONIBILE	3162	X'00000C5A'
ERRORE RETRA_MQRCCF_DISC	3163	X'00000C5B'
MQRCCF_ALLOC_RETRY_ERROR	3164	X'00000C5C'
MQRCCF_ALLOC_SLOW_TIMER_ERROR	3165	X'00000C5D'
ERRORE MQRCCF_ALLOC_FAST_TIMER_ERROR	3166	X'00000C5E'
MQRCCF_PORT_NUMBER_ERROR	3167	X'00000C5F'
MQRCCF_CHL_SYSTEM_NOT_ATTIVO	3168	X'00000C60'
MQRCCF_ENTITY_NAME_MISSING	3169	X'00000C61'
ERRORE NAME_PROFILE MQRCCF_	3170	X'00000C62'
ERRORE MQRCCF_AUTH_VALUE_	3171	X'00000C63'
MQRCCF_AUTH_VALUE_XX_ENCODE_CASE_ONE mancante	3172	X'00000C64'
MQRCCF_OBJECT_TYPE_MISSING	3173	X'00000C65'
ERRORE MQRCCF_CONNECTION_ID_ERROR	3174	X'00000C66'
ERRORE TIPO_LOG_MQRCCF	3175	X'00000C67'
MQRCCF_PROGRAMMA_NON disponibile	3176	X'00000C68'
MQRCCF_PROGRAM_AUTH_NON RIUSCITO	3177	X'00000C69'
MQRCCF_NON_TROVATO	3200	X'00000C80'
MQRCCF_SECURITY_SWITCH_OFF	3201	X'00000C81'
MQRCCF_SECURITY_REFRESH_FAILED	3202	X'00000C82'

Tabella 305. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
PARM_MQRCCF CONFLICT	3203	X'00000C83'
MQRCCF_COMMAND_INIBITO	3204	X'00000C84'
MQRCCF_OBJECT_BEING_DELETED	3205	X'00000C85'
MQRCCF_STORAGE_CLASS_IN_USE	3207	X'00000C87'
MQRCCF_NOME_OGGETTO_LIMITATO	3208	X'00000C88'
MQRCCF_OBJECT_LIMIT_EXCEEDED	3209	X'00000C89'
MQRCCF_OBJECT_OPEN_FORCE	3210	X'00000C8A'
MQRCCF_DISPOSITION_CONFLICT	3211	X'00000C8B'
MQRCCF_Q_MGR_NOT_IN_QSG	3212	X'00000C8C'
MQRCCF_ATTR_VALUE_FIXED	3213	X'00000C8D'
ERRORE MQRCCF_NAMELIST_	3215	X'00000C8F'
MQRCCF_NO_INIZIATORE_CANALE	3217	X'00000C91'
ERRORE_INIZIATORE_CANALE_MQRCCF_	3218	X'00000C92'
MQRCCF_COMMAND_LEVEL_CONFLICT	3222	X'00000C96'
MQRCCF_Q_ATTR_CONFLICT	3223	X'00000C97'
MQRCCF_EVENTS_DISABLED	3224	X'00000C98'
ERRORE MQRCCF_COMMAND_SCOPE	3225	X'00000C99'
ERRORE DI MQRCCF_COMMAND_REPLY_ERROR	3226	X'00000C9A'
MQRCCF_FUNCTION_RESTRICTED	3227	X'00000C9B'
MQRCCF_PARM_MISSING	3228	X'00000C9C'
ERRORE MQRCCF_PARM_VALUE_	3229	X'00000C9D'
ERRORE DI MQRCCF_COMMAND_LENGTH	3230	X'00000C9E'
ERRORE MQRCCF_COMMAND_ORIGIN_	3231	X'00000C9F'
MQRCCF_LISTENER_CONFLICT	3232	X'00000CA0'
MQRCCF_LISTENER_STARTED	3233	X'00000CA1'
MQRCCF_LISTENER_STOPPED	3234	X'00000CA2'
ERRORE MQRCCF_CHANNEL_ERRORE	3235	X'00000CA3'
ERRORE MQRCCF_CF_STRU	3236	X'00000CA4'
MQRCCF_UNKNOWN_USER_ID	3237	X'00000CA5'
ERRORE MQRCCF_UNEXPECTED_	3238	X'00000CA6'
MQRCCF_NO_XCF_PARTNER	3239	X'00000CA7'
MQRCCF_CFGR_PARM_ID_ERROR	3240	X'00000CA8'
ERRORE MQRCCF_CFIF_LENGTH_ERROR	3241	X'00000CA9'
ERRORE MQRCCF_CFIF_OPERAZIONE	3242	X'00000CAA'
ERRORE ID PARM_CFIF_MQRCCF	3243	X'00000CAB'
MQRCCF_CFSF_FILTER_VAL_LEN_ERR	3244	X'00000CAC'
ERRORE MQRCCF_CFSF_LENGTH_ERROR	3245	X'00000CAD'
MQRCCF_CFSF_OPERATOR_ERROR	3246	X'00000CAE'
MQRCCF_CFSF_PARM_ID_ERROR	3247	X'00000CAF'

Tabella 305. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQRCCF_TOO_MANY_FILTERS	3248	X'00000CB0'
MQRCCF_LISTENER_RUNNING	3249	X'00000CB1'
MQRCCF_LSTR_STATUS_NOT_FOUND	3250	X'00000CB2'
MQRCCF_SERVICE_RUNNING	3251	X'00000CB3'
MQRCCF_STATUS_NOT_FOUND	3252	X'00000CB4'
MQRCCF_SERVICE_STOPPED	3253	X'00000CB5'
MQRCCF_CFBS_DUPLICATE_PARM	3254	X'00000CB6'
ERRORE MQRCCF_CFBS_LENGTH	3255	X'00000CB7'
ERRORE MQRCCF_CFBS_PARM_ID_	3256	X'00000CB8'
MQRCCF_CFBS_STRING_LENGTH_ERR	3257	X'00000CB9'
MQRCCF_CFGR_LENGTH_ERROR	3258	X'00000CBA'
ERRORE CONTEGGIO FILE MQRCCF_CFGR	3259	X'00000CBB'
MQRCCF_CONN_NOT_STOPPED	3260	X'00000CBC'
MQRCCF_RICHIESTA_SERVIZIO_IN sospeso	3261	X'00000CBD'
MQRCCF_NO_START_CMD	3262	X'00000CBE'
MQRCCF_NO_STOP_CMD	3263	X'00000CBF'
ERRORE MQRCCF_CFBF_LENGTH_ERROR	3264	X'00000CC0'
MQRCCF_CFBF_PARM_ID_ERROR	3265	X'00000CC1'
ERRORE MQRCCF_CFBF_OPERAZIONE	3266	X'00000CC2'
ERRCCF_CFBF_FILTER_VAL_LEN_R MQRCCF_CFBF_FILTER_VAL_ERR	3267	X'00000CC3'
MQRCCF_LISTENER_STILL_ACTIVE	3268	X'00000CC4'
ERRORE CODA MQRCCF_DEF_XMIT_Q_CLUS_ERROR	3269	X'0000CC5'
MQRCCF_TOPICSTR_ALREADY_EXISTS	3300	X'00000CE4'
ERRORE MQRCCF_SHARING_CONVS_	3301	X'00000CE5'
TIPO_CONVS_SHARING_MQRCCF	3302	X'00000CE6'
MQRCCF_SECURITY_CASE_CONFLICT	3303	X'00000CE7'
ERRORE TIPO_MQRCCF_TOPIC_	3305	X'00000CE9'
ERRORE MQRCCF_MAX_INSTANCE_	3306	X'00000CEA'
ERR CLNT MQRCCF_MAX_INSTS_PER_ERR	3307	X'00000CEB'
MQRCCF_TOPIC_STRING_NOT_FOUND	3308	X'00000CEC'
MQRCCF_SUBSCRIPTION_POINT_ERR	3309	X'00000CED'
MQRCCF_SUB_ALREAD_EXISTS	3311	X'00000CEF'
MQRCCF_UNKNOWN_OBJECT_NAME	3312	X'00000CF0'
ERRORE NOME CODA MQRCCF_REMOTE_	3313	X'00000CF1'
MQRCCF_DURABILITY_NON_CONSENTITO	3314	X'00000CF2'
ERRORE HOBJ_MQRCCF	3315	X'00000CF3'
ERRORE NAME_DEST_MQRCCF	3316	X'00000CF4'
MQRCCF_DESTINAZIONE_NONVALIDA	3317	X'00000CF5'
MQRCCF_PUBSUB_INIBITO	3318	X'00000CF6'

Tabella 305. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQRCCF_CHLAUTH_TYPE_ERROR	3326	X'00000CFE'
ERRORE MQRCCF_CHLAUTH_ACTION_ERROR	3327	X'00000CFF'
ERRORE MQRCCF_CHLAUTH_USERSRC_	3335	X'00000D07'
TIPO_CHLAUTO_WRONG_MQRCCF	3336	X'00000D08'
MQRCCF_CHLAUTH_ALREAD_EXISTS	3337	X'00000D09'
MQRCCF_CHLAUTH_NOT_FOUND	3338	X'00000D0A'
MQRCCF_WRONG_CHLAUTH_ACTION	3339	X'00000D0B'
MQRCCF_ERRATO_CHLAUTH_USERSRC	3340	X'00000D0C'
ERRORE MQRCCF_CHLAUTH_WARN_ERROR	3341	X'00000D0D'
MQRCCF_MISG_CHLAUTH_MATCH	3342	X'00000D0E'
MQRCCF_IPADDR_RANGE_CONFLICT	3343	X'00000D0F'
MQRCCF_CHLAUTH_MAX_EXCEEDED	3344	X'00000D10'
ERRORE IPADDR_MQRCCF	3345	X'00000D11'
ERRORE MQRCCF_IPADDR_RANGE_ERROR	3346	X'00000D12'
MQRCCF_NOME_PROFILO_MANCANTE	3347	X'00000D13'
ERRORE MQRCCF_CHLAUTH_CLNTUSER_ERROR	3348	X'00000D14'
ERRORE MQRCCF_CHLAUTH_NAME_ERROR	3349	X'00000D15'
ERRORE MQRCCF_SUITE_B_ERROR	3353	X'00000D19'
MQRCCF_PSCLUS_DISABLED_TOPDEF	3359	X'00000D1F'
EXISTS MQRCCF_PSCLUS_TOPIC_	3360	X'00000D20'
PROTOCOLLO MQRCCF_INVALID_PROTOCOL	3365	X'00000D25'
MQRCCF_ACCESS_BLOCKED	3382	X'00000D36'
MQRCCF_OBJECT_ALREADY_EXISTS	4001	X'00000FA1'
MQRCCF_OBJECT_WRONG_TIPO	4002	X'00000FA2'
MQRCCF_LIKE_OBJECT_WRONG_TYPE	4003	X'00000FA3'
MQRCCF_OBJECT_OPEN	4004	X'00000FA4'
ERRORE MQRCCF_ATTR_VALUE_	4005	X'00000FA5'
MQRCCF_UNKNOWN_Q_MGR	4006	X'00000FA6'
Q_WRONG_MQRCCF_TIPO	4007	X'00000FA7'
MQRCCF_OBJECT_NAME_ERROR	4008	X'00000FA8'
MQRCCF_ALLOCATE_NON RIUSCITO	4009	X'00000FA9'
MQRCCF_HOST_NON_DISPONIBILE	4010	X'00000FAA'
ERRORE MQRCCF_CONFIGURATION_ERROR	4011	X'00000FAB'
MQRCCF_CONNECTION_RIFIUTATO	4012	X'00000FAC'
ERRORE MQRCCF_ENTRY_ERROR	4013	X'00000FAD'
MQRCCF_SEND_NON RIUSCITO	4014	X'00000FAE'
ERRORE MQRCCF_RECEIVED_DATA_	4015	X'00000FAF'
MQRCCF_RECEIVE_NON RIUSCITO	4016	X'00000FB0'
MQRCCF_CONNECTION_CLOSED	4017	X'00000FB1'

Tabella 305. Valori delle costanti (Continua)

Nome	Valore decimale	Valore esadecimale
MQRCCF_NO_STORAGE	4018	X'00000FB2'
MQRCCF_NO_COMMS_MANAGER	4019	X'00000FB3'
MQRCCF_LISTENER_NOT_STARTED	4020	X'00000FB4'
MQRCCF_BIND_NON RIUSCITO	4024	X'00000FB8'
MQRCCF_CHANNEL_INDOUBT	4025	X'00000FB9'
MQRCCF_MQCONN_NON RIUSCITO	4026	X'00000FBA'
MQRCCF_MQOPEN_NON RIUSCITO	4027	X'00000FBB'
MQRCCF_MQGET_FAILED	4028	X'00000FBC'
MQRCCF_MQPUT_NON RIUSCITO	4029	X'00000FBD'
ERRORE PING_MQRCCF	4030	X'00000FBE'
MQRCCF_CHANNEL_IN_USE	4031	X'00000FBF'
MQRCCF_CHANNEL_NOT_FOUND	4032	X'00000FC0'
MQRCCF_UNKNOWN_REMOTE_CHANNEL	4033	X'00000FC1'
MQRCCF_REMOTE_QM_UNAVAILABLE	4034	X'00000FC2'
MQRCCF_REMOTE_QM_TERMINATING	4035	X'00000FC3'
MQRCCF_MQINQ_NON RIUSCITO	4036	X'00000FC4'
MQRCCF_NOT_XMIT_Q	4037	X'00000FC5'
MQRCCF_CHANNEL_DISABLED	4038	X'00000FC6'
MQRCCF_USER_EXIT_NON_DISPONIBILE	4039	X'00000FC7'
MQRCCF_COMMIT_NON RIUSCITO	4040	X'00000FC8'
TIPO_WRONG_CHANNEL_MQRCCF	4041	X'00000FC9'
MQRCCF_CHANNEL_ALREADY_EXISTS	4042	X'00000FCA'
MQRCCF_DATA_TOO_LARGE	4043	X'00000FCB'
ERRORE MQRCCF_CHANNEL_NAME_ERROR	4044	X'00000FCC'
MQRCCF_XMIT_Q_NAME_ERROR	4045	X'00000FCD'
ERRORE NAME_MCA_MQRCCF_	4047	X'00000FCF'
MQRCCF_SEND_EXIT_NAME_ERROR	4048	X'00000FD0'
ERRORE MQRCCF_SEC_EXIT_NAME_ERROR	4049	X'00000FD1'
ERRORE NAME_EXIT_MSG_MQRCCF_	4050	X'00000FD2'
ERRORE MQRCCF_RCV_EXIT_NAME_ERROR	4051	X'00000FD3'
MQRCCF_XMIT_Q_NAME_WRONG_TYPE	4052	X'00000FD4'
MQRCCF_MCA_NAME_WRONG_TYPE	4053	X'00000FD5'
TIPO_DISC_MQRCCF_INT_WRONG_	4054	X'00000FD6'
MQRCCF_SHORT_RETRY_WRONG_TIPO	4055	X'00000FD7'
MQRCCF_SHORT_TIMER_GIUSTAMENTE	4056	X'00000FD8'
MQRCCF_LONG_RETRY_WRONG_TIPO	4057	X'00000FD9'
MQRCCF_LONG_TIMER_TIPO_ERRATO	4058	X'00000FDA'
TIPO_UT_AUTH_WRONG_MQRCCF_	4059	X'00000FDB'
ERRORE MQRCCF_KEEP_ALIVE_INT_ERROR	4060	X'00000FDC'

Tabella 305. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQRCCF_MISSING_CONN_NAME	4061	X'00000FDD'
ERRORE NAME_CONN_MQRCCF_	4062	X'00000FDE'
MQRCCF_MQSET_NON RIUSCITO	4063	X'00000FDF'
MQRCCF_CHANNEL_NOT_ATTIVO	4064	X'00000FE0'
MQRCCF_TERMINATED_BY_SEC_EXIT	4065	X'00000FE1'
MQRCCF_DYNAMIC_Q_SCOPE_ERROR	4067	X'00000FE3'
MQRCCF_CELL_DIR_NON_DISPONIBILE	4068	X'00000FE4'
ERRORE MQRCCF_MR_COUNT_	4069	X'00000FE5'
MMR_COUNT_WRONG_TYPE MQRCCF_	4070	X'00000FE6'
ERRORE NAME_MR_MQRCCF_EXIT_	4071	X'00000FE7'
TIPO_WRONG_MR_EXIT_NAME_MQRCCF_	4072	X'00000FE8'
ERRORE INTERNO MQRCCF_MR	4073	X'00000FE9'
TIPO_WRONG_MR_MQRCCF_INTERVAL_	4074	X'00000FEA'
ERRORE MQRCCF_NPM_SPE	4075	X'00000FEB'
TIPO_MQRCCF_NPM_SPEED_WRONG_	4076	X'00000FEC'
ERRORE INTERVAL_HB_MQRCCF_	4077	X'00000FED'
MQRCCF_HB_INTERVAL_WRONG_TIPO	4078	X'00000FEE'
ERRORE MQRCCF_CHAD	4079	X'00000FEF'
TIPO_WRONG_CHAD_MQRCCF	4080	X'00000FF0'
ERRORE MQRCCF_CHAD_EVENT_	4081	X'00000FF1'
TIPO_CHAD_MQRCCF_EVENT_WRONG_	4082	X'00000FF2'
ERRORE MQRCCF_CHAD_EXIT_	4083	X'00000FF3'
TIPO_CHAD_MQRCCF_EXIT_WRONG_	4084	X'00000FF4'
MQRCCF_SUPPRESSED_BY_EXIT	4085	X'00000FF5'
ERRORE MQRCCF_BATCH_INT_	4086	X'00000FF6'
TIPO_WRONG_BATCH_MQRCCF	4087	X'00000FF7'
ERRORE DI PREZZO MQRCCF_NET_NET	4088	X'00000FF8'
MQRCCF_NET_PRIORITY_WRONG_TIPO	4089	X'00000FF9'
MQRCCF_CHANNEL_CHIUSO	4090	X'00000FFA'
MQRCCF_Q_STATUS_NOT_FOUND	4091	X'00000FFB'
MQRCCF_SSL_CIPHER_SPEC_ERROR	4092	X'00000FFC'
ERRORE MQRCCF_SSL_PEER_NAME_ERROR	4093	X'00000FFD'
MQRCCF_SSL_CLIENT_AUTH_ERROR	4094	X'00000FFE'
MQRCCF_RETAINED_NOT_SUPORTED	4095	X'00000FFF'
z/OS MQRCCF_KWD_VALUE_XX_ENCODE_CASE_ONE tipo_scrittura	4096	X'00001000'

MQRCN_ * (Costanti di riconnessione client)

Tabella 306. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQRCN_NO	0	X'00000000'
MQRCN_YES	1	X'00000001'
MQRCN_Q_MGR	2	X'00000002'
MQRCN_DISABLED	3	X'00000003'

MQRCVTIME_ * (Tipi timeout di ricezione)

Tabella 307. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQRCVTIME_MULTIPLY	0	X'00000000'
MQRCVTIME_ADD	1	X'00000001'
MQRCVTIME_EQUAL	2	X'00000002'

MQREADA_ * (Leggi valori in testa)

Tabella 308. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQREADA_NO	0	X'00000000'
SÌ MQREADA_	1	X'00000001'
DISABILITA_MQREAD_LED	2	X'00000002'
MQREADA_INIBITO	3	X'00000003'
READA_BACKLOG	4	X'00000004'

MQRECORDING_ * (Opzioni di registrazione)

Tabella 309. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQRECORDING_DISABILITATO	0	X'00000000'
MQRECORDING_Q	1	X'00000001'
MQRECORDING_MSG	2	X'00000002'

MQREGO_ * (Opzioni di registrazione pubblicazione / sottoscrizione)

Tabella 310. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQREGO_NONE	0	X'00000000'
MQREGO_CORREL_ID_AS_IDENTITY	1	X'00000001'
MQREGO_ANONIMO	2	X'00000002'
LOCALE MQREGO	4	X'00000004'
MQREGO_DIRECT_REQUESTS	8	X'00000008'
MQREGO_NUOVA_PUBBLICAZIONI_SOLO	16	X'00000010'
MQREGO_PUBLISH_ON_REQUEST_ONLY	32	X'00000020'

Tabella 310. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQREGO_DEREGISTER_ALL	64	X'00000040'
MQREGO_INCLUDE_NOME_FLUSSO	128	X'00000080'
MQREGO_INFORM_IF_CONSERVATI	256	X'00000100'
MQREGO_DUPLICATES_OK	512	X'00000200'
MQREGO_NON_PERSISTENTE	1024	X'00000400'
PERSISTORA_MQREGO	2048	X'00000800'
MQREGO_PERSISTENT_AS_PUBLISH	4096	X'00001000'
MQREGO_PERSIST_AS_Q	8192	X'00002000'
NOME_INDIRIZZO_MQREGO	16384	X'00004000'
MQREGO_NO_ALTERAZIONE	32768	X'00008000'
MQREGO_FULL_XX_ENCODE_CASE_ONE risposta	65536	X'00010000'
MQREGO_JOIN_SHARED	131072	X'00020000'
MQREGO_JOIN_EXCLUSIVE	262144	X'00040000'
SOLO MQREGO_LEAVE_	524288	X'00080000'
ID_UTENTE_VARIABILE_MQREGO_	1048576	X'00100000'
LOCKED MQREGO	2097152	X'00200000'

MQRFH_ * (Regole e formattazione struttura intestazione e indicatori)

Regole e struttura intestazione di formattazione

Tabella 311. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQRFH	"RFH¬"
ID_STRUC_MQRFH_ARRAY	'R','F','H','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 312. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQRFH_VERSION_1	1	X'00000001'
MQRFH_VERSION_2	2	X'00000002'
MQRFH_STRUC_LENGTH_FIXED	32	X'00000020'
MQRFH_STRUC_LENGTH_FIXED_2	36	X'00000024'

Regole e intestazione di formattazione

Tabella 313. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQRFH_NONE	0	X'00000000'
MQRFH_NO_FLAGS	0	X'00000000'

MQRFH2_* (Tag delle opzioni di pubblicazione / sottoscrizione RFH2 Tag della cartella di livello superiore)

Tabella 314. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQRFH2_NAME_VALUE_VERSION	1	X'00000001'

MQRFH2_* (Nome tag tag opzioni di pubblicazione / sottoscrizione)

MQRFH2_PUBSUB_CMD_FOLDER	"psc"
MQRFH2_PUBSUB_RESP_FOLDER	"pscr"
MQRFH2_MSG_CONTENT_FOLDER	"mcd"
MQRFH2_USER_FOLDER	"usr"

MQRFH2_* (Nome tag XML tag opzioni di pubblicazione / sottoscrizione)

MQRFH2_PUBSUB_CMD_FOLDER_B	"<psc>"
MQRFH2_PUBSUB_CMD_FOLDER_E	"</psc>"
MQRFH2_PUBSUB_RESP_FOLDER_B	"<pscr>"
MQRFH2_PUBSUB_RESP_FOLDER_E	"</pscr>"
MQRFH2_MSG_CONTENT_FOLDER_B	"<mcd>"
MQRFH2_MSG_CONTENT_FOLDER_E	"</mcd>"
MQRFH2_USER_FOLDER_B	"<usr>"
MQRFH2_USER_FOLDER_E	"</usr>"

MQRL_ * (Lunghezza restituita)

Tabella 315. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQRL_UNDEFINED	-1	X'FFFFFFFF'

MQRMH_ * (Struttura intestazione messaggio di riferimento)

Tabella 316. Strutture di costanti

Nome	Struttura
ID_STRUC_MQRMH	"RMH¬"
ID_STRUC_MQRMH_ARRAY	'R','M','H','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 317. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQRMH_VERSION_1	1	X'00000001'
VERSIONE MQRMH_CURRENT_	1	X'00000001'

MQRMHF_ * (Indicatori intestazione messaggio di riferimento)

Tabella 318. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQRMHF_LAST	1	X'00000001'
MQRMHF_NO_LAST	0	X'00000000'

MQRO_ * (Opzioni report)

Tabella 319. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQRO_ECCEZIONE	16777216	X'01000000'
MQRO_EXCEPTION_WITH_DATA	50331648	X'03000000'
MQRO_EXCEPTION_WITH_FULL_DATA	117440512	X'07000000'
SCADENZA_MQRO	2097152	X'00200000'
MQRO_EXPIRATION_WITH_DATA	6291456	X'00600000'
MQRO_EXPIRATION_WITH_FULL_DATA	14680064	X'00E00000'
COA MQRO	256	X'00000100'
DATA_COA_WITH_MQRO	768	X'00000300'
DATI_COA_WITH_MQRO_FULL_DATA	1792	X'00000700'
COD MQRO	2048	X'00000800'
DATI MQRO_COD_WITH_	6144	X'00001800'
DAD_COD MQRO_WITH_FULL_DATA	14336	X'00003800'
MQRO_PAN	1	X'00000001'
MQRO_NAN	2	X'00000002'
ATTIVITÀ MQRO	4	X'00000004'
ID_MSG_NEW_MQRO	0	X'00000000'
MQRO_PASS_MSG_ID	128	X'00000080'
ID_COPY_MQRO_MSG_TO_CORREL_ID	0	X'00000000'
ID CORREL_PASS_MQRO_	64	X'00000040'
MQRO_DEAD_LETTER_Q	0	X'00000000'
MQRO_DISCARD_MSG	134217728	X'08000000'
MQRO_PASS_DISCARD_E_SCADENZA	16384	X'00004000'
MQRO_NONE	0	X'00000000'

*MQRO_ * (Maschere opzioni report)*

Tabella 320. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQRO_REJECT_UNSUP_MASK	270270464	X'101C0000'
MQRO_ACCEPT_UNSUP_MASK	-270532353	X'EFE000FF'
MQRO_ACCEPT_UNSUP_IF_XMIT_MASK	261888	X'0003FF00'

MQROUTE_ * (Trace-route)

Numero massimo di attività di traccia - instradamento (MQIACF_MAX_ACTIVITIES)

Tabella 321. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQROUTE_UNLIMITED_ACTIVITIES	0	X'00000000'

Dettagli tracerouta (MQIACF_ROUTE_DETAIL)

Tabella 322. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQROUTE_DETAIL_LOW	2	X'00000002'
MQROUTE_DETAIL_MEDIO	8	X'00000008'
MQROUTE_DETAIL_HIGH	32	X'00000020'

Inoltro traceroute (MQIACF_ROUTE_FORWARDING)

Tabella 323. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQROUTE_FORWARD_ALL	256	X'00000100'
MQROUTE_FORWARD_IF_SUPPORTED	512	X'00000200'
MQROUTE_FORWARD_REJ_UNSUP_MASK	-65536	X'FFFF0000'

Distribuzione traccia - instradamento (MQIACF_ROUTE_DELIVERY)

Tabella 324. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQROUTE_DELIVER_SÌ	4096	X'00001000'
MQROUTE_DELIVER_NO	8192	X'00002000'
MQROUTE_DELIVER_REJ_UNSUP_MASK	-65536	X'FFFF0000'

Accumulo instradamento traccia (MQIACF_ROUTE_ACCUMULO)

Tabella 325. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQROUTE_ACCUMULATE_NONE	65539	X'00010003'
MQROUTE_ACCUMULATE_IN_MSG	65540	X'00010004'
MQROUTE_ACCUMULATE_E_REPLY	65541	X'00010005'

MQRP_ * (Opzioni di sostituzione formato comando)

Tabella 326. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
SÌ MQRP	1	X'00000001'
MQRP_NO	0	X'00000000'

MQRQ_ * (Qualificatori motivo formato comando)

Tabella 327. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQRQ_CONN_NOT_AUTHORIZED	1	X'00000001'
MQRQ_OPEN_NON_AUTORIZZATO	2	X'00000002'
MQRQ_CLOSE_NOT_AUTHORIZED	3	X'00000003'
MQRQ_CMD_NOT_AUTHORIZED	4	X'00000004'
MQRQ_Q_MGR_STOPPING	5	X'00000005'
MQRQ_Q_MGR_QUIESCING	6	X'00000006'
MQRQ_CHANNEL_STOPPED_OK	7	X'00000007'
MQRQ_CHANNEL_STOPPED_ERROR	8	X'00000008'
MQRQ_CHANNEL_STOPPED_RETRY	9	X'00000009'
MQRQ_CHANNEL_STOPPED_DISABLED	10	X'0000000A'
MQRQ_BRIDGE_ARRESTATO_OK	11	X'0000000B'
ERRORE_ARRESTATO_BRIDGE_MQRQ_	12	X'0000000C'
MQRQ_SSL_HANDSHAKE_ERROR	13	X'0000000D'
MQRQ_SSL_CIPHER_SPEC_ERROR	14	X'0000000E'
ERRORE MQRQ_SSL_CLIENT_AUTH_ERROR	15	X'0000000F'
ERRORE MQRQ_SSL_PEER_NAME_ERROR	16	X'00000010'
MQRQ_SUB_NOT_AUTHORIZED	17	X'00000011'
MQRQ_SUB_DEST_NON_AUTORIZZATO	18	X'00000012'
MQRQ_SSL_UNKNOWN_REVOCATION	19	X'00000013'
MQRQ_SYS_CONN_NOT_AUTHORIZED	20	X'00000014'
INDIRIZZO_BLOCCO_CANALE_MQRQ	21	X'00000015'
MQRQ_CHANNEL_BLOCKED_USERID	22	X'00000016'
MQRQ_CHANNEL_BLOCKED_NOACCESS	23	X'00000017'
MQRQ_MAX_ACTIVE_CHANNELS	24	X'00000018'
MQRQ_MAX_CHANNELS	25	X'00000019'
MQRQ_SVRCONN_INST_LIMIT	26	X'0000001A'
MQRQ_CLIENT_INST_LIMIT!	27	X'0000001B'
MQRQ_CAF_NOT_INSTALLED	28	X'0000001C'
MQRQ_CSP_NOT_AUTHORIZED	29	X'0000001D'
MQRQ_FAILOVER_CONSENTITO	30	X'0000001E'
MQRQ_FAILOVER_NON consentito	31	X'0000001F'
MQRQ_STANDBY_ATTIVATA	32	X'00000020'
MQRQ_REPLICA_ATTIVATO	33	X'00000021'

MQRT_ * (Formato del comando Tipi di aggiornamento)

Tabella 328. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
CONFIGURAZIONE_MQRT	1	X'00000001'

Tabella 328. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
SCADENZIA_MQRT	2	X'00000002'
NSPROC MQRT	3	X'00000003'
PROXYSUB MQRT	4	X'00000004'

MQRU_ * (Solo richiesta)

Tabella 329. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQRU_PUBLISH_ON_REQUEST	1	X'00000001'
MQRU_PUBLISH_ALL	2	X'00000002'

MQSCA_ * (autenticazione client TLS)

Tabella 330. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSCA_XX_ENCODE_CASE_ONE obbligatorio	0	X'00000000'
MQSCA_XX_ENCODE_CASE_ONE facoltativo	1	X'00000001'

MQSCO_ * (opzioni di configurazione TLS)

Struttura delle opzioni di configurazione TLS

Tabella 331. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQSCO	"SCO¬"
ARRAY MQSCO_STRUC_ID_ARRAY	'S','C','O','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 332. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSCO_VERSION_1	1	X'00000001'
MQSCO_VERSION_2	2	X'00000002'
MQSCO_VERSION_3	3	X'00000003'
MQSCO_VERSION_4	4	X'00000004'
VERSIONE MQSCO_CURRENT_	4	X'00000004'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Conteggio reimpostazioni chiave opzioni di configurazione TLS

Tabella 333. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSCO_RESET_COUNT_DEFAULT	0	X'00000000'

Formato comando Ambito definizione coda

Tabella 334. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MGR coda MQSCO	1	X'00000001'
CCELL MQSCO	2	X'00000002'

MQSCOPE_ * (ambito di pubblicazione)

Tabella 335. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSCOPE_ALL	0	X'00000000'
MQSCOPE_AS_PARENT	1	X'00000001'
MQSCOPE_QMGR	4	X'00000004'

MQSCYC_ * (Caso di protezione)

Tabella 336. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSCYC_UPPER	0	X'00000000'
MIXED MQSCY	1	X'00000001'

MQSD_ * (Struttura descrittore oggetto)

Tabella 337. Nomi e strutture costanti	
Nome	**Struttura**
ID_STRUC_MQSD	"SD¬¬"
MQSD_STRUC_ID_ARRAY	'S','D','¬','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 338. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSD_VERSION_1	1	X'00000001'
VERSIONE MQSD_CURRENT_	1	X'00000001'

MQSECITEM_ * (Elementi di sicurezza in formato comando)

Tabella 339. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSECITEM_ALL	0	X'00000000'
MQSECITEM_MQADMIN	1	X'00000001'
MQSECITEM_MQNLIST	2	X'00000002'
MQSECITEM_MQPROC	3	X'00000003'
MQSECITEM_MQQUEUE	4	X'00000004'
MQSECITEM_MQCONN	5	X'00000005'
MQSECITEM_MQCMDS	6	X'00000006'
MQSECITEM_MXADMIN	7	X'00000007'

Tabella 339. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQSECITEM_MXNLIST	8	X'00000008'
MQSECITEM_MXPROC	9	X'00000009'
MQSECITEM_MXQUEUE	10	X'0000000A'
MQSECITEM_MXTOPIC	11	X'0000000B'

MQSECPROT_ * (Tipi di protocollo di sicurezza)

Tabella 340. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSECPROT_NONE	0	X'00000000'
MQSECPROT_SSLV30	1	X'00000001'
MQSECPROT_TLSV10	2	X'00000002'
MQSECPROT_TLSV12	4	X'00000004'

MQSECSW_ * (Formato del comando Switch di sicurezza e Stati switch)

Switch di sicurezza formato comando

Tabella 341. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
PROCESSO MQSECSO	1	X'00000001'
ELENCO NOMI MQSECSW_NAMELIST	2	X'00000002'
MQSECSW_Q	3	X'00000003'
TOPIC MQSECSW_	4	X'00000004'
CONTEXT MQSECSW_	6	X'00000006'
MQSECSW_ALTERNATE_UTENTE	7	X'00000007'
COMANDO MQSECSW_AND	8	X'00000008'
MQSEC_CONNECZIONE	9	X'00000009'
SISTEMA MQSECSW_SUBSYSTEM	10	X'0000000A'
MQSECSW_COMMAND_RESOURCES	11	X'0000000B'
MQSECSW_Q_MGR	15	X'0000000F'
MQSECSW_QSG	16	X'00000010'

Formato del comando Security Switch States

Tabella 342. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSECSW_OFF_FOUND	21	X'00000015'
FOUND MQSECSW_ON	22	X'00000016'
MQSECSW_OFF_NON_TROVATO	23	X'00000017'
MQSECSW_ON_NO_FOUND	24	X'00000018'
ERRORE MQSECSW_OFF	25	X'00000019'
MQSECSW_ON_OVERRIDDEN	26	X'0000001A'

MQSECTYPE_ * (Formato del comando Tipi di sicurezza)

Tabella 343. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQSECTYPE_AUTSERV	1	X'00000001'
SSL MQSECTYPE	2	X'00000002'
CLASSE_MQSECTYPE	3	X'00000003'

MQSEG_ * (Segmentazione)

Tabella 344. Nomi e valori costanti

Nome	Valore
MQSEG_INIBITO	'¬'
MQSEG_CONSENTITO	'A'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

MQSEL_ * (Valori selettori speciali)

Tabella 345. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQSEL_ANY_SELECTOR	-30001	X'FFFF8ACF'
MQSEL_ANY_USER_SELECTOR	-30002	X'FFFF8ACE'
MQSEL_ANY_SYSTEM_SELECTOR	-30003	X'FFFF8ACD'
MQSEL_ALL_SELECTORS	-30001	X'FFFF8ACF'
MQSEL_ALL_USER_SELECTORS	-30002	X'FFFF8ACE'
MQSEL_ALL_SYSTEM_SELECTORS	-30003	X'FFFF8ACD'

MQSELTYPE_ * (Tipi di selettore)

Tabella 346. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQSELTYPE_NONE	0	X'00000000'
MQSELTYPE_STANDARD	1	X'00000001'
MQSELTYPE_EXTENDED	2	X'00000002'

MQSID_ * (Identificatore sicurezza)

Tabella 347. Nomi e valori costanti

Nome	Valore
MQSID_NONE	X'00...00' (40 valori null)
MQSID_NON_ARRAY	'\0','\0',... (40 valori null)

MQSIDT_ * (Tipi di identificativo di sicurezza)

Tabella 348. Nomi e valori costanti

Nome	Valore esadecimale
MQSIDT_NONE	X'00'

Tabella 348. Nomi e valori costanti (Continua)	
Nome	**Valore esadecimale**
ID SICUREZZA MQSIDT_NT_SECURITY_ID	X'01'
ID SICUREZZA MQSIDT_WAS_SECURITY_ID	X'02'

MQSMPO_ * (Impostare le opzioni e la struttura della proprietà del messaggio)

Imposta struttura delle opzioni della proprietà del messaggio

Tabella 349. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQSMPO	"SMPO"
MATRICE MQSMPO_STRUC_ID_ARRAY	'S','M','P','O'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 350. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSMPO_VERSION_1	1	X'00000001'
VERSIONE MQSMPO_CURRENT_	1	X'00000001'

Imposta opzioni proprietà messaggio

Tabella 351. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSMPO_SET_FIRST	0	X'00000000'
MQSMPO_SET_PROP_UNDER_CURSOR	1	X'00000001'
MQSMPO_SET_PROP_AFTER_CURSOR	2	X'00000002'
MQSMPO_APPEND_PROPERTY	4	X'00000004'
MQSMPO_SET_PROP_BEFORE_CURSOR	8	X'00000008'
MQSMPO_NONE	0	X'00000000'

MQSO_ * (Opzioni di sottoscrizione)

Tabella 352. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSO_NONE	0	X'00000000'
MQSO_NON_DURABLE	0	X'00000000'
MQSO_READ_AHEAD_AS_Q_DEF	0	X'00000000'
ALTER MQSO	1	X'00000001'
CREA_MQSO	2	X'00000002'
RESUME MQSO	4	X'00000004'
MQSO_DURATA	8	X'00000008'
SUB_GROUP_MQSO	16	X'00000010'
MQSO_MANAGED	32	X'00000020'

Tabella 352. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQSO_SET_IDENTITY_CONTEXT	64	X'00000040'
IDUSER_FIX_MQSO	256	X'00000100'
IDER_ANY_MQSO	512	X'00000200'
MQSO_PUBLICATIONS_ON_REQUEST	2048	X'00000800'
MQSO_NUOVA_PUBBLICAZIONI_SOLO	4096	X'00001000'
MQSO_FAIL_IF_QUIESCING	8192	X'00002000'
MQSO_ALTERNATE_USER_AUTHORITY	262144	X'00040000'
MQSO_WILDCARD_CHAR	1048576	X'00100000'
MQSO_WILDCARD_TOPIC	2097152	X'00200000'
ID_CORREL_SET_MQSO	4194304	X'00400000'
MQSO_SCOPE_QMGR	67108864	X'04000000'
MQSO_NO_READ_AHEAD	134217728	X'08000000'
MQSO_READ_AHEAD	268435456	X'10000000'

MQSP_ * (Disponibilità punto di sincronizzazione)

Tabella 353. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSP_DISPONIBILE	1	X'00000001'
MQSP_NON_DISPONIBILE	0	X'00000000'

MQSPL_ * (Opzioni di protezione della politica di sicurezza)

Tabella 354. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
PASSTHRU MQSPL	0	X'00000000'
MQSP_REMOVE	1	X'00000001'
MQSPL_AS_POLICY	2	X'00000002'

MQSQQM_ * (Nome gestore code code condiviso)

Tabella 355. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSQQM_USO	0	X'00000000'
MQSQQM_IGNORE	1	X'00000001'

MQSR_ * (Azione)

Tabella 356. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
PUBLICAZIONE MQSR_ACTION_	1	X'00000001'

MQSRO_ * (Struttura opzioni richiesta sottoscrizione)

Tabella 357. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQSRO	"SRO¬"
MATRICE MQSRO_STRUC_ID_ARRAY	'S','R','O','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 358. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSRO_VERSION_1	1	X'00000001'
VERSIONE MQSRO_CURRENT_	1	X'00000001'
MQSRO_NONE	0	X'00000000'
MQSRO_FAIL_IF_QUIESCING	8192	X'00002000'

MQSS_ * (Stato segmento)

Tabella 359. Nomi e strutture costanti	
Nome	**Struttura**
MQSS_NOT_A_SEGMENT	'¬'
ISCRIZIONE MQSS_SEGMENT	'S'
ISCRIZIONE MQSS_LAST_SEGMENT	'L'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

MQSSL_ * (Requisiti FIPS TLS)

Nota: Su AIX, Linux, and Windows, IBM MQ fornisce la conformità FIPS 140-2 tramite il modulo crittografico IBM Crypto for C (ICC) . Il certificato per questo modulo è stato spostato nello stato cronologico. I clienti devono visualizzare il certificatoIBM Crypto for C (ICC) ed essere a conoscenza di eventuali consigli forniti da NIST. Un modulo FIPS 140-3 di sostituzione è attualmente in corso e il relativo stato può essere visualizzato ricercandolo in NIST CMVP modules in process list.

IBM MQ Operator 3.2.0 e l'immagine del contenitore del gestore code 9.4.0.0 sono basati su UBI 9. La conformità FIPS 140-3 è attualmente in sospeso e il suo stato può essere visualizzato ricercando "Red Hat Enterprise Linux 9 - OpenSSL FIPS Provider" in NIST CMVP modules in process list.

Tabella 360. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSSL_FIPS_NO	0	X'00000000'
SÌ MQSSL_FIPS	1	X'00000001'

MQSTAT_ * (Opzioni statistiche)

Tabella 361. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSTAT_TYPE_ASYNC_ERROR	0	X'00000000'
RICONNESSIONE_TIPO_MQSTAT	0	X'00000000'
ERRORE_RICONNESSIONE_TIPO_MQSTAT_	0	X'00000000'

MQSTS_ * (Struttura della struttura di notifica dello stato)

Tabella 362. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQST	"STAT"
MATRICE MQSTS_STRUC_ID_ARRAY	'S','T','A','T'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 363. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSTS_VERSION_1	1	X'00000001'
VERSIONE MQSTS_CURRENT_	1	X'00000001'

MQSUB_ * (Sottoscrizioni durevoli)

Sottoscrizioni consentite durevoli

Tabella 364. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSUB_DURABLE_AS_PARENT	0	X'00000000'
MQSUB_DURABLE_ALLOWED	1	X'00000001'
MQSUB_DURABLE_INIBITED	2	X'00000002'

Intervallo sottoscrizioni durevoli

Tabella 365. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSUB_DURABLE_ALL	-1	X'FFFFFFFF'
MQSUB_DURABLE_SÌ	1	X'00000001'
MQSUB_DURABLE_NO	2	X'00000002'

MQSUBTYPE_ * (Tipi di sottoscrizione formato comando)

Tabella 366. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
API MQSUBTYPE_	1	X'00000001'
MQSUBTYPE_ADMIN	2	X'00000002'
MQSUBTYPE_PROXY	3	X'00000003'
MQSUBTYPE_ALL	-1	X'FFFFFFFF'
MQSUBTYPE_UTENTE	-2	X'FFFFFFFE'

MQSUS_ * (Stato sospensione formato comando)

Tabella 367. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
SÌ MQS	1	X'00000001'
MQSUS_NO	0	X'00000000'

MQSVC_ * (Servizio)

Tipi di servizi

Tabella 368. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
COMANDO TYPE_MQSVC	0	X'00000000'
SERVER MQSVC_TIPO	1	X'00000001'

Controlli servizio

Tabella 369. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSVC_CONTROL_Q_MGR	0	X'00000000'
MQSVC_CONTROL_Q_MGR_START	1	X'00000001'
MQSVC_CONTROL_MANUAL	2	X'00000002'

Stato servizio

Tabella 370. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
STOPPED MQSVC_STATUS_ED	0	X'00000000'
STATO_MQSVC	1	X'00000001'
MQSVC_STATUS_RUNNING	2	X'00000002'
STOPPING MQSVC_STATUS_	3	X'00000003'
RETRYING MQSVC_STATUS_	4	X'00000004'

MQSYNCPOINT_ * (Formato del comando: valori del punto di sincronizzazione per la migrazione Pub / Sot)

Tabella 371. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSYNCPOINT_SÌ	0	X'00000000'
MQSYNCPOINT_IFPER	1	X'00000001'

MQSYSP_ * (Valori dei parametri di sistema in formato comando)

Tabella 372. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQSYSP_NO	0	X'00000000'
SÌ MQSYSP	1	X'00000001'
MQSYSP_XX_ENCODE_CASE_ONE aggiunto	2	X'00000002'
TIPO_MQSYSP_INIZIALE	10	X'0000000A'
IMPOSTA_TIPO_MQSYSP	11	X'0000000B'
COPIA MQSYSP_TYPE_LOG_COPY	12	X'0000000C'
STATO LOG TYPE_MQSYSP	13	X'0000000D'

Tabella 372. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQSYSP_TYPE_ARCHIVE_TAPE	14	X'0000000E'
BLK ALLOC_MQSYSP	20	X'00000014'
TRK_ALLOC_MQSYSP	21	X'00000015'
MQSYSP_ALLOC_CYL	22	X'00000016'
MQSYSP_STATUS_BUSY	30	X'0000001E'
MQSYSP_STATUS_PREMOUNT	31	X'0000001F'
STATO_MQSYSP_DISPONIBILE	32	X'00000020'
MQSYSP_STATUS_UNKNOWN	33	X'00000021'
MQSYSP_STATUS_ALLOC_ARCHIVE	34	X'00000022'
MQSYSP_STATUS_COPYING_BSDS	35	X'00000023'
STATO_MQSYSP_COPYING_LOG	36	X'00000024'

MQTA_ * (attributi argomento)

Caratteri jolly

Tabella 373. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQTA_BLOCK	1	X'00000001'
PASSTHRU MQTA_	2	X'00000002'

Sottoscrizioni consentite

Tabella 374. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQTA_SUB_AS_PARENT	0	X'00000000'
MQTA_SUB_INIBITO	1	X'00000001'
MQTA_SUB_CONSENTITA	2	X'00000002'

Propagazione secondaria proxy

Tabella 375. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQTA_PROXY_SUB_FORCE	1	X'00000001'
MQTA_PROXY_SUB_FIRSTUSE	2	X'00000002'

Pubblicazioni consentite

Tabella 376. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQTA_PUB_AS_PARENT	0	X'00000000'
MQTA_PUB_INIBITO	1	X'00000001'
MQTA_PUB_ALLOWED	2	X'00000002'

MQTC_ * (Controlli trigger)

Tabella 377. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQT_DISATTIVO	0	X'00000000'
MQT_ATTIVO	1	X'00000001'

MQTCPKEEP_ * (TCP Keepalive)

Tabella 378. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQTCPKEEP_NO	0	X'00000000'
SÌ MQTCPKEEP	1	X'00000001'

MQTCPSTACK_ * (tipi di stack TCP)

Tabella 379. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQTCPSTACK_SINGLE	0	X'00000000'
MQTCPSTACK_MULTIPLE	1	X'00000001'

MQTIME_ * (Formato del comando Unità di tempo)

Tabella 380. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQTIME_UNIT_MINS	0	X'00000000'
UNIT_MQTIME_SECS	1	X'00000001'

MQTM_ * (Struttura messaggio trigger)

Tabella 381. Strutture di costanti

Nome	Struttura
ID_STRUC_MQTM	"TM¬¬"
ID_STRUC_MQTM_ARRAY	'T','M','¬','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 382. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQTM_VERSION_1	1	X'00000001'
VERSIONE MQTM_CURRENT_	1	X'00000001'

MQTMC_ * (Struttura formato carattere messaggio trigger)

Tabella 383. Strutture di costanti

Nome	Struttura
ID_STRUC_MQTMC	"TMC¬"
ID_STRUC_MQTMC_ARRAY	'T','M','C','¬'
MQTMC_VERSION_1	"¬¬¬1"

Tabella 383. Strutture di costanti (Continua)	
Nome	**Struttura**
MQTMC_VERSION_2	"¬¬¬2"
VERSIONE MQTMC_CURRENT_	"¬¬¬2"
MQTMC_VERSION_1_ARRAY	'¬','¬','¬','1'
MQTMC_VERSION_2_ARRAY	'¬','¬','¬','2'
ARRAY MQTMC_CURRENT_VERSION_	'¬','¬','¬','2'

MQTOPT_ * (Tipo argomento)

Tabella 384. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
LOCALE MQTOP_	0	X'00000000'
MQTOP_CLUSTER	1	X'00000001'
MQTOPT_ALL	2	X'00000002'

MQTRAXSTR_ * (Avvio automatico traccia iniziatore canale)

Tabella 385. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQTRAXSTR_NO	0	X'00000000'
MQTRAXSTR_SÌ	1	X'00000001'

MQTSCOPE_ * (Ambito sottoscrizione)

Tabella 386. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQTSCOPE_QMGR	1	X'00000001'
MQTSCOPE_ALL	2	X'00000002'

MQTT_ * (Tipi di trigger)

Tabella 387. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQTT_NONE	0	X'00000000'
MQTT_FIRST	1	X'00000001'
MQTT_EVERY	2	X'00000002'
DEPTH MQT	3	X'00000003'

MQTYPE_ * (Tipi di dati proprietà)

Tabella 388. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQTYPE_AS_SET	0	X'00000000'
MQTYPE_NULL	2	X'00000002'
BOOLEAN MQTIPO	4	X'00000004'
MQTYPE_BYTE_STRING	8	X'00000008'

Tabella 388. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQTYPE_INT8	16	X'00000010'
MQTYPE_INT16	32	X'00000020'
MQTYPE_INT32	64	X'00000040'
LONG_MQTYPE	64	X'00000040'
MQTYPE_INT64	128	X'00000080'
MQTYPE_FLOAT32	256	X'00000100'
MQTYPE_FLOAT64	512	X'00000200'
MQTYPE_STRING	1024	X'00000400'

MQUA_ * (Selettori di attributi utente di pubblicazione / sottoscrizione)

Tabella 389. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQUA_FIRST	65536	X'00010000'
MQUA_LAST	999999999	X'3B9AC9FF'

MQUIDSUPP_ * (Supporto ID utente formato comando)

Tabella 390. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQUIDSUPP_NO	0	X'00000000'
SÌ MQUIDSUPP	1	X'00000001'

MQUNDELIVERED_ * (Formato del comando Valori non consegnati per la migrazione Pub / Sot)

Tabella 391. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQUNDELIVER_NORMAL	0	X'00000000'
MQUNDELIVER_SAFE	1	X'00000001'
MQUNDELIVER_DISCARD	2	X'00000002'
KEEP MQUNDELIVER_	3	X'00000003'

MQUOWST_ * (Formato del comando Stati UOW)

Tabella 392. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQUOWST_NONE	0	X'00000000'
MQUOWST_ACTIVE	1	X'00000001'
MQUOWST_PREPARED	2	X'00000002'
MQUOWST_UNRISOLTO	3	X'00000003'

MQUOWT_ * (Formato del comando Tipi UOW)

Tabella 393. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQUOWT_Q_MGR	0	X'00000000'
MQUOWT_CICS	1	X'00000001'
MQUOWT_RRS	2	X'00000002'
IMS MQUOWT_	3	X'00000003'
MQUOWT_XA	4	X'00000004'

MQUS_ * (Utilizzi coda)

Tabella 394. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQUS_NORMALE	0	X'00000000'
MQUS_TRASMISSIONE	1	X'00000001'

MQUSAGE_ * (Formato del comando Valori di utilizzo della serie di pagine e valori di utilizzo del dataset)

Valori di utilizzo della serie di pagine del formato del comando

Tabella 395. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQUSAGE_PS_DISPONIBILE	0	X'00000000'
MQUSAGE_PS_DEFINITO	1	X'00000001'
MQUSAGE_PS_OFFLINE	2	X'00000002'
MQUSAGE_PS_NOT_DEFINED	3	X'00000003'
MQUSAGE_PS_SOSPESO	4	X'00000004'
UTENTE_ESPAND_MQUSAGE_	1	X'00000001'
MQUSAGE_EXPAND_SISTEMA	2	X'00000002'
MQUSAGE_EXPAND_NONE	3	X'00000003'

Formato del comando Valori di utilizzo del dataset

Tabella 396. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQUSAGE_DS_OLDEST_ACTIVE_UOW	10	X'0000000A'
MQUSAGE_DS_OLDEST_PS_RECOVERY	11	X'0000000B'
MQUSAGE_DS_OLDEST_CF_RECOVERY	12	X'0000000C'

MQVL_ * (Lunghezza valore)

Tabella 397. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQVL_NULL_TERMINATO	-1	X'FFFFFFFF'
MQVL_EMPTY_STRING	0	X'00000000'

MQVU_ * (ID utente variabile)

Tabella 398. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQVU_FIX_UTENTE	1	X'00000001'
MQVU_ANY_USER	2	X'00000002'

MQWDR_ * (Struttura record di destinazione uscita carico di lavoro cluster)

Tabella 399. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQWDR	"WDR¬"
MATRICE MQWDR_STRUC_ID_ARRAY	'W','D','R','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 400. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQWDR_VERSION_1	1	X'00000001'
MQWDR_VERSION_2	2	X'00000002'
VERSIONE MQWDR_CURRENT_	2	X'00000002'
MQWDR_LENGTH_1	124	X'0000007C'
MQWDR_LENGTH_2	136	X'00000088'
MQWDR_CURRENT_LENGTH	136	X'00000088'

MQWI_ * (Intervallo di attesa)

Tabella 401. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQWI_ILLIMITATO	-1	X'FFFFFFFF'

MQWIH_ * (Indicatori e struttura intestazione delle informazioni sul carico di lavoro)

Struttura intestazione informazioni carico di lavoro

Tabella 402. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQWIH	"WIH¬"
MATRICE MQWIH_STRUC_ID_ARRAY	'W','I','H','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 403. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQWIH_VERSION_1	1	X'00000001'
VERSIONE MQWIH_CURRENT_	1	X'00000001'
MQWIH_LENGTH_1	120	X'00000078'

Tabella 403. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQWIH_CURRENT_LENGTH	120	X'00000078'

Indicatori intestazione informazioni carico di lavoro

Tabella 404. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQWIH_NONE	0	X'00000000'

MQWQR_ * (Struttura record coda di uscita carico di lavoro cluster)

Tabella 405. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQWQR	"WQR¬"
ARRAY MQWQR_STRUC_ID_ARRAY	'W','Q','R','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 406. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQWQR_VERSION_1	1	X'00000001'
MQWQR_VERSION_2	2	X'00000002'
MQWQR_VERSION_3	3	X'00000003'
VERSIONE MQWQR_CURRENT_	3	X'00000003'
MQWQR_LENGTH_1	200	X'000000C8'
MQWQR_LENGTH_2	208	X'000000D0'
MQWQR_LENGTH_3	212	X'000000D4'
MQWQR_XX_ENCODE_CASE_ONE valore_corrente	212	X'000000D4'

MQWS_ * (Schema carattere jolly)

Tabella 407. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQWS_DEFAULT	0	X'00000000'
CAR MQWS	1	X'00000001'
TOPIC MQWS	2	X'00000002'

MQWXP_ * (Struttura parametro uscita carico di lavoro cluster)

MQWXP_ * (Struttura parametro uscita carico di lavoro cluster)

Tabella 408. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQWX	"WXP¬"
MATRA_ID_STRUTTURA_MQWXP	'W','X','P','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 409. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQWXP_VERSION_1	1	X'00000001'
MQWXP_VERSION_2	2	X'00000002'
MQWXP_VERSION_3	3	X'00000003'
MQWXP_VERSION_4	4	X'00000004'
VERSIONE MQWXP_CURRENT_	4	X'00000004'

MQWXP_ * (Indicatori carico di lavoro cluster)

Tabella 410. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQWXP_PUT_BY_CLUSTER_CHL	2	X'00000002'

Riferimenti correlati

“Campi in MQWXP - Struttura dei parametri di uscita del carico di lavoro del cluster” a pagina 1586
Descrizione dei campi in MQWXP - Struttura del parametro di uscita del carico di lavoro del cluster

MQXACT_ * (Tipi di chiamante API)

Tabella 411. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQXACT_EXTERNAL	1	X'00000001'
MQXACT_INTERNO	2	X'00000002'

MQXC_ * (Comandi di uscita)

Tabella 412. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQX_MQOPEN	1	X'00000001'
MQX_MQCLOSE	2	X'00000002'
MQX_MQGET	3	X'00000003'
MQXC_MQPUT	4	X'00000004'
MQXC_MQPUT1	5	X'00000005'
MQX_MQINQ	6	X'00000006'
MQXC_MQSET	8	X'00000008'
MQX_MQBACK	9	X'00000009'
MQX_MQCMIT	10	X'0000000A'

MQXCC_ * (Risposta uscita)

Tabella 413. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQXCC_OK	0	X'00000000'
MQXCC_SUPPRESS_FUNZIONE	-1	X'FFFFFFFF'
FUNZIONE SKIP_MQXCC	-2	X'FFFFFFFE'
MQXCC_SEND_AND_REQUEST_SEC_MSG	-3	X'FFFFFFFD'

Tabella 413. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MQXCC_SEND_SEC_MSG	-4	X'FFFFFFFC'
MQXCC_SUPPRESS_EXIT	-5	X'FFFFFFFB'
MQXCC_CLOSE_CHANNEL	-6	X'FFFFFFFA'
RICHIESTA_MQXCC	-7	X'FFFFFFF9'
MQXCC_NON RIUSCITO	-8	X'FFFFFFF8'

MQXDR_ * (Esci dalla risposta)

Tabella 414. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQXDR_OK	0	X'00000000'
MQXDR_CONVERSION_FAILED	1	X'00000001'

MQXE_ * (Ambienti)

Tabella 415. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQXE_ALTRO	0	X'00000000'
MQXE_MCA	1	X'00000001'
MQXE_MCA_SVRCONN	2	X'00000002'
SERVER MQXE_COMMAND_	3	X'00000003'
MQXE_MQSC	4	X'00000004'

MQXEPO_ * (Registra struttura opzioni punto di ingresso e opzioni di uscita)

Struttura delle opzioni del punto di ingresso del registro

Tabella 416. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQXEPO_	"XEPO"
MATRICE MQXEPO_STRUC_ID_ARRAY	'X','E','P','O'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 417. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQXEPO_VERSION_1	1	X'00000001'
VERSIONE MQXEPO_CURRENT_	1	X'00000001'

Opzioni di uscita

Tabella 418. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQXEPO_NONE	0	X'00000000'

MQXF_ * (Identificativi funzione API)

Tabella 419. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
INIT MQXF	1	X'00000001'
TERM_MQXF	2	X'00000002'
CONN MQXF	3	X'00000003'
MQXF_CONNX	4	X'00000004'
DISC MQXF	5	X'00000005'
MQXF_OPEN	6	X'00000006'
CLOSE MQXF	7	X'00000007'
MQXF_PUT1	8	X'00000008'
MQXF_PUT	9	X'00000009'
GET MQXF	10	X'0000000A'
CONV_DATA_MQXF_GET	11	X'0000000B'
INQ MQXF	12	X'0000000C'
SET MQXF	13	X'0000000D'
BEGIN MQXF	14	X'0000000E'
CMIT_MQXF	15	X'0000000F'
MQXF_BACK	16	X'00000010'
STAT_MQXF	18	X'00000012'
CB MQXF	19	X'00000013'
CTL MQXF	20	X'00000014'
CALLBACK MQXF	21	X'00000015'
SUB MQXF	22	X'00000016'
MQXF_SUBRQ	23	X'00000017'
XACLOSE MQXF	24	X'00000018'
XACOMMIT MQXF	25	X'00000019'
MQXF_XACOMPLETE	26	X'0000001A'
XAEND MQXF	27	X'0000001B'
XAFORGET MQXF	28	X'0000001C'
XAOPEN MQXF	29	X'0000001D'
XAPREPARE MQXF	30	X'0000001E'
MQXF_XARECOVER	31	X'0000001F'
MQXF_XAROLLBACK	32	X'00000020'
XASTART MQXF	33	X'00000021'
MQXF_AXREG	34	X'00000022'
MQXF_AXUNREG	35	X'00000023'

MQXP_ * (struttura parametro uscita attraversamento API)

Tabella 420. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQXP	"XP¬¬"
MATRA_ID_STRUTTURA_MQXP	'X','P','¬','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 421. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQXP_VERSION_1	1	X'00000001'

MQXPDA_ * (Area di determinazione dei problemi)

Tabella 422. Nomi e valori costanti	
Nome	**Valore**
MQXPDA_NONE	X'00...00' (48 valori null)
MQXPDA_NON_ARRAY	'\0','\0',... (48 valori null)

MQXPT_ * (Tipi di trasporto)

Tabella 423. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQXPT_ALL	-1	X'FFFFFFFF'
LOCALE MQXPT	0	X'00000000'
MQXPT_LU62	1	X'00000001'
TCP MQXPT	2	X'00000002'
NETBIOS MQXPT	3	X'00000003'
SPX MQXPT	4	X'00000004'
DECNET MQXPT	5	X'00000005'
UDP MQXPT	6	X'00000006'

MQXQH_ * (Struttura intestazione coda di trasmissione)

Tabella 424. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQXQH	"XQH¬"
MQXQH_STRUC_ID_ARRAY	'X','Q','H','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 425. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQXQH_VERSION_1	1	X'00000001'
VERSIONE MQXQH_CURRENT_	1	X'00000001'

MQXR_ * (Motivi uscita)

Tabella 426. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQXR_BEFORE	1	X'00000001'
MQXR_XX_ENCODE_CASE_ONE dopo	2	X'00000002'
MQXR_CONNECZIONE	3	X'00000003'
INIT MQXR	11	X'0000000B'
MQXR_TERM	12	X'0000000C'
MQXR_MSG	13	X'0000000D'
XMIT MQXR	14	X'0000000E'
MQXR_SEC_MSG	15	X'0000000F'
MQXR_INIT_SEC	16	X'00000010'
MQXR_RETRY	17	X'00000011'
MQXR_AUTO_CLUSSDR	18	X'00000012'
MQXR_AUTO_RECEIVER	19	X'00000013'
MQXR_CLWL_OPEN	20	X'00000014'
MQXR_CLWL_PUT	21	X'00000015'
MQXR_CLWL_MOVE	22	X'00000016'
REPOS CLW_MQXR	23	X'00000017'
MQXR_CLWL_REPOS_MOVE	24	X'00000018'
MQXR_END_BATCH	25	X'00000019'
MQXR_ACK_RICEVUTO	26	X'0000001A'
SVRCONN MQXR_AUTO_	27	X'0000001B'
MQXR_AUTO_CLUSRCVR	28	X'0000001C'
PARM_SEC_MQXR	29	X'0000001D'

MQXR2_* (Risposta uscita 2)

Tabella 427. Valori delle costanti

Nome	Valore decimale	Valore esadecimale
MQXR2_PUT_WITH_DEF_ACTION	0	X'00000000'
MQXR2_PUT_WITH_DEF_USERID	1	X'00000001'
MQXR2_PUT_WITH_MSG_USERID	2	X'00000002'
MQXR2_USE_AGENT_BUFFER	0	X'00000000'
MQXR2_USE_EXIT_BUFFER	4	X'00000004'
MQXR2_DEFAULT_CONTINUATION	0	X'00000000'
MQXR2_CONTINUE_CHAIN	8	X'00000008'
MQXR2_SUPPRESS_CHAIN	16	X'00000010'
MQXR2_STATIC_CACHE	0	X'00000000'
MQXR2_DYNAMIC_CACHE	32	X'00000020'

MQXT_ * (Identificativi uscita)

Tabella 428. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQXT_API_CROSSING_EXIT	1	X'00000001'
MQXT_API_EXIT	2	X'00000002'
MQXT_CHANNEL_SEC_EXIT	11	X'0000000B'
MQXT_CHANNEL_MSG_EXIT	12	X'0000000C'
MQXT_CHANNEL_SEND_EXIT	13	X'0000000D'
MQXT_CHANNEL_RCV_EXIT	14	X'0000000E'
MQXT_CHANNEL_MSG_RETRY_EXIT	15	X'0000000F'
MQXT_CHANNEL_AUTO_DEF_EXIT	16	X'00000010'
MQXT_CLUSTER_WORKLOAD_EXIT	20	X'00000014'
MQXT_PUBSUB_ROUTING_EXIT	21	X'00000015'

MQXUA_ * (Valore area utente di uscita)

Tabella 429. Nomi e valori costanti	
Nome	**Valore**
MQXUA_NONE	X'00...00' (16 valori null)
MQXUA_NON_ARRAY	'\0','\0',... (16 valori null)

MQXWD_ * (Struttura descrittore attesa uscita)

Tabella 430. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQXWD_	"XWD¬"
MQXWD_STRUC_ID_ARRAY	'X','W','D','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 431. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQXWD_VERSION_1	1	X'00000001'

MQZAC_ * (Struttura contesto applicazione)

Tabella 432. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQZAC	"ZAC¬"
MATRA_ID_STRUTTURA_MQZAC_	'Z','A','C','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 433. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZAC_VERSION_1	1	X'00000001'
VERSIONE MQZAC_CURRENT_	1	X'00000001'

MQZAD_ * (Struttura dati autorità)

Tabella 434. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQZAD	"ZAD¬"
ARRAY MQZAD_STRUC_ID_ARRAY	'Z','A','D','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 435. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZAD_VERSION_1	1	X'00000001'
MQZAD_VERSION_2	2	X'00000002'
VERSIONE MQZAD_CURRENT_	2	X'00000002'

MQZAET_ * (Tipi di entità servizi installabili)

Tabella 436. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZAET_NONE	0	X'00000000'
PRINCIPALE_MQZAET	1	X'00000001'
GRUPPO_MQZ	2	X'00000002'
MQZAET_SCONOSCIUTO	3	X'00000003'

MQZAO_ * (Autorizzazioni servizi installabili)

Tabella 437. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
CONNECT MQZAO_	1	X'00000001'
MQZAO_BROWSE	2	X'00000002'
INPUT MQZAO_	4	X'00000004'
OUTPUT MQZAO_	8	X'00000008'
INQUIRE MQZAO_	16	X'00000010'
MQZAO_SET	32	X'00000020'
Contesto MQZAO_PASS_IDENTITY_CONTEXT	64	X'00000040'
MQZAO_PASS_ALL_CONTEXT	128	X'00000080'
MQZAO_SET_IDENTITY_CONTEXT	256	X'00000100'
MQZAO_SET_ALL_CONTEXT	512	X'00000200'
MQZAO_ALTERNATE_USER_AUTHORITY	1024	X'00000400'
MQZAO_PUBBLICA	2048	X'00000800'
MQZAO_SUBSCRIBE	4096	X'00001000'
RESUME MQZAO_	8192	X'00002000'
MQZAO_ALL_MQI	16383	X'00003FFF'
CREA_MQZAO_	65536	X'00010000'
MQZAO_DELETE	131072	X'00020000'
DISPLAY MQZAO_	262144	X'00040000'

Tabella 437. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
MODIFICA_MQZO	524288	X'00080000'
CLEAR MQZAO_	1048576	X'00100000'
CONTROL MQZAO_	2097152	X'00200000'
MQZAO_CONTROL_XX_ENCODE_CASE_ONE uscita	4194304	X'00400000'
MQZAO_AUTHORIZE	8388608	X'00800000'
MQZAO_ALL_ADMIN	16646144	X'00FE0000'
MQZAO_ALL	16662527	X'00FE3FFF'
MQZAO_REMOVE	16777216	X'01000000'
MQZAO_NONE	0	X'00000000'

MQZAS_ * (Versione interfaccia servizio servizi installabili)

Tabella 438. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZAS_VERSION_1	1	X'00000001'
MQZAS_VERSION_2	2	X'00000002'
MQZAS_VERSION_3	3	X'00000003'
MQZAS_VERSION_4	4	X'00000004'
MQZAS_VERSION_5	5	X'00000005'
MQZAS_VERSION_6	6	X'00000006'

MQZAT_ * (Tipi di autenticazione)

Tabella 439. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZAT_CONTESTO_INIZIALE	0	X'00000000'
MQZAT_CONTESTO_MODIFICA	1	X'00000001'

MQZCI_ * (Indicatore di continuazione servizi installabili)

Tabella 440. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZCI_PREDEFINITO	0	X'00000000'
CONTINUE MQZCI	0	X'00000000'
STOP MQZCI	1	X'00000001'

MQZED_ * (Struttura dati entità)

Tabella 441. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQZ_	"ZED¬"
MQZED_STRUC_ID_ARRAY	'Z','E','D','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 442. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZED_VERSION_1	1	X'00000001'
MQZED_VERSION_2	2	X'00000002'
VERSIONE MQZED_CURRENT_	2	X'00000002'

MQZFP_ * (Struttura parametri liberi)

Tabella 443. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQZFP	"ZFP¬"
ID_STRUC_MQZFP_ARRAY	'Z','F','P','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 444. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZFP_VERSION_1	1	X'00000001'
VERSIONE MQZFP_CURRENT_	1	X'00000001'

MQZIC_ * (Struttura contesto identità)

Tabella 445. Strutture di costanti	
Nome	**Struttura**
ID_STRUC_MQZIC	"ZIC¬"
ARRAY MQZIC_STRUC_ID_ARRAY	'Z','I','C','¬'

Nota: Il simbolo ¬ rappresenta un singolo carattere vuoto.

Tabella 446. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZIC_VERSION_1	1	X'00000001'
VERSIONE MQZIC_CURRENT_	1	X'00000001'

MQZID_ * (ID funzione per i servizi)

ID funzione comuni a tutti i servizi

Tabella 447. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
IDMQZ_INIT	0	X'00000000'
MQZID_TERM	1	X'00000001'

ID funzione per il servizio Authority

Tabella 448. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZID_INIT_AUTHORITY	0	X'00000000'

Tabella 448. Valori delle costanti (Continua)		
Nome	**Valore decimale**	**Valore esadecimale**
AUTORIZZAZIONE MQZID_TERM	1	X'00000001'
MQZID_CHECK_AUTHORITY	2	X'00000002'
MQZID_COPY_ALL_AUTHORITY	3	X'00000003'
MQZID_DELETE_AUTHORITY	4	X'00000004'
MQZID_SET_AUTHORITY	5	X'00000005'
MQZID_GET_AUTHORITY	6	X'00000006'
MQZID_GET_EXPLICIT_AUTHORITY	7	X'00000007'
MQZID_AGGIORNA_CACHE	8	X'00000008'
MQZID_ENUMERATE_AUTHORITY_DATA	9	X'00000009'
UTENTE_AUTENTICAZIONE_MQZID	10	X'0000000A'
MQZID_FREE_UTENTE	11	X'0000000B'
INQUIRE MQZID	12	X'0000000C'
MQZID_CHECK_PRIVILEGED	13	X'0000000D'

ID funzione per il Servizio nomi

Tabella 449. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZID_INIT_NAME	0	X'00000000'
NOME MQZID_TERM	1	X'00000001'
MQZID_NOME_RICERCA	2	X'00000002'
NOME MQZID_INSERT	3	X'00000003'
MQZID_DELETE_NAME	4	X'00000004'

ID funzione per il servizio ID utente

Tabella 450. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
IDMQZ_INIT_IDUTENTE	0	X'00000000'
IDMQZ_TERM_IDUTENTE	1	X'00000001'
IDMQZ_FIND_IDUTENTE	2	X'00000002'

MQZIO_ * (Opzioni di inizializzazione dei servizi installabili)

Tabella 451. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZIO_PRIMARIO	0	X'00000000'
MQZIO_SECONDARY	1	X'00000001'

MQZNS_ * (Versione interfaccia servizio nomi)

Tabella 452. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZNS_VERSION_1	1	X'00000001'

MQZSE_ * (Avvio servizi installabili - Indicatore enumerazione)

Tabella 453. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
INIZIO_MQZSE	1	X'00000001'
CONTINUE MQZSE	0	X'00000000'

MQZSL_ * (Indicatore selettore servizi installabili)

Tabella 454. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZSL_NON_RESTITUITO	0	X'00000000'
MQZSL_RESTITUITO	1	X'00000001'

MQZTO_ * (Opzioni di terminazione dei servizi installabili)

Tabella 455. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZTO_PRIMARIO	0	X'00000000'
MQZTO_SECONDARY	1	X'00000001'

MQZUS_ * (Versione interfaccia servizio ID utente)

Tabella 456. Valori delle costanti		
Nome	**Valore decimale**	**Valore esadecimale**
MQZUS_VERSION_1	1	X'00000001'

Tipi di dati utilizzati in MQI

Informazioni sui tipi di dati che possono essere utilizzati in MQI (Message Queue Interface). Descrizioni, campi e dichiarazioni di lingua per le lingue pertinenti con ciascun tipo di dati.

Tipi di dati e programmazione per MQI

Introduzione ai tipi di dati Elementary e Structure e come utilizzare la programmazione MQI tramite C, COBOL o High Level Assembler .

Tipi di dati elementari

Informazioni sui tipi di dati utilizzati in MQI o nelle funzioni di uscita. Questi sono descritti in dettaglio, seguiti da esempi che mostrano come dichiarare i tipi di dati elementari nei linguaggi di programmazione supportati.

I tipi di dati utilizzati in MQI o nelle funzioni di uscita sono:

- Tipi di dati elementari o
- Aggregati di tipi di dati elementari (schiere o strutture)

I seguenti tipi di dati elementari vengono utilizzati in MQI o nelle funzioni di uscita:

Tabella 457. Nomi, tipi e descrizioni dei tipi di dati elementari

Nome tipo di dati elementare	Tipo dati	Descrizione
MQBOOL	Booleano	Il tipo di dati MQBOOL rappresenta un valore booleano. Il valore 0 rappresenta false. Qualsiasi altro valore rappresenta true. Un MQBOOL deve essere allineato come per il tipo di dati MQLONG.
MQBYTE	Byte	Il tipo di dati MQBYTE rappresenta un singolo byte di dati. Nessuna interpretazione particolare viene posizionata sul byte; viene trattata come una stringa di bit e non come un numero binario o un carattere. Non è richiesto alcun allineamento speciale. Quando i dati MQBYTE vengono inviati tra gestori code che utilizzano serie di caratteri o codifiche differenti, i dati MQBYTE non vengono convertiti in alcun modo. I campi *MsgId* e *CorrelId* nella struttura MQMD sono simili. Un array di MQBYTE viene talvolta utilizzato per rappresentare un'area di memoria principale non nota al gestore code. Ad esempio, l'area potrebbe contenere i dati del messaggio dell'applicazione o una struttura. L'allineamento dei limiti di questa area deve essere compatibile con la natura dei dati contenuti al suo interno. Nel linguaggio di programmazione C, qualsiasi tipo di dati può essere utilizzato per i parametri di funzione mostrati come array di MQBYTE. Ciò è dovuto al fatto che tali parametri vengono sempre passati per indirizzo e in C il parametro della funzione viene dichiarato come un puntatore a vuoto.

Tabella 457. Nomi, tipi e descrizioni dei tipi di dati elementari (Continua)		
Nome tipo di dati elementare	**Tipo dati**	**Descrizione**
MQBYTEn	Stringa di *n* byte	Ogni tipo di dati MQBYTEn rappresenta una stringa di *n* byte, dove *n* può assumere uno dei seguenti valori: 8, 16, 24, 32, 40 o 128. Ogni byte è descritto dal tipo di dati MQBYTE. Non è richiesto alcun allineamento speciale. Se i dati nella stringa di byte sono più brevi della lunghezza definita della stringa, i dati devono essere riempiti con valori null per riempire la stringa. Quando il gestore code restituisce stringhe di byte all'applicazione (ad esempio, sulla chiamata MQGET), il gestore code riempie con valori null la lunghezza definita della stringa. Le costanti denominate sono disponibili per definire le lunghezze dei campi stringa di byte. Questi sono elencati in "Costanti" a pagina 61
MQCAR	Carattere	Il tipo di dati MQCHAR rappresenta un carattere a byte singolo o un byte di un carattere a byte doppio o a più byte. Non è richiesto alcun allineamento speciale. Quando i dati MQCHAR vengono inviati tra gestori code che utilizzano diverse serie di caratteri o codifiche, i dati MQCHAR di solito richiedono la conversione affinché i dati vengano interpretati correttamente. Il gestore code esegue questa operazione automaticamente per i dati MQCHAR nella struttura MQMD. La conversione dei dati MQCHAR nei dati del messaggio applicazione è controllata dall'opzione MQGMO_CONVERT specificata nella chiamata MQGET; consultare la descrizione di questa opzione in "MQGMO - Opzioni Get - message" a pagina 376 per ulteriori dettagli.

Tabella 457. Nomi, tipi e descrizioni dei tipi di dati elementari (Continua)		
Nome tipo di dati elementare	**Tipo dati**	**Descrizione**
MQCARN	Stringa di *n* caratteri	Ogni tipo di dati MQCHARn rappresenta una stringa di *n* caratteri, dove *n* può assumere uno qualsiasi dei seguenti valori: 4, 8, 12, 20, 28, 32, 48, 64, 128 o 256. Ogni carattere è descritto dal tipo di dati MQCHAR. Non è richiesto alcun allineamento speciale. Se i dati nella stringa sono più brevi della lunghezza definita della stringa, i dati devono essere riempiti con spazi vuoti per riempire la stringa. In alcuni casi è possibile utilizzare un carattere null per terminare prematuramente la stringa, invece di riempirla di spazi vuoti; il carattere null e i caratteri successivi vengono trattati come spazi vuoti, fino alla lunghezza definita della stringa. Le posizioni in cui è possibile utilizzare un valore null vengono identificate nelle descrizioni della chiamata e del tipo di dati. Quando il gestore code restituisce delle stringhe di caratteri all'applicazione (ad esempio, sulla chiamata MQGET), il gestore code riempisce sempre con spazi vuoti la lunghezza definita della stringa; il gestore code non utilizza il carattere null per delimitare la stringa. Sono disponibili costanti denominate che definiscono la lunghezza dei campi stringa di caratteri e sono elencate in “Costanti” a pagina 61.

Tabella 457. Nomi, tipi e descrizioni dei tipi di dati elementari (Continua)		
Nome tipo di dati elementare	**Tipo dati**	**Descrizione**
MQFLOAT32	Numero a virgola mobile a 32 bit	Il tipo di dati MQFLOAT32 è un numero a virgola mobile a 32 bit rappresentato utilizzando il formato a virgola mobile IEEE standard. Un MQFLOAT32 deve essere allineato su un limite di 4 byte. L'utilizzo di MQFLOAT32 in C su z/OS richiede l'utilizzo dell'indicatore del compilatore FLOAT (IEEE). L'utilizzo di MQFLOAT32 in COBOL è limitato ai compilatori che supportano numeri a virgola mobile in formato IEEE. Ciò potrebbe richiedere l'utilizzo dell'indicatore del compilatore FLOAT (NATIVE).
MQFLOAT64	Numero a virgola mobile a 64 bit	Il tipo di dati MQFLOAT64 è un numero a virgola mobile a 64 bit rappresentato utilizzando il formato a virgola mobile IEEE standard. Un MQFLOAT64 deve essere allineato su un limite di 8 byte. L'utilizzo di MQFLOAT64 in C su z/OS richiede l'utilizzo dell'indicatore del compilatore FLOAT (IEEE). L'utilizzo di MQFLOAT64 in COBOL è limitato ai compilatori che supportano numeri a virgola mobile in formato IEEE. Ciò potrebbe richiedere l'utilizzo dell'indicatore del compilatore FLOAT (NATIVE).
MQHCONFIG	Handle di configurazione	Il tipo di dati MQHCONFIG rappresenta un handle di configurazione, ossia il componente che viene configurato per un determinato servizio installabile. Un handle di configurazione deve essere allineato sul suo limite naturale. Le applicazioni non devono basarsi sul formato dei dati memorizzati all'interno di questo handle. Se valido, il suo valore è destinato ad essere utilizzabile in ulteriori chiamate MQI, ma non ha alcun significato oltre a tale scopo.

Tabella 457. Nomi, tipi e descrizioni dei tipi di dati elementari (Continua)

Nome tipo di dati elementare	Tipo dati	Descrizione
MQHCONN	ID interno connessioni	Il tipo di dati MQHCONN rappresenta un handle di connessione, ossia la connessione a uno specifico gestore code. Una gestione di connessione deve essere allineata su un limite di 4 byte. Le applicazioni non devono basarsi sul formato dei dati memorizzati all'interno di questo handle. Se valido, il suo valore è destinato ad essere utilizzabile in ulteriori chiamate MQI, ma non ha alcun significato oltre a tale scopo.
MQHMSG	handle del messaggio	Il tipo di dati MQHMSG rappresenta un handle del messaggio che fornisce l'accesso a un messaggio. Un gestore messaggi deve essere allineato su un limite di 8 byte. Le applicazioni non devono basarsi sul formato dei dati memorizzati all'interno di questo handle. Se valido, il suo valore è destinato ad essere utilizzabile in ulteriori chiamate MQI, ma non ha alcun significato oltre a tale scopo.
MQHOBG	Handle di oggetti	Il tipo di dati MQHOBJ rappresenta un handle di oggetto che fornisce l'accesso a un oggetto. Una gestione oggetto deve essere allineata su un limite di 4 byte. Le applicazioni non devono basarsi sul formato dei dati memorizzati all'interno di questo handle. Se valido, il suo valore è destinato ad essere utilizzabile in ulteriori chiamate MQI, ma non ha alcun significato oltre a tale scopo.
MQINT8	Numero intero con segno a 8 bit	Il tipo di dati MQINT8 è un numero intero con segno a 8 bit che può assumere qualsiasi valore compreso tra -128 e +127, a meno che non sia altrimenti limitato dal contesto.

Tabella 457. Nomi, tipi e descrizioni dei tipi di dati elementari (Continua)

Nome tipo di dati elementare	Tipo dati	Descrizione
MQINT16	Numero intero con segno a 16 bit	Il tipo di dati MQINT16 è un numero intero con segno a 16 bit che può assumere qualsiasi valore compreso tra -32 768 e +32 767, a meno che non sia altrimenti limitato dal contesto. Un MQINT16 deve essere allineato su un limite di 2 byte.
MQINT32	Numero intero con segno a 32 bit	Il tipo di dati MQINT32 è un numero intero binario con segno a 32 bit che può assumere qualsiasi valore compreso tra -2 147 483 648 e + 2 147 483 647, a meno che non sia altrimenti limitato dal contesto. Consultare le definizioni di MQLONG.
MQINT64	Numero intero firmato a 64 bit	Il tipo di dati MQINT64 è un numero intero con segno a 64 bit che può assumere qualsiasi valore compreso tra -9 223 372 036 854 775 808 e + 9 223 372 036 854 775 807, a meno che non sia altrimenti limitato dal contesto. Per COBOL, l'intervallo valido è limitato da -999 999 999 999 999 999 a +999 999 999 999 999 999. Un numero intero a 64 bit deve essere allineato su un limite di 8 byte.
MQLONG	Numero intero con segno a 32 bit	Il tipo di dati MQLONG è un numero intero binario con segno a 32 bit che può assumere qualsiasi valore compreso tra -2 147 483 648 e + 2 147 483 647, a meno che non sia altrimenti limitato dal contesto. Per COBOL, l'intervallo valido è limitato da -999 999 999 a +999 999 999 999. MQLONG deve essere allineato su un limite di 4 byte.
IDMQP	Identificativo del processo	L'identificativo del processo IBM MQ . Questo è lo stesso identificativo utilizzato nella traccia di MQ e nei dump di FFST™ , ma potrebbe essere diverso dall'identificativo del processo del sistema operativo.

Tabella 457. Nomi, tipi e descrizioni dei tipi di dati elementari (Continua)

Nome tipo di dati elementare	Tipo dati	Descrizione
MQPTR	Puntatore	Il tipo di dati MQPTR è l'indirizzo dei dati di qualsiasi tipo. Un puntatore deve essere allineato sul suo limite naturale; si tratta di un limite di 16 byte su IBM ie di un limite di 8 byte su altre piattaforme. Alcuni linguaggi di programmazione supportano i puntatori digitati; l'MQI li utilizza anche in alcuni casi (ad esempio, PMQCHAR e PMQLONG nel linguaggio di programmazione C).
IDMQT	Identificativo thread	L'identificativo del thread IBM MQ . Si tratta dello stesso identificativo utilizzato nella traccia di MQ e nei dump di FFST™ , ma potrebbe essere diverso dall'identificativo del thread del sistema operativo.
MQUINT8	Numero intero senza segno a 8 bit	Il tipo di dati MQUINT8 è un numero intero senza segno a 8 bit che può assumere qualsiasi valore compreso tra 0 e +255, a meno che non sia altrimenti limitato dal contesto.
MQUINT16	Numero intero senza segno a 16 bit	Il tipo di dati MQUINT16 è un numero intero senza segno a 16 bit che può assumere qualsiasi valore compreso tra 0 e +65 535, a meno che non sia altrimenti limitato dal contesto. Un MQUINT16 deve essere allineato su un limite di 2 byte.
MQUINT32	Numero intero senza segno a 32 bit	Il tipo di dati MQUINT32 è un numero intero binario senza segno a 32 bit. Consultare la definizione di MQULONG.

Tabella 457. Nomi, tipi e descrizioni dei tipi di dati elementari (Continua)

Nome tipo di dati elementare	Tipo dati	Descrizione
MQUINT64	Numero intero senza segno a 64 bit	Il tipo di dati MQINT64 è un numero intero senza segno a 64 bit che può assumere qualsiasi valore compreso tra 0 e +18 446 744 073 709 551 615, a meno che non sia altrimenti limitato dal contesto. Per COBOL, l'intervallo valido è compreso tra 0 e 999 999 999 999 999 999. Un numero intero a 64 bit deve essere allineato su un limite di 8 byte.
MQULONG	Numero intero senza segno a 32 bit	Il tipo di dati MQULONG è un numero intero binario senza segno a 32 bit che può assumere qualsiasi valore compreso tra 0 e + 4 294 967 294, a meno che non sia altrimenti limitato dal contesto. Per COBOL, l'intervallo valido è limitato da 0 a +999 999 999. Un MQULONG deve essere allineato su un limite di 4 byte.
PMQACH	Puntatore	Puntatore a una struttura dati di tipo MQACH
PMQAIR	Puntatore	Puntatore a una struttura dati di tipo MQAIR
PMQAXC	Puntatore	Puntatore a una struttura dati di tipo MQAXC
PMQAXP	Puntatore	Puntatore ad una struttura di dati di tipo MQAXP
PMQBMO	Puntatore	Puntatore a una struttura dati di tipo MQBMHO
PMQBO	Puntatore	Puntatore a una struttura dati di tipo MQBO
PMQBOOL	Puntatore	Puntatore ai dati di tipo MQBOOL
PMQBYTE	Puntatore	Puntatore ai dati di tipo MQBYTE
PMQBYTEn	Puntatore	Puntatore ai dati di tipo MQBYTEn, dove n può essere 8, 16, 24, 32, 40, 128
PMQCBC	Puntatore	Puntatore ad una struttura di dati di tipo MQCBC
CBD PMQ	Puntatore	Puntatore ad una struttura dati di tipo MQCBD
PMQCAR	Puntatore	Puntatore ai dati di tipo MQCHAR

Tabella 457. Nomi, tipi e descrizioni dei tipi di dati elementari (Continua)

Nome tipo di dati elementare	Tipo dati	Descrizione
PMQCARN	Puntatore	Puntatore a un tipo di dati MQCHARN, dove n può essere 4, 8, 12, 20, 28, 32, 48, 64, 128, 256, 264
PMQCARV	Puntatore	Puntatore ad una struttura dati di tipo MQCHARV
PMQCIH	Puntatore	Puntatore a una struttura dati di tipo MQCIH
PMQCMO	Puntatore	Puntatore a una struttura dati di tipo MQCMHO
NMQPMQ	Puntatore	Puntatore a una struttura di dati di tipo MQCNO
PMQCSP	Puntatore	Puntatore a una struttura dati di tipo MQCSP
MMQCTLO	Puntatore	Puntatore a una struttura di dati di tipo MQCTLO
PMQDH	Puntatore	Puntatore a una struttura dati di tipo MQDH
PMQDHO	Puntatore	Puntatore a una struttura dati di tipo MQDHO
PMQDLH	Puntatore	Puntatore a una struttura dati di tipo MQDLH
PMQDMHO	Puntatore	Puntatore a una struttura dati di tipo MQDMHO
PMQDMPO	Puntatore	Puntatore a una struttura dati di tipo MQDMPO
PMQEPH	Puntatore	Puntatore a una struttura di dati di tipo MQEPH
PMQFLOAT32	Puntatore	Puntatore a una struttura dati di tipo MQFLOAT32
PMQFLOAT64	Puntatore	Puntatore a una struttura di dati di tipo MQFLOAT64
PMQFUNC	Puntatore	Puntatore a una funzione
PMQGMO	Puntatore	Puntatore a una struttura dati di tipo MQGMO
PMQHCONFIG	Puntatore	Puntatore ai dati di tipo MQHCONFIG
PMQHCONN	Puntatore	Puntatore ai dati di tipo MQHCONN
PMQHMSG	Puntatore	Puntatore ai dati di tipo MQHMSG
PMQHOBG	Puntatore	Puntatore ai dati di tipo MQHOBJ
PMQIIH	Puntatore	Puntatore a una struttura dati di tipo MQIIH

Tabella 457. Nomi, tipi e descrizioni dei tipi di dati elementari (Continua)

Nome tipo di dati elementare	Tipo dati	Descrizione
PMQIMPO	Puntatore	Puntatore a una struttura di dati di tipo MQIMPO
PMQINT8	Puntatore	Puntatore ai dati di tipo MQINT8
PMQINT16	Puntatore	Puntatore ai dati di tipo MQINT16
PMQINT32	Puntatore	Puntatore ai dati di tipo MQINT32
PMQINT64	Puntatore	Puntatore ai dati di tipo MQINT64
PMQLONG	Puntatore	Puntatore ai dati di tipo MQLONG
MMQP	Puntatore	Puntatore alla struttura di tipo MQMD
MQMDE	Puntatore	Puntatore a una struttura dati di tipo MQMDE
PMQMD1	Puntatore	Puntatore ad una struttura dati di tipo MQMD1
PMQMD2	Puntatore	Puntatore a una struttura di dati di tipo MQMD2
PMQMHBO	Puntatore	Puntatore a una struttura di dati di tipo MQMHBO
PMQOD	Puntatore	Puntatore a una struttura di dati di tipo MQOD
PMQOR	Puntatore	Puntatore a una struttura di dati di tipo MQOR
PMQPD	Puntatore	Puntatore a una struttura di dati di tipo MQPD
IDPMQP	Puntatore	Puntatore a un identificatore di processo
MMQP	Puntatore	Puntatore a una struttura dati di tipo MQMD
PMQPMO	Puntatore	Puntatore a una struttura di dati di tipo MQPMO
PMQPTR	Puntatore	Puntatore ai dati di tipo MQPTR
PMQRFH	Puntatore	Puntatore a una struttura di dati di tipo MQRFH
PMQRFH2	Puntatore	Puntatore a una struttura dati di tipo MQRFH2
PMQRMH	Puntatore	Puntatore a una struttura dati di tipo MQRMH
RMQP	Puntatore	Puntatore ad una struttura dati di tipo MQRR
PMQSCO	Puntatore	Puntatore a una struttura di dati di tipo MQSCO

Tabella 457. Nomi, tipi e descrizioni dei tipi di dati elementari (Continua)		
Nome tipo di dati elementare	**Tipo dati**	**Descrizione**
SMQP	Puntatore	Puntatore a una struttura di dati di tipo MQSD
PMQSMPO	Puntatore	Puntatore a una struttura di dati di tipo MQSMPO
SRO PMQ	Puntatore	Puntatore a una struttura dati di tipo MQSRO
PMSSTS	Puntatore	Puntatore a una struttura di dati di tipo MQSTS
IDMQT	Puntatore	Puntatore a un ID sottoprocesso
TMQP	Puntatore	Puntatore a una struttura di dati di tipo MQTM
PMQTMC2	Puntatore	Puntatore ad una struttura dati di tipo MQTMC2
PMQUINT8	Puntatore	Puntatore a un tipo di dati MQUINT8
PMQUINT16	Puntatore	Puntatore a un tipo di dati MQUINT16
PMQUINT32	Puntatore	Puntatore a un tipo di dati MQUINT32
PMQUINT64	Puntatore	Puntatore a un tipo di dati MQUINT64
PMQULONG	Puntatore	Puntatore a un tipo di dati MQULONG
IDVOPMQ	Puntatore	
PMQWIH	Puntatore	Puntatore a una struttura dati di tipo MQWIH
QQMQP	Puntatore	Puntatore a una struttura dati di tipo MQXQH

Dichiarazioni del tipo di dati C

Tabella 458. Nomi e rappresentazioni dei tipi di dati C	
Tipo dati	**Rappresentazione**
MQBOOL	`typedef MQLONG MQBOOL;`
MQBYTE	`typedef unsigned char MQBYTE;`
MQBYTE8	`typedef MQBYTE MQBYTE8[8];`

Tabella 458. Nomi e rappresentazioni dei tipi di dati C (Continua)

Tipo dati	Rappresentazione
MQBYTE16	`typedef MQBYTE MQBYTE16[16];`
MQBYTE24	`typedef MQBYTE MQBYTE24[24];`
MQBYTE32	`typedef MQBYTE MQBYTE32[32];`
MQBYTE40	`typedef MQBYTE MQBYTE40[40];`
MQCAR	`typedef char MQCHAR;`
MQCHAR4	`typedef MQCHAR MQCHAR4[4];`
MQCHAR8	`typedef MQCHAR MQCHAR8[8];`
MQCHAR12	`typedef MQCHAR MQCHAR12[12];`
MQCHAR20	`typedef MQCHAR MQCHAR20[20];`
MQCHAR28	`typedef MQCHAR MQCHAR28[28];`
MQCHAR32	`typedef MQCHAR MQCHAR32[32];`
MQCHAR48	`typedef MQCHAR MQCHAR48[48];`
MQCHAR64	`typedef MQCHAR MQCHAR64[64];`
MQCHAR128	`typedef MQCHAR MQCHAR128[128];`
MQCHAR256	`typedef MQCHAR MQCHAR256[256];`
MQFLOAT32	`typedef float MQFLOAT32;`

Tabella 458. Nomi e rappresentazioni dei tipi di dati C (Continua)

Tipo dati	Rappresentazione
MQFLOAT64	`typedef double MQFLOAT64;`
MQHCONFIG	`typedef void MQPOINTER MQHCONFIG;`
MQHCONN	`typedef MQLONG MQHCONN;`
MQHOBG	`typedef MQLONG MQHOBJ;`
MQINT8	`typedef signed char MQINT8;`
MQINT16	`typedef short MQINT16;`

MQINT64	UNIX Su UNIXa 64 bit:
	`typedef long;`
	AIX Su AIX:
	`typedef int64_t;`
	IBM i z/OS Linux Su Linux, IBM ie z/OS:
	`typedef long long;`
	Windows Su Windows:
	`typedef _int64;`

MQLONG	IBM i Su IBM i:
	`typedef long MQLONG;`
	z/OS ALW Su altre piattaforme:
	`if defined(MQ_64_BIT)`
	`  typedef int MQLONG;`
	`else`
	`  typedef long MQLONG;`

Tabella 458. Nomi e rappresentazioni dei tipi di dati C (Continua)

Tipo dati	Rappresentazione
IDMQP	`typedef MQLONG MQPID;`
MQPTR	`typedef void MQPOINTER MQPTR;`
IDMQT	`typedef MQLONG MQTID;`
MQUINT8	`typedef unsigned char MQUINT8;`
MQUINT16	`typedef unsigned short MQUINT16;`

MQUINT64	UNIX Su UNIXa 64 bit:
	`typedef unsigned long;`
	AIX Su AIX:
	`typedef uint64_t;`
	IBM i z/OS Linux Su Linux, IBM ie z/OS:
	`typedef unsigned long long;`
	Windows Su Windows:
	`typedef unsigned _int64;`

MQULONG	IBM i Su IBM i:
	`typedef unsigned long MQULONG;`
	z/OS ALW Su altre piattaforme:
	`if defined(MQ_64_BIT)`
	`  typedef unsigned int MQULONG;`
	`else`
	`  typedef unsigned long MQULONG;`

PMQBO	`typedef MQBO MQPOINTER PMQBO;`

Tabella 458. Nomi e rappresentazioni dei tipi di dati C (Continua)

Tipo dati	Rappresentazione
PMQBOOL	`typedef MQBOOL MQPOINTER PMQBOOL;`
PMQBYTE	`typedef MQBYTE MQPOINTER PMQBYTE;`
PMQBYTE8	`typedef MQBYTE8[8] MQPOINTER PMQBYTE8[8];`
PMQBYTE16	`typedef MQBYTE16[16] MQPOINTER PMQBYTE16[16];`
PMQBYTE24	`typedef MQBYTE24[24] MQPOINTER PMQBYTE24[24];`
PMQBYTE32	`typedef MQBYTE32[32] MQPOINTER PMQBYTE32[32];`
PMQBYTE40	`typedef MQBYTE40[40] MQPOINTER PMQBYTE40[40];`
PMQBYTE128	`typedef MQBYTE128[128] MQPOINTER PMQBYTE128[128];`
PMQCAR	`typedef MQCHAR MQPOINTER PMQCHAR;`
PMQCHAR4	`typedef MQCHAR4[4] MQPOINTER PMQCHAR4[4];`
PMQCHAR8	`typedef MQCHAR8[8] MQPOINTER PMQCHAR8[8];`
PMQCHAR12	`typedef MQCHAR12[12] MQPOINTER PMQCHAR12[12];`
PMQCHAR20	`typedef MQCHAR20[20] MQPOINTER PMQCHAR20[20];`
PMQCHAR28	`typedef MQCHAR28[28] MQPOINTER PMQCHAR28[28];`
PMQCHAR32	`typedef MQCHAR32[32] MQPOINTER PMQCHAR32[32];`
PMQCHAR48	`typedef MQCHAR48[48] MQPOINTER PMQCHAR48[48];`

Tabella 458. Nomi e rappresentazioni dei tipi di dati C (Continua)

Tipo dati	Rappresentazione
PMQCHAR64	`typedef MQCHAR64[64] MQPOINTER PMQCHAR64[64];`
PMQCHAR128	`typedef MQCHAR128[128] MQPOINTER PMQCHAR128[128];`
PMQCHAR256	`typedef MQCHAR256[256] MQPOINTER PMQCHAR256[256];`
PMQCHAR264	`typedef MQCHAR264[264] MQPOINTER PMQCHAR264[264];`
PMQCIH	`typedef MQCIH MQPOINTER PMQCIH;`
NMQPMQ	`typedef MQCNO MQPOINTER PMQCNO;`
PMQDLH	`typedef MQDLH MQPOINTER PMQDLH;`
PMQFUNC	`typedef void MQPOINTER PMQFUNC;`
PMQFLOAT32	`typedef MQFLOAT32 MQPOINTER PMQFLOAT32;`
PMQFLOAT64	`typedef MQFLOAT64 MQPOINTER PMQFLOAT64;`
PMQGMO	`typedef MQGMO MQPOINTER PMQGMO;`
PMQHCONFIG	`typedef MQHCONFIG MQPOINTER PMQHCONFIG;`
PMQHCONN	`typedef MQHCONN MQPOINTER PMQHCONN;`
PMQHOBG	`typedef MQHOBJ MQPOINTER PMQHOBJ;`
PMQIIH	`typedef MQIIH MQPOINTER PMQIIH;`
PMQINT8	`typedef MQINT8 MQPOINTER PMQINT8;`

Tabella 458. Nomi e rappresentazioni dei tipi di dati C (Continua)

Tipo dati	Rappresentazione
PMQINT16	`typedef MQINT16 MQPOINTER PMQINT16;`
PMQLONG	`typedef MQLONG MQPOINTER PMQLONG;`
MMQP	`typedef MQMD MQPOINTER PMQMD;`
PMQMD1	`typedef MQMD1[1] MQPOINTER PMQMD1[1];`
MQMDE	`typedef MQMDE MQPOINTER PMQMDE;`
PMQOD	`typedef MQOD MQPOINTER PMQOD;`
PMQPMO	`typedef MQPMO MQPOINTER PMQPMO;`
PMQPTR	`typedef MQPTR MQPOINTER PMQPTR;`
PMQRFH	`typedef MQRFH MQPOINTER PMQRFH;`
PMQRFH2	`typedef MQRFH2[2] MQPOINTER PMQRFH2[2];`
PMQRMH	`typedef MQRMH MQPOINTER PMQRMH;`
TMQP	`typedef MQTM MQPOINTER PMQTM;`
PMQTMC2	`typedef MQTMC2[2] MQPOINTER PMQTMC2[2];`
PMQUINT8	`typedef MQUINT8 MQPOINTER PMQUINT8;`
PMQUINT16	`typedef MQUINT16 MQPOINTER PMQUINT16;`
PMQULONG	`typedef MQULONG MQPOINTER PMQULONG;`

Tabella 458. Nomi e rappresentazioni dei tipi di dati C (Continua)

Tipo dati	Rappresentazione
IDVOPMQ	`typedef void MQPOINTER PMQVOID;`
PMQWIH	`typedef MQWIH MQPOINTER PMQWIH;`
QQMQP	`typedef MQXQH MQPOINTER PMQXQH;`
PPMQBO	`typedef PMQBO MQPOINTER PPMQBO;`
PPMQBYTE	`typedef PMQBYTE MQPOINTER PPMQBYTE;`
PPMQCAR	`typedef PMQCHAR MQPOINTER PPMQCHAR;`
MQCNO PP	`typedef PMQCNO MQPOINTER PPMQCNO;`
MQGPP	`typedef PMQGMO MQPOINTER PPMQGMO;`
CONNMQPP	`typedef PMQHCONN MQPOINTER PPMQHCONN;`
PPMQHOBG	`typedef PMQHOBJ MQPOINTER PPMQHOBJ;`
PMQLONG	`typedef PMQLONG MQPOINTER PPMQLONG;`
MMQPP	`typedef PMQMD MQPOINTER PPMQMD;`
PPMQOD	`typedef PMQOD MQPOINTER PPMQOD;`
PPMQPMO	`typedef PMQPMO MQPOINTER PPMQPMO;`
PPMQULONG	`typedef PMQULONG MQPOINTER PPMQULONG;`
IDVOMQPP	`typedef PMQVOID MQPOINTER PPMQVOID;`
Dove `defined(MQ_64_BIT)` indica una piattaforma a 64 bit.	

Consultare “Tipi di dati” a pagina 265 per una descrizione della variabile della macro MQPOINTER.

Dichiarazioni del tipo di dati COBOL

Tabella 459. Rappresentazioni e nomi dei tipi di dati COBOL

Tipo dati	**Rappresentazione**
MQBOOL	`PIC S9(9) BINARY`
MQBYTE	`PIC X`
MQBYTE8	`PIC X(8)`
MQBYTE16	`PIC X(16)`
MQBYTE24	`PIC X(24)`
MQBYTE32	`PIC X(32)`
MQBYTE40	`PIC X(40)`
MQCAR	`PIC X`
MQCHAR4	`PIC X(4)`
MQCHAR8	`PIC X(8)`
MQCHAR12	`PIC X(12)`
MQCHAR20	`PIC X(20)`
MQCHAR28	`PIC X(28)`
MQCHAR32	`PIC X(32)`
MQCHAR48	`PIC X(48)`

Tabella 459. Rappresentazioni e nomi dei tipi di dati COBOL (Continua)

Tipo dati	Rappresentazione
MQCHAR64	`PIC X(64)`
MQCHAR128	`PIC X(128)`
MQCHAR256	`PIC X(256)`
MQFLOAT32	`USAGE COMP-1`
MQFLOAT64	`USAGE COMP-2`

MQHCONN	Attivoz/OS
	`PIC S9(9) COMP-5`
	Su altre piattaforme
	`PIC S9(9) BINARY`

MQHOBG	`PIC S9(9) BINARY`
MQINT8	`PIC S9(2) BINARY`
MQINT16	`PIC S9(4) BINARY`
MQINT64	`PIC S9(18) BINARY`
MQLONG	`PIC S9(9) BINARY`
MQPTR	`POINTER`
MQUINT8	`PIC 9(2) BINARY`
MQUINT16	`PIC 9(4) BINARY`

Tabella 459. Rappresentazioni e nomi dei tipi di dati COBOL (Continua)	
Tipo dati	**Rappresentazione**
MQUINT64	`PIC 9(18) BINARY`
MQULONG	`PIC 9(9) BINARY`

Dichiarazioni tipo di dati PL/I

Tabella 460. Nomi e rappresentazioni dei tipi di dati PL/I	
Tipo dati	**Rappresentazione**
MQBOOL	`fixed bin(31)`
MQBYTE	`char(1)`
MQBYTE8	`char(8)`
MQBYTE16	`char(16)`
MQBYTE24	`char(24)`
MQBYTE32	`char(32)`
MQBYTE40	`char(40)`
MQCAR	`char(1)`
MQCHAR4	`char(4)`
MQCHAR8	`char(8)`
MQCHAR12	`char(12)`
MQCHAR20	`char(20)`

Tabella 460. Nomi e rappresentazioni dei tipi di dati PL/I (Continua)

Tipo dati	Rappresentazione
MQCHAR28	`char(28)`
MQCHAR32	`char(32)`
MQCHAR48	`char(48)`
MQCHAR64	`char(64)`
MQCHAR128	`char(128)`
MQCHAR256	`char(256)`
MQFLOAT32	`binary float(21) ieee`
MQFLOAT64	`binary float(52) ieee`
MQHCONN	`fixed bin(31)`
MQHOBG	`fixed bin(31)`
MQINT8	`fixed bin(7)`
MQINT16	`fixed bin(15)`
MQINT64	`fixed bin(63)`
MQLONG	`fixed bin(31)`
MQPTR	`pointer`
MQUINT8	`fixed bin(8)`

Tabella 460. Nomi e rappresentazioni dei tipi di dati PL/I (Continua)	
Tipo dati	**Rappresentazione**
MQUINT16	`fixed bin(16)`
MQUINT64	`fixed bin(64)`
MQULONG	`fixed bin(32)`

Dichiarazioni del tipo di dati High Level Assembler

Tabella 461. Nomi e rappresentazioni del tipo di dati assembler System/390	
Tipo dati	**Rappresentazione**
MQBOOL	`DS F`
MQBYTE	`DS XL1`
MQBYTE8	`DS XL8`
MQBYTE16	`DS XL16`
MQBYTE24	`DS XL24`
MQBYTE32	`DS XL32`
MQBYTE40	`DS XL40`
MQCAR	`DS CL1`
MQCHAR4	`DS CL4`
MQCHAR8	`DS CL8`
MQCHAR12	`DS CL12`

Tabella 461. Nomi e rappresentazioni del tipo di dati assembler System/390 (Continua)

Tipo dati	Rappresentazione
MQCHAR20	`DS CL20`
MQCHAR28	`DS CL28`
MQCHAR32	`DS CL32`
MQCHAR48	`DS CL48`
MQCHAR64	`DS CL64`
MQCHAR128	`DS CL128`
MQCHAR256	`DS CL256`
MQFLOAT32	`DS EB`
MQFLOAT64	`DS DB`
MQHCONN	`DS F`
MQHOBG	`DS F`
MQINT8	`DS XL1`
MQINT16	`DS H`
MQINT64	`DS D`
MQLONG	`DS F`
MQPTR	`DS F`

Tabella 461. Nomi e rappresentazioni del tipo di dati assembler System/390 (Continua)	
Tipo dati	**Rappresentazione**
MQUINT8	`DS XL1`
MQUINT16	`DS H`
MQUINT64	`DS D`
MQULONG	`DS F`

Tipi di dati della struttura

Un riepilogo dei tipi di dati della struttura, le regole per associare le strutture MQI in maniera coerente e le convenzioni utilizzate in ciascuna descrizione del tipo di dati della struttura.

- "Riepilogo dei tipi di dati della struttura utilizzati nelle chiamate MQI o nelle funzioni di uscita" a pagina 261
- "Riepilogo dei tipi di dati della struttura utilizzati nei dati del messaggio" a pagina 262
- "Regole per l'associazione coerente delle strutture MQI" a pagina 263
- "Convenzioni utilizzate in ciascuna descrizione del tipo di dati della struttura" a pagina 263

Riepilogo dei tipi di dati della struttura utilizzati nelle chiamate MQI o nelle funzioni di uscita

Tabella 462. Tipi di dati della struttura utilizzati nelle chiamate MQI o nelle funzioni di uscita		
Struttura	**Descrizione**	**Chiamate dove utilizzate**
MQACH	Intestazione catena uscita API	
MQAIR	Record di informazioni di autenticazione	MQCONNX
MQAXC	Contesto uscita API	
MQAXP	Parametro uscita API	
MQBMHO	Buffer per le opzioni di gestione del messaggio	MQBUFMH
MQBO	Opzioni di inizio	MQBEGIN
MQCBD	Descrittore callback	MQCB
MQCBO	Opzioni create - bag	Borsa mqCreate
MQCHARV	Stringa lunghezza variabile	MQINQMP
MQCNO	Opzioni di connessione	MQCONNX
MQCSP	Parametri di sicurezza	MQCONNX
MQCTLO	Opzioni di callback	MQCTL
MQDMPO	Opzioni della proprietà Elimina messaggio	MQDLTMP

Tabella 462. Tipi di dati della struttura utilizzati nelle chiamate MQI o nelle funzioni di uscita (Continua)		
Struttura	**Descrizione**	**Chiamate dove utilizzate**
MQGMO	Opzioni di acquisizione del messaggio	MQGET
MQIMPO	Opzioni della proprietà del messaggio di interrogazione	MQINQMP
MQMD	Descrittore messaggio	MQBUFMH, MQMHBUF, MQCB, MQGET, MQPUT, MQPUT1
MQMHBO	Gestione dei messaggi per le opzioni di buffer	MQMHBUF
MQOD	descrittore oggetto	MQOPEN, MQPUT1
MQOR	Record oggetto	MQOPEN, MQPUT1
MQPD	Descrittore proprietà	MQSETMP
MQPMO	Opzioni di inserimento del messaggio	MQPUT, MQPUT1
MQPMR	Record di inserimento del messaggio	MQPUT, MQPUT1
MQRR	Record di risposta	MQOPEN, MQPUT, MQPUT1
MQSCO	Opzioni di configurazione TLS	MQCONNX
MQSD	Descrittore sottoscrizione	MQSUB
MQSMPO	Imposta opzione proprietà messaggio	MQSETMP
MQSRO	Opzioni di richiesta di sottoscrizione	MQSUBRQ
MQSTS	Struttura di report di stato	MQSTAT

Riepilogo dei tipi di dati della struttura utilizzati nei dati del messaggio

Tabella 463. Tipi di dati della struttura utilizzati nei dati del messaggio	
Struttura	**Descrizione**
MQCIH	Intestazione informazioni CICS
MQCFH	Intestazione PCF
MQEPH	Intestazione PCF incorporata
MQDH	Intestazione della distribuzione
MQDLH	Intestazione lettera non recapitata (messaggio non consegnato)
MQIIH	Intestazione informazioni IMS
MQMDE	Estensione descrittore messaggio
MQRFH	Regole e intestazione di formattazione
MQRFH2	Regole e intestazione di formattazione 2
MQRMH	Intestazione messaggio di riferimento

Tabella 463. Tipi di dati della struttura utilizzati nei dati del messaggio (Continua)	
Struttura	**Descrizione**
MQTM	messaggio di trigger
MQTMC2	Messaggio trigger (formato carattere 2)
MQWIH	Intestazione informazioni sul lavoro
MQXQH	Intestazione coda di trasmissione

Nota: La struttura MQDXP (parametro di uscita conversione dati) è descritta in "Uscita conversione dati" a pagina 932, insieme alle chiamate di conversione dati associate.

Regole per l'associazione coerente delle strutture MQI

I linguaggi di programmazione variano nel loro livello di supporto per le strutture e alcune regole e convenzioni vengono adottate per mappare le strutture MQI in modo coerente in ciascun linguaggio di programmazione:

1. Le strutture devono essere allineate ai loro confini naturali.
   - La maggior parte delle strutture MQI richiede un allineamento a 4 byte.
   - Su IBM i, le strutture che contengono i puntatori richiedono l'allineamento a 16 byte; questi sono: MQCNO, MQOD, MQPMO.
2. Ogni campo in una struttura deve essere allineato sul suo limite naturale.
   - I campi con tipi di dati che corrispondono a MQLONG devono essere allineati su limiti di 4 byte.
   - I campi con tipi di dati equivalenti a MQPTR devono essere allineati sui limiti di 16 byte su IBM ie sui limiti di 4 byte in altri ambienti.
   - Altri campi sono allineati sui limiti di 1 byte.
3. La lunghezza di una struttura deve essere un multiplo del suo allineamento limite.
   - La maggior parte delle strutture MQI ha lunghezze che sono multipli di 4 byte.
   - Su IBM i, le strutture che contengono puntatori hanno lunghezze che sono multipli di 16 byte.
4. Se necessario, è necessario aggiungere byte o campi di riempimento per garantire la conformità con le regole precedenti.

Convenzioni utilizzate in ciascuna descrizione del tipo di dati della struttura

La descrizione di ogni tipo di dati della struttura comprende:

- Una panoramica dello scopo e dell'uso della struttura
- Descrizioni dei campi nella struttura, in una forma indipendente dal linguaggio di programmazione
- Esempi di come la struttura viene dichiarata in ciascuno dei linguaggi di programmazione supportati

La descrizione di ciascun tipo di dati della struttura contiene le seguenti sezioni:

Nome struttura
Il nome della struttura, seguito da un riepilogo dei campi nella struttura.

Panoramica
Una breve descrizione dello scopo e dell'uso della struttura.

Campi
Descrizioni dei campi. Per ogni campo, il nome del campo è seguito dal suo tipo di dati elementare tra parentesi (). Nel testo, i nomi dei campi vengono visualizzati utilizzando un carattere corsivo, ad esempio *Version*.

C'è anche una descrizione dello scopo del campo, insieme a un elenco di tutti i valori che il campo può prendere. I nomi delle costanti vengono visualizzati in maiuscolo, ad esempio MQGMO_STRUC_ID.

Una serie di costanti con lo stesso prefisso viene visualizzata utilizzando il carattere *, ad esempio: MQIA_ *.

Nelle descrizioni dei campi, vengono utilizzati i termini seguenti:

input
Si forniscono informazioni nel campo quando si effettua una chiamata.

output
Il gestore code restituisce le informazioni nel campo quando la chiamata viene completata o non riesce.

I/O
Le informazioni vengono fornite nel campo quando si effettua una chiamata e il gestore code le modifica quando la chiamata viene completata o non riesce.

Valori iniziali
Una tabella che mostra i valori iniziali per ciascun campo nei file di definizione dati forniti con MQI.

Dichiarazione C
Tipica dichiarazione della struttura in C.

Dichiarazione COBOL
Dichiarazione tipica della struttura in COBOL.

Dichiarazione PL/I
Tipica dichiarazione della struttura in PL/I.

Dichiarazione High Level Assembler
Dichiarazione tipica della struttura nel linguaggio assembler System/390 .

Dichiarazione Visual Basic
Tipica dichiarazione della struttura in Visual Basic.

Programmazione C

Informazioni che consentono di utilizzare l'MQI dal linguaggio di programmazione C.

File di intestazione

Tabella 464. File di intestazione C

File	Indice
CMQC	Prototipi di funzioni, tipi di dati e costanti denominate per MQI principale
CMQXC	Prototipi di funzione, tipi di dati e costanti denominate per l'exit di conversione dati
CMQEC	Prototipi di funzione, tipi di dati e costanti denominate per la MQI principale, l'uscita di conversione dati e la struttura dei punti di ingresso interfaccia (CMQEC include CMQXC e CMQC.)

Tabella 464. File di intestazione C (Continua)	
File	**Indice**
CMQSTRC	Funzioni che convertono le definizioni di costanti MQI nell'equivalente di testo. **Attenzione:** z/OS Applicabile a z/OS da IBM MQ 9.1. I programmi che utilizzano questo file di intestazione devono essere compilati con l'opzione del compilatore LONGNAME.

Per migliorare la portabilità delle applicazioni, codificare il nome del file di intestazione in minuscolo nella direttiva del preprocessore `#include` :

```
#include "cmqec.h"
```

Funzioni

Non è necessario specificare tutti i parametri passati per indirizzo ogni volta che si richiama una funzione.

- Passa i parametri che sono *solo input* e di tipo MQHCONN, MQHOBJ o MQLONG per valore.
- Passare tutti gli altri parametri per indirizzo.

Quando un particolare parametro non è richiesto, utilizzare un puntatore nullo come parametro sul richiamo della funzione, al posto dell'indirizzo dei dati del parametro. I parametri per i quali ciò è possibile sono identificati nelle descrizioni delle chiamate.

Non viene restituito alcun parametro come valore della funzione; nella terminologia C, ciò significa che tutte le funzioni restituiscono `void`.

Gli attributi della funzione sono definiti dalla variabile macro MQENTRY; il valore di questa variabile macro dipende dall'ambiente.

Parametri con tipo di dati non definito

Il parametro **`Buffer`** nelle funzioni MQGET, MQPUT e MQPUT1 ha un tipo di dati non definito. Questo parametro viene utilizzato per inviare e ricevere i dati del messaggio dell'applicazione.

I parametri di questo tipo vengono mostrati negli esempi C come array di MQBYTE. È possibile dichiarare i parametri in questo modo, ma di solito è più conveniente dichiararli come la struttura particolare che descrive il layout dei dati nel messaggio. Dichiarare il parametro della funzione effettiva come un puntatore a void e specificare l'indirizzo di qualsiasi tipo di dati come parametro sul richiamo della funzione.

Tipi di dati

Definire tutti i tipi di dati utilizzando l'istruzione C `typedef` . Per ciascun tipo di dati, definire anche il tipo di dati puntatore corrispondente. Il nome del tipo di dati del puntatore è il nome del tipo di dati elementari o della struttura preceduto dalla lettera P per indicare un puntatore. Definire gli attributi del puntatore utilizzando la variabile macro MQPOINTER; il valore di questa variabile macro dipende dall'ambiente. Quanto segue illustra come dichiarare i tipi di dati del puntatore:

```
#define MQPOINTER *               /* depends on environment */
...
typedef MQLONG MQPOINTER PMQLONG;  /* pointer to MQLONG */
typedef MQMD   MQPOINTER PMQMD;    /* pointer to MQMD */
```

Manipolazione di stringhe binarie

Dichiarare le stringhe di dati binari come uno dei tipi di dati MQBYTEn.

Ogni volta che si copiano, confrontano o si impostano campi di questo tipo, utilizzare le funzioni C **memcpy**, **memcmp**o **memset** ; ad esempio:

```
#include <string.h>
#include "cmqc.h"

MQMD MyMsgDesc;

memcpy(MyMsgDesc.MsgId,           /* set "MsgId" field to nulls    */
       MQMI_NONE,                 /* ...using named constant       */
       sizeof(MyMsgDesc.MsgId));

memset(MyMsgDesc.CorrelId,        /* set "CorrelId" field to nulls */
       0x00,                      /* ...using a different method   */
       sizeof(MQBYTE24));
```

Non utilizzare le funzioni stringa **strcpy**, **strcmp**, **strncpy**o **strncmp**, poiché non funzionano correttamente per i dati dichiarati con i tipi di dati MQBYTEn.

Manipolazione delle stringhe di caratteri

Quando il gestore code restituisce i dati carattere all'applicazione, il gestore code riempierà sempre i dati carattere con spazi vuoti fino alla lunghezza definita del campo. Il gestore code *non* restituisce stringhe con terminazione null.

Pertanto, durante la copia, il confronto o la concatenazione di tali stringhe, utilizzare le funzioni di stringa **strncpy**, **strncmp**o **strncat**.

Non utilizzare le funzioni di stringa che richiedono che la stringa termini con un valore null (**strcpy**, **strcmp**, **strcat**). Inoltre, non utilizzare la funzione **strlen** per determinare la lunghezza della stringa; utilizzare invece la funzione **sizeof** per determinare la lunghezza del campo.

Valori iniziali per le strutture

I file di intestazione definiscono varie variabili macro che è possibile utilizzare per fornire valori iniziali per le strutture MQ quando si dichiarano istanze di tali strutture.

Queste variabili macro hanno i nomi nel formato MQxxx_DEFAULT, dove MQxxx rappresenta il nome della struttura. Essi vengono utilizzati nel modo seguente:

```
MQMD   MyMsgDesc = {MQMD_DEFAULT};
MQPMO  MyPutOpts = {MQPMO_DEFAULT};
```

Per alcuni campi di caratteri (ad esempio, i campi *StrucId* che si verificano nella maggior parte delle strutture o il campo *Format* che si verifica in MQMD), MQI definisce valori particolari che sono validi. Per ogni valore valido, vengono fornite *due* variabili macro:

- Una variabile macro definisce il valore come una stringa con una lunghezza, escludendo le corrispondenze null implicite, esattamente la lunghezza definita del campo. Ad esempio, per il campo *Format* in MQMD viene fornita la seguente variabile macro (¬ rappresenta un singolo carattere vuoto):

  ```
  #define MQFMT_STRING "MQSTR¬¬¬"
  ```

  Utilizzare questo modulo con funzioni memcpy e memcmp .
- L'altra variabile macro definisce il valore come un array di caratteri; il nome di questa variabile macro è il nome del modulo stringa con suffisso _ARRAY. Ad esempio:

  ```
  #define MQFMT_STRING_ARRAY 'M','Q','S','T','R','¬','¬','¬'
  ```

  Utilizzare questo modulo per inizializzare il campo quando si dichiara un'istanza della struttura con valori diversi da quelli forniti dalla variabile macro MQMD_DEFAULT. (Questo non è sempre necessario; in alcuni ambienti è possibile utilizzare il formato stringa del valore in entrambe le situazioni. Tuttavia, è

possibile utilizzare il modulo dell'array per le dichiarazioni, poiché ciò è necessario per la compatibilità con il linguaggio di programmazione C + +.)

Valori iniziali per le strutture dinamiche

Quando è richiesto un numero variabile di istanze di una struttura, le istanze vengono generalmente create nella memoria principale ottenuta dinamicamente utilizzando le funzioni `calloc` o `malloc` . Per inizializzare i campi in tali strutture, considerare la seguente tecnica:

1. Dichiarare un'istanza della struttura utilizzando la variabile macro MQxxx_DEFAULT appropriata per inizializzare la struttura. Questa istanza diventa il modello per altre istanze:

   ```
   MQMD Model = {MQMD_DEFAULT}; /* declare model instance */
   ```

   Le parole chiave `static` o `auto` possono essere codificate nella dichiarazione per fornire all'istanza del modello una durata statica o dinamica, come richiesto.
2. Utilizzare le funzioni `calloc` o `malloc` per ottenere l'archiviazione per un'istanza dinamica della struttura:

   ```
   PMQMD Instance;
   Instance = malloc(sizeof(MQMD)); /* get storage for dynamic instance */
   ```
3. Utilizzare la funzione `memcpy` per copiare l'istanza del modello nell'istanza dinamica:

   ```
   memcpy(Instance,&Model,sizeof(MQMD)); /* initialize dynamic instance */
   ```

Utilizza da C++

Per il linguaggio di programmazione C + +, i file di intestazione contengono le seguenti istruzioni aggiuntive incluse solo quando si utilizza un compilatore C + +:

```
#ifdef __cplusplus
  extern "C" {
#endif

/* rest of header file */

#ifdef __cplusplus
  }
#endif
```

Convenzioni di notazione

Queste informazioni mostrano come richiamare le funzioni e dichiarare i parametri.

In alcuni casi, i parametri sono array con una dimensione non fissa. Per questi, viene utilizzata una n minuscola per indicare una costante numerica. Quando si codifica la dichiarazione per quel parametro, sostituire n con il valore numerico richiesto.

Programmazione COBOL

Informazioni che consentono di utilizzare MQI dal linguaggio di programmazione COBOL.

file COPY

Vengono forniti diversi file COPY per consentire la scrittura di programmi applicativi COBOL che utilizzano MQI. Esistono due file contenenti costanti denominate e due file per ognuna delle strutture.

Ogni struttura è fornita in due forme: una forma con valori iniziali e una forma senza:

- Utilizzare le strutture con valori iniziali nella WORKING - STORAGE SECTION di un programma COBOL; sono contenute in file COPY con nomi con suffisso la lettera V (for Values).
- Utilizzare le strutture senza valori iniziali nella `LINKAGE SECTION` di un programma COBOL; sono contenute in file COPY con nomi con il suffisso L (per Linkage).

I file COPY vengono riepiloghi nella seguente tabella. Non tutti i file elencati sono disponibili in tutti gli ambienti.

Tabella 465. File COBOL COPY

File (con valori iniziali)	File (senza valori iniziali)	Indice
CMQAIRV	RCPMAIRL	Record di informazioni di autenticazione
CMQBOV	CMQnota di carico	Struttura delle opzioni di inizio
CMQCIHV	CMQCIHL	Struttura dell'intestazione delle informazioni CICS
CMQCNOV	CMQCNOL	Struttura delle opzioni di collegamento
CMQDHV	CMQDHL	Struttura intestazione distribuzione
CMQDLHV	CMQDLHL	Struttura intestazione lettera non recapitabile
CMQDXPV	CMQDXPL	Struttura del parametro di uscita conversione dati
MMQGMOV	MMQGMOL	Richiama la struttura delle opzioni del messaggio
CMQIIHV	HL CMMQII	Struttura dell'intestazione delle informazioni IMS
MCMMDV	MDL CMMQ	Struttura descrittore del messaggio
CMQMDEV	CMQMDEL	Struttura di estensione del descrittore del messaggio
CMQMD1V	CMQMD1L	Struttura descrittore messaggi versione 1
CMQODV	CMQODL	Struttura descrittore oggetto
CMQORV	CMQORL	Struttura record oggetto
MCMPMOV	MMQPMOL	Struttura delle opzioni del messaggio di inserimento
CMQRFHV	CMQRFHL	Regole e struttura intestazione di formattazione
CMQRFH2V	CMQRFH2L	Regole e formattazione della struttura dell'intestazione versione 2
CMQRMHV	CMQRMHL	Struttura intestazione messaggio di riferimento
CMQRRV	CMQRRL	Struttura del record di risposta
CMQSCOV	CMQSCOL	Opzioni di configurazione TLS
MCMTMV	MMQTML	Struttura del messaggio trigger
CMQTMCV	CMQTMCL	Struttura del messaggio trigger (formato carattere)
CMQTMC2V	CMQTMC2L	Struttura del messaggio trigger (formato carattere) versione 2
CMQWIHV	CMQWIHL	Struttura intestazione informazioni di lavoro
CMQXQHV	CMQXQHL	Struttura intestazione coda di trasmissione

Tabella 465. File COBOL COPY (Continua)		
File (con valori iniziali)	**File (senza valori iniziali)**	**Indice**
CMQV	-	Costanti denominate per MQI principale
CMQXV	-	Costanti denominate per l'uscita di conversione dati
CMQMD2V	CMQMD2L	Struttura descrittore messaggio versione 2

Strutture

Nel file COPY, ogni dichiarazione di struttura inizia con un elemento level-10 ; ciò consente di dichiarare diverse istanze della struttura codificando la dichiarazione level-01 e quindi utilizzando l'istruzione COPY per copiare il resto della dichiarazione di struttura. Per fare riferimento all'istanza appropriata, utilizzare la parola chiave IN :

```
* Declare two instances of MQMD
 01  MY-MQMD.
     COPY CMQMDV.
 01  MY-OTHER-MQMD.
     COPY CMQMDV.
*
* Set MSGTYPE field in MY-OTHER-MQMD
     MOVE MQMT-REQUEST TO MQMD-MSGTYPE IN MY-OTHER-MQMD.
```

Allineare le strutture sui limiti appropriati. Se si utilizza l'istruzione COPY per includere una struttura che segue un elemento che non è l'elemento level-01 , assicurarsi che la struttura inizi all'offset appropriato dall'inizio dell'elemento level-01 . La maggior parte delle strutture MQI richiedono un allineamento a 4 byte; le eccezioni sono MQCNO, MQOD e MQPMO, che richiedono un allineamento a 16 byte su IBM i.

In questa sezione, i nomi dei campi nelle strutture vengono mostrati senza prefisso. In COBOL, i nomi dei campi sono preceduti dal nome della struttura seguito da un trattino. Tuttavia, se il nome della struttura termina con una cifra numerica, che indica che la struttura è una seconda o successiva versione della struttura originale, la cifra numerica viene omessa dal prefisso. I nomi dei campi in COBOL vengono visualizzati in maiuscolo (anche se, se necessario, è possibile utilizzare caratteri minuscoli o maiuscoli e minuscoli). Ad esempio, il campo *MsgType* descritto in “MQMD - Descrittore messaggi” a pagina 431 diventa MQMD - MSGTYPE in COBOL.

Le strutture del suffisso V vengono dichiarate con i valori iniziali per tutti i campi; è necessario impostare solo i campi in cui si desidera un valore diverso dal valore iniziale fornito.

Puntatori

Alcune strutture devono indirizzare dati facoltativi che potrebbero non essere contigui con la struttura, come ad esempio i record MQOR e MQRR indirizzati dalla struttura MQOD.

Per indirizzare questi dati facoltativi, le strutture contengono campi dichiarati con il tipo di dati puntatore. Tuttavia, COBOL non supporta il tipo di dati puntatore in tutti gli ambienti. Per questo motivo, i dati facoltativi possono essere indirizzati utilizzando campi che contengono l'offset dei dati dall'inizio della struttura.

Se si desidera trasferire un'applicazione tra ambienti, verificare se il tipo di dati puntatore è disponibile in tutti gli ambienti previsti. In caso contrario, l'applicazione deve indirizzare i dati facoltativi utilizzando i campi di scostamento invece dei campi puntatore.

Negli ambienti in cui i puntatori non sono supportati, dichiarare i campi del puntatore come stringhe di byte della lunghezza appropriata, con il valore iniziale che è la stringa di byte null. Non modificare questo valore iniziale se si utilizzano i campi di scostamento.

Costanti con nome

In queste informazioni, vengono visualizzati i nomi delle costanti contenenti il carattere di sottolineatura (_) come parte del nome. In COBOL, utilizzare il trattino (-) al posto del carattere di sottolineatura.

Le costanti che hanno valori stringa di caratteri utilizzano le virgolette singole come delimitatore di stringa ('). In alcuni ambienti, potrebbe essere necessario specificare un'opzione del compilatore appropriata per far sì che il compilatore accetti le virgolette singole come delimitatore di stringa al posto delle virgolette doppie.

Le costanti denominate vengono dichiarate nei file COPY come elementi level-10 . Per utilizzare le costanti, dichiarare l'elemento level-01 esplicitamente e quindi utilizzare l'istruzione COPY da copiare nelle dichiarazioni delle costanti:

```
* Declare a structure to hold the constants
 01  MY-MQ-CONSTANTS.
     COPY CMQV.
```

Il metodo precedente fa sì che le costanti occupino la memoria nel programma anche se non vi si fa riferimento. Se si includono le costanti in molti programmi separati all'interno della stessa unità di esecuzione, esistono più copie delle costanti, consumando inutilmente la memoria principale. Evitare questo effetto utilizzando una delle tecniche riportate di seguito:

- Aggiungere la clausola GLOBAL alla dichiarazione level-01 :

  ```
  * Declare a global structure to hold the constants
   01  MY-MQ-CONSTANTS GLOBAL.
       COPY CMQV.
  ```

  Questo assegna la memoria per una sola serie di costanti all'interno dell'unità di esecuzione. Le costanti, tuttavia, possono essere indicate da qualsiasi programma all'interno dell'unità di esecuzione, non solo dal programma che contiene la dichiarazione level-01 .

  Nota: la clausola GLOBAL non è supportata in tutti gli ambienti.
- Copiare manualmente in ciascun programma solo le costanti a cui fa riferimento tale programma. Non utilizzare l'istruzione COPY per copiare tutte le costanti nel programma.

Convenzioni di notazione

Le sezioni precedenti in questo argomento mostrano come richiamare chiamate e dichiarare parametri. In alcuni casi, i parametri sono tabelle o stringhe di caratteri la cui dimensione non è fissa. Per questi, viene utilizzata una n minuscola per indicare una costante numerica. Quando si codifica la dichiarazione per quel parametro, sostituire n con il valore numerico richiesto.

Programmazione High Level Assembler

Informazioni che consentono di utilizzare MQI dal linguaggio di programmazione Assembler System/390 .

Macro

Esistono due macro per le costanti denominate e una macro per ognuna delle strutture. Questi file sono riepilogati nella seguente tabella.

Tabella 466. Macro Assembler	
File	**Indice**
CMQA	Costanti denominate (equazioni) per MQI principale
CMQCIHA	Struttura dell'intestazione delle informazioni CICS
CMQCNOA	Struttura delle opzioni di collegamento
CMQDLHA	Struttura intestazione lettera non recapitabile
CMQDXPA	Struttura del parametro di uscita conversione dati
MMQGMOA	Richiama la struttura delle opzioni del messaggio
CMQIIHA	Struttura dell'intestazione delle informazioni IMS
MMQMDA	Struttura del descrittore del messaggio
CMQMDEA	Struttura di estensione del descrittore del messaggio
CMQODA	Struttura descrittore oggetto
MMQPMOA	Struttura delle opzioni del messaggio di inserimento
CMQRFHA	Regole e struttura intestazione di formattazione
CMQRFH2A	Regole e formattazione della struttura dell'intestazione versione 2
CMQRMHA	Struttura intestazione messaggio di riferimento
CMQTMA	Struttura del messaggio trigger
CMQTMC2A	Struttura del messaggio trigger (formato carattere) versione 2
MQVERA	Controllo versione struttura
CMQWIHA	Struttura intestazione informazioni di lavoro
CMQXA	Costanti denominate per l'uscita di conversione dati
CMQXPA	Struttura del parametro di uscita incrociata API
CMQXQHA	Struttura intestazione coda di trasmissione

Strutture

Le strutture sono generate da macro che hanno vari parametri per controllare l'azione della macro. Vedere “Strutture” a pagina 272

macro CMQVERA

Questa macro permette di impostare il valore predefinito da utilizzare per il parametro DCLVER sulle macro di struttura.

Il valore specificato da CMQVERA viene utilizzato dalla macro della struttura solo se si omette il parametro DCLVER dal richiamo della macro della struttura. Il valore predefinito viene impostato codificando la macro CMQVERA con il parametro DCLVER :

DCLVER=XX_ENCODE_CASE_ONE corrente
La versione predefinita è impostata sulla versione corrente (più recente).

DCLVER=SPECIFICATO
La versione predefinita è impostata sulla versione specificata dal parametro VERSION .

È necessario specificare il parametro **DCLVER** e il valore deve essere maiuscolo. Il valore impostato da CMQVERA rimane il valore predefinito fino al successivo richiamo di CMQVERA o alla fine dell'assembly. Se si omette CMQVERA, il valore predefinito è DCLVER=CURRENT.

Convenzioni di notazione

Altri argomenti mostrano come richiamare le chiamate e dichiarare i parametri. In alcuni casi, i parametri sono schiere o stringhe di caratteri con una dimensione non fissa per la quale viene utilizzata una n minuscola per rappresentare una costante numerica. Quando si codifica la dichiarazione per quel parametro, sostituire n con il valore numerico richiesto.

Strutture
Le strutture vengono generate da macro che hanno vari parametri per controllare l'azione della macro.

Nota: Di tanto in tanto vengono introdotte nuove versioni delle strutture IBM MQ . I campi aggiuntivi in una nuova versione possono far sì che una struttura che in precedenza era più piccola di 256 byte diventi più grande di 256 byte. Per questo motivo, scrivere istruzioni assembler destinate a copiare una struttura IBM MQ o impostare una struttura IBM MQ su null, per lavorare correttamente con strutture che potrebbero essere più grandi di 256 byte. In alternativa, utilizzare il parametro macro DCLVER o la macro CMQVERA con il parametro VERSION per dichiarare una specifica versione della struttura.

- "Specifica del nome della struttura" a pagina 272
- "Specifica del formato della struttura" a pagina 272
- "Controllo della versione della struttura" a pagina 273
- "Dichiarazione di una struttura incorporata all'interno di un'altra" a pagina 273
- "Specifica dei valori iniziali per i campi" a pagina 273
- "Controllo dell'elenco" a pagina 273

Specifica del nome della struttura

Per dichiarare più di un'istanza di una struttura, la macro prefissa il nome di ciascun campo nella struttura con una stringa specificabile dall'utente e un carattere di sottolineatura.

La stringa utilizzata è l'etichetta specificata sul richiamo della macro. Se non viene specificata alcuna etichetta, il nome della struttura viene utilizzato per creare il prefisso:

```
* Declare two object descriptors
         CMQODA  ,        Prefix used="MQOD_" (the default)
MY_MQOD  CMQODA  ,        Prefix used="MY_MQOD_"
```

Le dichiarazioni di struttura mostrate in questa sezione utilizzano il prefisso predefinito.

Specifica del formato della struttura

Le dichiarazioni di struttura possono essere generate dalla macro in uno dei due formati, controllati dal parametro DSECT :

DSECT = SÌ
Un'istruzione dell'assembler DSECT viene utilizzata per avviare una nuova sezione di dati; la definizione della struttura segue immediatamente l'istruzione DSECT . L'etichetta sul richiamo della macro viene utilizzata come nome della sezione dati; se non viene specificata alcuna etichetta, viene utilizzato il nome della struttura.

DSECT = NO
Le istruzioni dell'Assembler DC vengono utilizzate per definire la struttura nella posizione corrente nella routine. I campi vengono inizializzati con valori, che possono essere specificati codificando i parametri rilevanti sul richiamo della macro. I campi per cui non è specificato alcun valore nel richiamo della macro vengono inizializzati con valori predefiniti.

Il valore specificato deve essere maiuscolo. Se il parametro DSECT non viene specificato, viene utilizzato `DSECT = NO` .

Controllo della versione della struttura

Per impostazione predefinita, le macro dichiarano sempre la versione più recente di ciascuna struttura.

Sebbene sia possibile utilizzare il parametro della macro VERSION per specificare un valore per il campo *Version* nella struttura, tale parametro definisce il valore iniziale per il campo *Version* e non controlla la versione della struttura effettivamente dichiarata. Per controllare la versione della struttura dichiarata, utilizzare il parametro DCLVER :

DCLVER=XX_ENCODE_CASE_ONE corrente
La versione dichiarata è la versione corrente (più recente).

DCLVER=SPECIFICATO
La versione dichiarata è la versione specificata dal parametro VERSION . Se si omette il parametro VERSION , il valore predefinito è la versione 1.

Se si specifica il parametro VERSION , il valore deve essere una costante numerica a definizione automatica o la costante denominata per la versione richiesta (ad esempio, MQCNO_VERSION_3). Se si specifica un altro valore, la struttura viene dichiarata come se fosse stato specificato DCLVER=CURRENT , anche se il valore di VERSION si risolve in un valore valido.

Il valore specificato deve essere maiuscolo. Se si omette il parametro DCLVER , il valore utilizzato viene ricavato dalla variabile della macro globale MQDCLVER . È possibile impostare questa variabile utilizzando la macro CMQVERA.

Dichiarazione di una struttura incorporata all'interno di un'altra

Per dichiarare una struttura come componente di un'altra struttura, utilizzare il parametro NESTED :

NESTED=SÌ
La dichiarazione di struttura è nidificata in un'altra.

NESTED=NO
La dichiarazione di struttura non è nidificata in un'altra.

Il valore specificato deve essere maiuscolo. Se si omette il parametro NESTED , si presuppone NESTED=NO .

Specifica dei valori iniziali per i campi

Specificare il valore da utilizzare per inizializzare un campo in una struttura codificando il nome di tale campo (senza il prefisso) come parametro sul richiamo della macro, accompagnato dal valore richiesto.

Ad esempio, per dichiarare una struttura del descrittore del messaggio con il campo *MsgType* inizializzato con MQMT_REQUEST e il campo *ReplyToQ* inizializzato con la stringa "MY_REPLY_TO_QUEUE", utilizzare quanto segue:

```
MY_MQMD  CMQMDA  MSGTYPE=MQMT_REQUEST,                                X
               REPLYTOQ=MY_REPLY_TO_QUEUE
```

Se si specifica una costante con nome (equare) come valore sul richiamo della macro, utilizzare la macro CMQA per definire la costante con nome. Non racchiudere i valori della stringa di caratteri tra virgolette singole.

Controllo dell'elenco

Controllare l'aspetto della dichiarazione di struttura nell'elenco assembler utilizzando il parametro LIST :

ELENCO = SÌ
La dichiarazione di struttura viene visualizzata nell'elenco assembler.

ELENCO = NO
La dichiarazione di struttura non viene visualizzata nell'elenco assembler.

Il valore specificato deve essere maiuscolo. Se si omette il parametro LIST , viene utilizzato LIST = NO .

MQAIR - Record informazioni di autenticazione

La struttura MQAIR consente a una applicazione in esecuzione come IBM MQ MQI client di specificare le informazioni su un programma di autenticazione da utilizzare per la connessione client. La struttura è un parametro di input sulla chiamata MQCONNX.

Disponibilità

La struttura MQAIR è disponibile per i client seguenti:

- AIX
- Linux
- Windows

Serie di caratteri e codifica

I dati in MQAIR devono essere nella serie di caratteri e nella codifica del gestore code locale; tali dati sono forniti dall'attributo del gestore code **CodedCharSetId** e MQENC_NATIVE.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 467. Campi in MQAIR

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQAIR	'AIR¬'
Versione (numero versione struttura)	MQAIR_VERSION_1	1
AuthInfoTipo (tipo di informazioni di autenticazione)	LDAP CRL MQAIT_	1
AuthInfoConnName (nome connessione del server CRL LDAP)	Nessuna	Stringa null o spazi vuoti
LDAPUserNamePtr (indirizzo del nome utente LDAP)	Nessuna	Puntatore null o byte null
LDAPUserNameLDAPUserName (offset del nome utente LDAP dall'inizio di MQSCO)	Nessuna	0
LDAPUserNameLunghezza (lunghezza del nome utente LDAP)	Nessuna	0
LDAPPassword (password per accedere al server LDAP)	Nessuna	Stringa null o spazi vuoti
Nota: I restanti campi vengono ignorati se *Versione* è inferiore a MQAIR_VERSION_2.		
OCSPResponderURL (URL a cui è possibile contattare il responder OCSP)	Nessuna	Stringa null o spazi vuoti

Tabella 467. Campi in MQAIR (Continua)

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macroMQAIR_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQAIR MyAIR = {MQAIR_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQAIR

```
typedef struct tagMQAIR MQAIR;
struct tagMQAIR {
  MQCHAR4    StrucId;             /* Structure identifier */
  MQLONG     Version;             /* Structure version number */
  MQLONG     AuthInfoType;        /* Type of authentication
                                     information */
  MQCHAR264  AuthInfoConnName;    /* Connection name of CRL LDAP
                                     server */
  PMQCHAR    LDAPUserNamePtr;     /* Address of LDAP user name */
  MQLONG     LDAPUserNameOffset;  /* Offset of LDAP user name from start
                                     of MQAIR structure */
  MQLONG     LDAPUserNameLength;  /* Length of LDAP user name */
  MQCHAR32   LDAPPassword;        /* Password to access LDAP server */
  MQCHAR256  OCSPResponderURL;    /* URL of OCSP responder */

};
```

Dichiarazione COBOL per MQAIR

```
**   MQAIR structure
  10 MQAIR.
**    Structure identifier
   15 MQAIR-STRUCID            PIC X(4).
**    Structure version number
   15 MQAIR-VERSION            PIC S9(9) BINARY.
**    Type of authentication information
   15 MQAIR-AUTHINFOTYPE       PIC S9(9) BINARY.
**    Connection name of CRL LDAP server
   15 MQAIR-AUTHINFOCONNNAME   PIC X(264).
**    Address of LDAP user name
   15 MQAIR-LDAPUSERNAMEPTR    POINTER.
**    Offset of LDAP user name from start of MQAIR structure
   15 MQAIR-LDAPUSERNAMEOFFSET PIC S9(9) BINARY.
**    Length of LDAP user name
   15 MQAIR-LDAPUSERNAMELENGTH PIC S9(9) BINARY.
**    Password to access LDAP server
   15 MQAIR-LDAPPASSWORD       PIC X(32).
**    URL of OCSP responder
   15 MQAIR-OCSPRESPONDERURL   PIC X(256).
```

Dichiarazione Visual Basic per MQAIR

```
Type MQAIR
  StrucId            As String*4   'Structure identifier'
  Version            As Long       'Structure version number'
  AuthInfoType       As Long       'Type of authentication information'
  AuthInfoConnName   As String*264 'Connection name of CRL LDAP server'
  LDAPUserNamePtr    As MQPTR      'Address of LDAP user name'
```

```
  LDAPUserNameOffset As Long       'Offset of LDAP user name from start'
                                   'of MQAIR structure'
  LDAPUserNameLength As Long       'Length of LDAP user name'
  LDAPPassword       As String*32  'Password to access LDAP server'
End Type
```

StrucId (MQCHAR4) per MQAIR

Si tratta dell'identificatore della struttura del record delle informazioni di autenticazione. È sempre un campo di immissione. Il valore è MQAIR_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQAIR
Identificativo per il record delle informazioni di autenticazione.

Per il linguaggio di programmazione C, viene definita anche la costante MQAIR_STRUC_ID_ARRAY. Ha lo stesso valore di MQAIR_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQAIR

Questo è il numero di versione della struttura record delle informazioni di autenticazione. È sempre un campo di immissione.

Il valore deve essere uno dei seguenti.

MQAIR_VERSION_1
Record delle informazioni di autenticazione Version-1 .

Questo è il valore iniziale di questo campo.

MQAIR_VERSION_2
Record delle informazioni di autenticazione Version-2 .

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQAIR_CURRENT_
Versione corrente del record delle informazioni di autenticazione.

Tipo AuthInfo(MQLONG) per MQAIR

Questo è il tipo di informazioni di autenticazione contenute nel record.

Il valore può essere uno dei seguenti due parametri:

LDAP CRL MQAIT_
Controllo revoca certificato utilizzando il server LDAP.

OOCSP MQAIT
Controllo della revoca del certificato utilizzando OCSP.

Se il valore non è valido, la chiamata ha esito negativo con codice motivo MQRC_AUTH_INFO_TYPE_ERROR.

Questo è un campo di immissione. Il valore iniziale di questo campo è MQAIT_CRL_LDAP.

AuthInfoConnName (MQCHAR264) per MQAIR

Si tratta del nome host o dell'indirizzo di rete di un host su cui è in esecuzione il server LDAP. Questo può essere seguito da un numero di porta facoltativo, racchiuso tra parentesi. Il numero di porta predefinito è 389.

Se il valore è più breve della lunghezza del campo, terminare il valore con un carattere null o riempirlo con spazi vuoti fino alla lunghezza del campo. Se il valore non è valido, la chiamata ha esito negativo con codice motivo MQRC_AUTH_INFO_CONN_NAME_ERROR.

Questo è un campo di immissione. La lunghezza di questo campo viene fornita da MQ_AUTH_INFO_CONN_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e gli spazi in altri linguaggi di programmazione.

LDAPUserNamePtr (PMQCHAR) per MQAIR

Questo è il nome utente LDAP.

Consiste nel DN (Distinguished Name) dell'utente che sta tentando di accedere al server CRL LDAP. Se il valore è più breve della lunghezza specificata da *LDAPUserNameLength*, terminare il valore con un carattere null o riempirlo con spazi vuoti fino alla lunghezza *LDAPUserNameLength*. Il campo viene ignorato se *LDAPUserNameLength* è zero.

È possibile specificare il nome utente LDAP in uno dei modi seguenti:

- Utilizzando il campo puntatore *LDAPUserNamePtr*

  In questo caso, l'applicazione può dichiarare una stringa separata dalla struttura MQAIR e impostare *LDAPUserNamePtr* sull'indirizzo della stringa.

  Considerare l'utilizzo di *LDAPUserNamePtr* per i linguaggi di programmazione che supportano il tipo di dati del puntatore in un modo che sia portabile in ambienti differenti (ad esempio, il linguaggio di programmazione C).
- Utilizzando il campo offset *LDAPUserNameOffset*

  In questo caso, l'applicazione deve dichiarare una struttura composta contenente la struttura MQSCO seguita dall'array di record MQAIR seguito dalle stringhe del nome utente LDAP e impostare *LDAPUserNameOffset* sull'offset della stringa del nome appropriata dall'inizio della struttura MQAIR. Verificare che questo valore sia corretto e che abbia un valore che possa essere utilizzato all'interno di un MQLONG (il linguaggio di programmazione più restrittivo è COBOL, per cui l'intervallo valido è compreso tra -999 999 999 e +999 999 999).

  Considerare l'utilizzo di *LDAPUserNameOffset* per i linguaggi di programmazione che non supportano il tipo di dati puntatore o che implementano il tipo di dati puntatore in un modo che potrebbe non essere portabile in ambienti differenti (ad esempio, il linguaggio di programmazione COBOL).

Qualunque sia la tecnica scelta, utilizzare solo uno tra *LDAPUserNamePtr* e *LDAPUserNameOffset* ; la chiamata ha esito negativo con codice motivo MQRC_LDAP_USER_NAME_ERROR se entrambi sono diversi da zero.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e, in caso contrario, una stringa di byte completamente null.

Nota: Su piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata.

Offset LDAPUserName(MQLONG) per MQAIR

Questo è l'offset in byte del nome utente LDAP dall'inizio della struttura MQAIR.

L'offset può essere positivo o negativo. Il campo viene ignorato se *LDAPUserNameLength* è zero.

È possibile utilizzare *LDAPUserNamePtr* o *LDAPUserNameOffset* per specificare il nome utente LDAP, ma non entrambi; consultare la descrizione del campo *LDAPUserNamePtr* per i dettagli.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

LDAPUserNameLunghezza (MQLONG) per MQAIR

Questa è la lunghezza in byte del nome utente LDAP indicato dal campo *LDAPUserNamePtr* o *LDAPUserNameOffset* .

Il valore deve essere compreso tra zero e MQ_DISTINGUISHED_NAME_LENGTH. Se il valore non è valido, la chiamata ha esito negativo con codice motivo MQRC_LDAP_USER_NAME_LENGTH_ERR.

Se il server LDAP interessato non richiede un nome utente, impostare questo campo su zero.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

LDAPPassword (MQCHAR32) per MQAIR

Questa è la parola d'ordine necessaria per accedere al server CRL LDAP. Se il valore è più breve della lunghezza del campo, terminare il valore con un carattere null o riempirlo con spazi vuoti fino alla lunghezza del campo.

Se il server LDAP non richiede una password o si omette il nome utente LDAP, *LDAPPassword* deve essere null o vuoto. Se si omette il nome utente LDAP e *LDAPPassword* non è null o vuoto, la chiamata non riesce con codice motivo MQRC_LDAP_PASSWORD_ERROR.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da MQ_LDAP_PASSWORD_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e gli spazi in altri linguaggi di programmazione.

OCSPResponderURL (MQCHAR256) per MQAIR

Per una struttura MQAIR che rappresenta dettagli di connessione per un responder OCSP, questo campo contiene l'URL a cui è possibile contattare il responder.

Il valore di questo campo è un URL HTTP. Questo campo ha la precedenza su un URL in un'estensione del certificato AIA (AuthorityInfoAccess).

Il valore viene ignorato a meno che non siano vere entrambe le seguenti istruzioni:

- La struttura MQAIR è Versione 2 o successiva (il campo Versione è impostato su MQAIR_VERSION_2 o superiore).
- Il campo Tipo AuthInfoè impostato su MQAIT_OCSP.

Se il campo non contiene un URL HTTP nel formato corretto (e non viene ignorato), la chiamata MQCONNX ha esito negativo con codice motivo MQRC_OCSP_URL_ERROR.

Questo campo è sensibile al maiuscolo / minuscolo. Deve iniziare con la stringa http:// in minuscolo. Il resto dell'URL potrebbe essere sensibile al maiuscolo / minuscolo, a seconda dell'implementazione del server OCSP.

Questo campo non è soggetto alla conversione dei dati.

MQBMHO - Buffer per le opzioni di gestione dei messaggi

La struttura MQBMHO consente alle applicazioni di specificare le opzioni che controllano il modo in cui i gestori dei messaggi vengono prodotti dai buffer. La struttura è un parametro di input sulla chiamata MQBUFMH.

Serie di caratteri e codifica

I dati in MQBMHO devono essere nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (MQENC_NATIVE).

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 468. Campi in MQBMHO

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQBMHO_	'BMHO'
Versione (numero versione struttura)	MQBMHO_VERSION_1	1

Tabella 468. Campi in MQBMHO (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
Opzioni (opzioni che controllano l'azione di MQBMHO)	MQBMHO_NONE	0

Note:

1. Nel linguaggio di programmazione C, la variabile macroMQBMHO_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQBMHO MyBMHO = {MQBMHO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQBMHO

```
typedef struct tagMQBMHO MQBMHO;
struct tagMQBMHO {
  MQCHAR4  StrucId;        /* Structure identifier */
  MQLONG   Version;        /* Structure version number */
  MQLONG   Options;        /* Options that control the action of
                              MQBUFMH */
};
```

Dichiarazione COBOL per MQBMHO

```
**   MQBMHO structure
  10 MQBMHO.
**    Structure identifier
   15 MQBMHO-STRUCID               PIC X(4).
**    Structure version number
   15 MQBMHO-VERSION               PIC S9(9) BINARY.
**    Options that control the action of MQBUFMH
   15 MQBMHO-OPTIONS               PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQBMHO

```
Dcl
 1 MQBMHO based,
   3 StrucId        char(4),       /* Structure identifier */
   3 Version        fixed bin(31), /* Structure version number */
   3 Options        fixed bin(31), /* Options that control the action
                                      of MQBUFMH */
```

Dichiarazione High Level Assembler per MQBMHO

```
MQBMHO                  DSECT
MQBMHO_STRUCID          DS   CL4  Structure identifier
MQBMHO_VERSION          DS   F    Structure version number
MQBMHO_OPTIONS          DS   F    Options that control the
*                                  action of MQBUFMH
MQBMHO_LENGTH           EQU  *-MQBMHO
MQBMHO_AREA             DS   CL(MQBMHO_LENGTH)
```

StrucId (MQCHAR4) per MQBMHO per MQBMHO

Questo è l'identificatore della struttura del buffer per la struttura dell'handle del messaggio. È sempre un campo di immissione. Il suo valore è MQBMHO_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQBMHO_
Identificativo per il buffer nella struttura di gestione del messaggio.

Per il linguaggio di programmazione C, viene definita anche la costante MQBMHO_STRUC_ID_ARRAY. Ha lo stesso valore di MQBMHO_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQBMHO per MQBMHO
Questo è il numero di versione del buffer per la struttura dell'handle del messaggio. È sempre un campo di immissione.

Il valore deve essere:

MQBMHO_VERSION_1
Numero di versione per il buffer nella struttura dell'handle del messaggio.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQBMHO_CURRENT_
La versione corrente del buffer nella struttura dell'handle del messaggio.

Opzioni (MQLONG) per MQBMHO
Buffer per la struttura dell'handle del messaggio - Campo Opzioni

Il valore può essere:

MQBMHO_DELETE_PROPERTIES
Le proprietà aggiunte all'handle del messaggio vengono eliminate dal buffer. Se la chiamata ha esito negativo, non viene eliminata alcuna proprietà.

Opzioni predefinite: se non è necessaria l'opzione descritta, utilizzare la seguente opzione:

MQBMHO_NONE
Nessuna opzione specificata.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQBMHO_DELETE_PROPERTIES.

MQBNO - Opzioni di bilanciamento

La seguente tabella riepiloga i campi nella struttura.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 469. Campi in MQBNO

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQBNO_	'BNO¬'
Versione (numero versione struttura)	MQBNO_VERSION_1	1
ApplicationType (tipo di opzione di bilanciamento impostato nella struttura)	MQBNO_VALTYPE_SIMPLE	0
Timeout (timeout dopo il quale il ribilanciamento potrebbe interrompere l'attività dell'applicazione)	MQBNO_TIMEOUT_AS_DEFAULT	0
BalanceOptions (opzioni di bilanciamento impostate dall'applicazione emittente)	MQBNO_OPTIONS_NONE	0

Tabella 469. Campi in MQBNO (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macro MQBNO_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQBNO MyBNO = {MQBNO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQBNO

```
typedef struct tagMQBNO MQBNO;
struct tagMQBNO {
  MQCHAR4    StrucId;           /* Structure identifier */
  MQLONG     Version;           /* Structure version number */
  MQLONG     Type;              /* Type of balancing options set in the
                                   structure */
  MQLONG     Timeout;           /* Timeout after which re-balancing might
                                   interrupt application activity */
  MQLONG     BalanceOptions;    /* Balancing options set by the issuing
                                   application */
};
```

Dichiarazione COBOL per MQBNO

```
**   MQBNO structure
  10 MQBNO.
**    Structure identifier
   15 MQBNO-STRUCID          PIC X(4).
**    Structure version number
   15 MQBNO-VERSION          PIC S9(9) BINARY.
**    Type of balancing options set in the structure
   15 MQBNO-TYPE             PIC S9(9) BINARY.
**    Timeout after which re-balancing might interrupt application activity
   15 MQBNO-TIMEOUT          PIC S9(9) BINARY.
**    Balancing options set by the issuing application
   15 MQBNO-BALANCEOPTIONS          PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQBNO

```
dcl
 1 MQBNO based,
   3 StrucId          char(4),       /* Structure identifier */
   3 Version          fixed bin(31), /* Structure version number */
   3 Type             fixed bin(31), /* Type of balancing options set in the
                                        structure*/
   3 Timeout          fixed bin(31), /* Timeout after which re-balancing might
                                        interrupt application activity */
   3 BalanceOptions   fixed bin(31), /* Balancing options set by the issuing
                                        application*/
```

Riferimenti correlati

“MQCNO - Opzioni di connessione” a pagina 321

La struttura MQCNO consente all'applicazione di specificare le opzioni relative alla connessione al gestore code. La struttura è un parametro di input / output sulla chiamata MQCONNX.

StrucId (MQCHAR4) per MQBNO
Questo è l'identificativo della struttura delle opzioni di bilanciamento. È sempre un campo di immissione. Il valore iniziale è BNO.

Il valore deve essere:

BNO
Identificativo per la struttura delle opzioni di bilanciamento.

Per il linguaggio di programmazione C, viene definita anche la costante MQBNO_STRUC_ID_ARRAY. Questa costante ha lo stesso valore di BNO, ma è un array di caratteri invece di una stringa.

È necessario fornire un valore valido per **StrucId** o viene restituito MQRC_BNO_ERROR.

Versione (MQLONG) per MQBNO
Questo è un numero di versione della struttura delle opzioni di bilanciamento. È sempre un campo di immissione.

Il valore deve essere:

MQBNO_VERSION_1
Numero di versione per la struttura delle opzioni di bilanciamento.

È necessario fornire un valore valido per **Version** o viene restituito MQRC_BNO_ERROR.

ApplicationType (MQLONG) per MQBNO
Il tipo di opzione di bilanciamento impostato nella struttura.

I valori possibili sono:

MQBNO_BALTYPE_SIMPLE
Bilanciamento semplice; non vengono applicate regole specifiche oltre a quelle descritte in Influencing application re-balancing in uniform clusters.

REQREP MQBNO_BALTYPE_REQREP
Bilanciamento richiesta - risposta; dopo ogni chiamata MQPUT, è prevista una chiamata MQGET corrispondente per un messaggio di risposta. Il bilanciamento viene ritardato fino a quando non viene ricevuto un messaggio di questo tipo o fino a quando non viene superato il messaggio di richiesta SCADENZA.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQBNO_BALTYPE_SIMPLE.

È necessario fornire un solo valore per il campo **ApplicationType** oppure viene restituito MQRC_BNO_ERROR.

Nota: Un valore aggiuntivo per questo campo di MQBNO_BALTYPE_RA_MANAGED è riservato per l'utilizzo da parte di IBM MQ Resource Adapter per ambienti JEE. Anche se è un errore per un'applicazione fornire direttamente questo valore, può, ad esempio, essere riportato quando si interroga lo stato dell'applicazione.

Timeout (MQLONG) per MQBNO
Il **Timeout** dopo il quale il ribilanciamento potrebbe interrompere l'attività dell'applicazione.

I valori possibili sono:

MQBNO_TIMEOUT_AS_DEFAULT
Il valore di timeout predefinito impostato.

MQBNO_TIMEOUT_IMMEDIATE
Si verifica un timeout immediato.

MQBNO_TIMEOUT_NEVER
Non si verifica alcun timeout.

Il valore iniziale di questo campo è MQBNO_TIMEOUT_AS_DEFAULT.

È necessario fornire un solo valore dai valori definiti oppure un valore compreso tra 0 e 999999999 secondi per il campo **Timeout** o viene restituito MQRC_BNO_ERROR.

BalanceOptions (MQLONG) per MQBNO

Le opzioni di bilanciamento impostate dall'applicazione emittente.

I valori possibili sono:

MQBNO_OPTIONS_NONE
Nessuna opzione impostata

MQBNO_OPTIONS_IGNORE_TRANS
L'impostazione di questa opzione consente alle applicazioni di essere ribilanciate anche se nel mezzo di una transazione.

Il valore iniziale di questo campo è MQBNO_OPTIONS_NONE.

È possibile fornire qualsiasi combinazione dei valori definiti utilizzando il carattere o il carattere logico per il campo **BalanceOptions** . I valori non validi causano la restituzione di MQRC_BNO_ERROR.

MQBO - Opzioni di inizio

La struttura MQBO consente all'applicazione di specificare le opzioni relative alla creazione di un'unità di lavoro. La struttura è un parametro di input / output sulla chiamata MQBEGIN.

Disponibilità

La struttura MQBO è disponibile sulle piattaforme seguenti:

- AIX
- IBM i
- Linux
- Windows

La struttura MQBO non è disponibile per IBM MQ MQI clients.

Serie di caratteri e codifica

I dati in MQBO devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e la codifica del gestore code locale fornita da MQENC_NATIVE. Tuttavia, se l'applicazione è in esecuzione come client MQ MQI, la struttura deve essere nella serie di caratteri e nella codifica del client.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 470. Campi in MQBO per MQBO

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQBO	'BO¬¬'
Versione (numero versione struttura)	MQBO_VERSION_1	1
Opzioni (opzioni che controllano l'azione di MQBEGIN)	MQBO_NONE	0

Tabella 470. Campi in MQBO per MQBO (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macroMQBO_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQBO MyBO = {MQBO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQBO

```
typedef struct tagMQBO MQBO;
struct tagMQBO {
  MQCHAR4  StrucId;  /* Structure identifier */
  MQLONG   Version;  /* Structure version number */
  MQLONG   Options;  /* Options that control the action of MQBEGIN */
};
```

Dichiarazione COBOL per MQBO

```
**   MQBO structure
  10 MQBO.
**    Structure identifier
   15 MQBO-STRUCID PIC X(4).
**    Structure version number
   15 MQBO-VERSION PIC S9(9) BINARY.
**    Options that control the action of MQBEGIN
   15 MQBO-OPTIONS PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQBO

```
dcl
 1 MQBO based,
  3 StrucId char(4),       /* Structure identifier */
  3 Version fixed bin(31), /* Structure version number */
  3 Options fixed bin(31); /* Options that control the action of
                              MQBEGIN */
```

Dichiarazione Visual Basic per MQBO

```
Type MQBO
  StrucId As String*4 'Structure identifier'
  Version As Long     'Structure version number'
  Options As Long     'Options that control the action of MQBEGIN'
End Type
```

StrucId (MQCHAR4) per MQBO

Questo è l'identificativo della struttura delle opzioni iniziali. È sempre un campo di immissione. Il suo valore è MQBO_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQBO
Identificativo per la struttura delle opzioni iniziali.

Per il linguaggio di programmazione C, viene definita anche la costante MQBO_STRUC_ID_ARRAY. Ha lo stesso valore di MQBO_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQBO

Questo è il numero di versione della struttura di opzioni iniziali. È sempre un campo di immissione.

Il valore deve essere:

MQBO_VERSION_1
Numero di versione per la struttura delle opzioni di inizio.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQBO_CURRENT_
La versione corrente della struttura delle opzioni iniziali.

Opzioni (MQLONG) per MQBO

Questo campo è sempre un campo di input. Il valore iniziale è MQBO_NONE.

Il valore deve essere:

MQBO_NONE
Nessuna opzione specificata.

MQCBC - Contesto callback

La struttura di MQCBC viene utilizzata per specificare le informazioni di contesto inoltrate a una funzione di callback. La struttura è un parametro di input / output sulla chiamata a una routine del consumatore di messaggi.

Disponibilità

La struttura MQCBC è disponibile sulle piattaforme seguenti:

- AIX
- IBM i
- Linux
- Windows
- z/OS

e per IBM MQ MQI clients collegati a questi sistemi.

Versione

La versione corrente di MQCBC è MQCBC_VERSION_2.

Serie di caratteri e codifica

I dati in MQCBC devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da MQENC_NATIVE. Tuttavia, se l'applicazione è in esecuzione come un client MQI MQ , la struttura sarà nella serie di caratteri e nella codifica del client.

Campi

Non esistono valori iniziali per la struttura **MQCBC** . La struttura viene passata come un parametro ad una routine di callback. Il gestore code inizializza la struttura; le applicazioni non la inizializzano mai.

Note:

- Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.
- Non esistono valori iniziali per la struttura MQCBC. La struttura viene passata come un parametro ad una routine di callback. Il gestore code inizializza la struttura; le applicazioni non la inizializzano mai.

Tabella 471. Campi in MQCBC	
Campo	**Descrizione**
StrucID	Identificativo struttura
Versione	Numero di versione della struttura
CallType	Perché la funzione è stata richiamata
HOBJ	Handle di oggetti
CallbackArea	Campo per la funzione di callback da utilizzare
ConnectionArea	Campo per la funzione di callback da utilizzare
CompCode	Codice di completamento
Motivo	Codice di errore
Stato	Indicazione dello stato del consumatore attuale
DataLength	Lunghezza messaggio
BufferLength	Lunghezza del buffer di messaggi in byte
Indicatori	Indicatori generali
Nota: Il campo rimanente viene ignorato se la versione è inferiore a `MQCBC_VERSION_2`	
ReconnectDelay	Numero di millesimi di secondo prima del tentativo di riconnessione

Dichiarazioni di lingua

Dichiarazione C per MQCBC

```
typedef struct tagMQCBC MQCBC;
struct tagMQCBC {
   MQCHAR4   StrucId;               /* Structure identifier */
   MQLONG    Version;               /* Structure version number */
   MQLONG    CallType;              /* Why Function was called */
   MQHOBJ    Hobj;                  /* Object Handle */
   MQPTR     CallbackArea;          /* Callback data passed to the function */
   MQPTR     ConnectionArea;        /* MQCTL data area passed to the function */
   MQLONG    CompCode;              /* Completion Code */
   MQLONG    Reason;                /* Reason Code */
   MQLONG    State;                 /* Consumer State */
   MQLONG    DataLength;            /* Message Data Length */
   MQLONG    BufferLength;          /* Buffer Length */
   MQLONG    Flags;                 /* Flags containing information about
                                       this consumer */
  /* Ver:1 */
   MQLONG    ReconnectDelay;        /* Number of milliseconds before */
  /* Ver:2 */  };                  /* reconnect attempt */
```

Dichiarazione COBOL per MQCBC

```
** MQCBC structure
 10  MQCBC.
** Structure Identifier
 15  MQCBC-STRUCID                    PIC X(4).
** Structure Version
 15  MQCBC-VERSION                    PIC S9(9) BINARY.
 ** Call Type
```

```
 15  MQCBC-CALLTYPE                    PIC S9(9) BINARY.
** Object Handle
 15  MQCBC-HOBJ                       PIC S9(9) BINARY.
** Callback User Area
 15  MQCBC-CALLBACKAREA               POINTER
** Connection Area
 15  MQCBC-CONNECTIONAREA             POINTER
** Completion Code
 15  MQCBC-COMPCODE                   PIC S9(9) BINARY.
** Reason Code
 15  MQCBC-REASON                     PIC S9(9) BINARY.
** Consumer State
 15  MQCBC-STATE                      PIC S9(9) BINARY.
** Data Length
 15  MQCBC-DATALENGTH                 PIC S9(9) BINARY.
** Buffer Length
 15  MQCBC-BUFFERLENGTH               PIC S9(9) BINARY.
** Flags
 15  MQCBC-FLAGS                      PIC S9(9) BINARY.
** Ver:1 **
** Number of milliseconds before reconnect attempt
 15  MQCBC-RECONNECTDELAY PIC S9(9) BINARY.
** Ver:2 **
```

Dichiarazione PL/I per MQCBC

```
dcl
 1 MQCBC based,
   3 StrucId             char(4),       /* Structure identifier */
   3 Version             fixed bin(31), /* Structure version */
   3 CallType            fixed bin(31), /* Callback type */
   3 Hobj                fixed bin(31), /* Object Handle */
   3 CallbackArea        pointer,       /* User area passed to the function */
   3 ConnectionArea      pointer,       /* Connection User Area */
   3 CompCode            fixed bin(31); /* Completion Code */
   3 Reason              fixed bin(31); /* Reason Code */
   3 State               fixed bin(31); /* Consumer State */
   3 DataLength          fixed bin(31); /* Message Data Length */
   3 BufferLength        fixed bin(31); /* Message Buffer length */
   3 Flags               fixed bin(31); /* Consumer Flags */
/* Ver:1 */
   3 ReconnectDelay      fixed bin(31); /* Number of milliseconds before */
/* Ver:2 */                             /* reconnect attempt */
```

Dichiarazione High Level Assembler per MQCBC

```
    MQCBC                 DSECT
    MQCBC                 DS  0F    Force fullword alignment
    MQCBC_STRUCID         DS  CL4   Structure identifier
    MQCBC_VERSION         DS  F     Structure version number
    MQCBC_CALLTYPE        DS  F     Why Function was called
    MQCBC_HOBJ            DS  F     Object Handle
    MQCBC_CALLBACKAREA    DS  A     Callback data passed to the function
    MQCBC_CONNECTIONAREA  DS  A     MQCTL Data area passed to the function
    MQCBC_COMPCODE        DS  F     Completion Code
    MQCBC_REASON          DS  F     Reason Code
    MQCBC_STATE           DS  F     Consumer State
    MQCBC_DATALENGTH      DS  F     Message Data Length
    MQCBC_BUFFERLENGTH    DS  F     Buffer Length
    MQCBC_FLAGS           DS  F     Flags containing information about this consumer
    MQCBC_RECONNECTDELAY  DS  F     Number of milliseconds before reconnect
    MQCBC_LENGTH          EQU *-MQCBC
                          ORG       MQCBC
    MQCBC_AREA            DS  CL(MQCBC_LENGTH)
```

StrucId (MQCHAR4) per MQCBC

Questo è l'identificativo della struttura del contesto di callback. È sempre un campo di immissione. Il valore è MQCBC_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQCBC
Identificativo per la struttura del contesto di callback.

Per il linguaggio di programmazione C, viene definita anche la costante MQCBC_STRUC_ID_ARRAY. Ha lo stesso valore di MQCBC_STRUC_ID, ma è un array di caratteri anziché una stringa.

Versione (MQLONG) per MQCBC

Questo è il numero di versione della struttura del contesto di callback. È sempre un campo di immissione.

Il valore deve essere:

MQCBC_VERSION_1
Struttura del contesto di callback versione 1.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQCBC_CURRENT_
La versione corrente della struttura del contesto di callback.

La funzione di callback viene sempre passata all'ultima versione della struttura.

CallType (MQLONG) per MQCBC

Campo contenente informazioni sul motivo per cui questa funzione è stata richiamata; vengono definiti i seguenti valori.

Tipi di chiamata di consegna del messaggio: questi tipi di chiamata contengono informazioni su un messaggio. I parametri **DataLength** e **BufferLength** sono validi per questi tipi di chiamata.

MQCBC_MSG_REMOVED
La funzione del destinatario del messaggio è stata richiamata con un messaggio che è stato eliminato in maniera distruttiva dall'handle dell'oggetto.

Se il valore di *CompCode* è MQCC_WARNING, il valore del campo *Reason* è MQRC_TRUNCATED_MSG_ACCEPTED o uno dei codici che indicano un problema di conversione dati.

MQCBCT_MSG_NOT_REMOVED
La funzione del destinatario del messaggio è stata richiamata con un messaggio che non è stato ancora eliminato in maniera distruttiva dall'handle dell'oggetto. Il messaggio può essere eliminato in modo distruttivo dall'handle dell'oggetto utilizzando *MsgToken*.

Il messaggio potrebbe non essere stato rimosso perché:

- Le opzioni MQGMO hanno richiesto un'operazione di esplorazione, MQGMO_BROWSE_ *
- Il messaggio è più grande del buffer disponibile e le opzioni MQGMO non specificano MQGMO_ACCEPT_TRUNCATED_MSG

Se il valore di *CompCode* è MQCC_WARNING, il valore del campo *Reason* è MQRC_TRUNCATED_MSG_FAILED o uno dei codici che indicano un problema di conversione dati.

Tipi di chiamata di controllo callback: questi tipi di chiamata contengono informazioni sul controllo del callback e non contengono dettagli su un messaggio. Questi tipi di chiamata sono richiesti utilizzando Opzioni nella struttura MQCBD.

I parametri **DataLength** e **BufferLength** non sono validi per questi tipi di chiamata.

MQCBC_REGISTER_CALL
Lo scopo di questo tipo di chiamata è consentire alla funzione di callback di eseguire alcune impostazioni iniziali.

La funzione di richiamata viene richiamata immediatamente dopo la registrazione del callback, ossia al ritorno da una chiamata MQCB utilizzando un valore per il campo *Operation* di MQOP_REGISTER.

Questo tipo di chiamata viene utilizzato sia per i consumer di messaggi che per i gestori eventi.

Se richiesto, questo è il primo richiamo della funzione di callback.

Il valore del campo *Reason* è MQRC_NONE.

MQCBCT_START_CALL
Lo scopo di questo tipo di chiamata è quello di consentire alla funzione di callback di eseguire alcune operazioni di impostazione quando viene avviata, ad esempio, reintegrando le risorse che erano state ripulite quando era stata precedentemente arrestata.

La funzione di callback viene richiamata quando la connessione viene avviata utilizzando MQOP_START o MQOP_START_WAIT.

Se una funzione di callback è registrata all'interno di un'altra funzione di callback, questo tipo di chiamata viene richiamato quando viene restituito il callback.

Questo tipo di chiamata viene utilizzato solo per gli utenti del messaggio.

Il valore del campo *Reason* è MQRC_NONE.

MQCBCT_STOP_CALL
Lo scopo di questo tipo di chiamata è quello di consentire alla funzione di callback di eseguire una ripulitura quando viene arrestata per un certo periodo, ad esempio, ripulendo le risorse aggiuntive che sono state acquisite durante l'utilizzo dei messaggi.

La funzione di richiamata viene richiamata quando una chiamata MQCTL viene emessa utilizzando un valore per il campo di *Operation* di MQOP_STOP.

Questo tipo di chiamata viene utilizzato solo per gli utenti del messaggio.

Il valore del campo *Reason* è impostato per indicare il motivo dell'arresto.

MQCBC_DEREGISTER_CALL
Lo scopo di questo tipo di chiamata è di consentire alla funzione di callback di eseguire la ripulitura finale alla fine del processo di consumo. La funzione callback viene richiamata quando:

- La funzione di callback viene annullata utilizzando una chiamata MQCB con MQOP_DEREGISTER.
- La coda è chiusa, causando l'annullamento implicito della registrazione. In questa istanza la funzione di callback viene passata MQHO_UNUSABLE_HOBJ come handle dell'oggetto.
- La chiamata MQDISC viene completata - causando una chiusura implicita e, quindi, un annullamento della registrazione. In questo caso, la connessione non viene disconnessa immediatamente e non viene ancora eseguito il commit di tutte le transazioni in corso.

Se una di queste azioni viene eseguita all'interno della funzione di callback, l'azione viene richiamata una volta restituita la callback.

Questo tipo di chiamata viene utilizzato sia per i consumer di messaggi che per i gestori eventi.

Se richiesto, questo è l'ultimo richiamo della funzione di callback.

Il valore del campo *Reason* è impostato per indicare il motivo dell'arresto.

MQCBCT_EVENT_CALL
Funzione del gestore eventi

La funzione del gestore eventi è stata richiamata senza un messaggio quando il gestore code o la connessione vengono arrestati o disattivati.

Questa chiamata può essere utilizzata per eseguire l'azione appropriata per tutte le funzioni di callback.

Funzione consumatore messaggi

La funzione dell'utente del messaggio è stata richiamata senza un messaggio quando è stato rilevato un errore (*CompCode* = MQCC_FAILED) specifico per l'handle dell'oggetto; ad esempio, *Reason* code = MQRC_GET_INHIBITED.

Il valore del campo *Reason* è impostato per indicare il motivo della chiamata.

MQCBCT_MC_EVENT_CALL

La funzione del gestore eventi è stata richiamata per gli eventi multicast; al gestore eventi vengono inviati IBM MQ eventi multicast invece di eventi 'normali' IBM MQ .

Per ulteriori informazioni su MQCBCT_MC_EVENT_CALL, consultare Report di eccezioni multicast.

Hobj (MQHOBJ) per MQCBC
Questo è l'handle dell'oggetto per chiamate al destinatario del messaggio.

Per un gestore eventi, questo valore è MQHO_NONE

L'applicazione può utilizzare questo handle e il token del messaggio nel blocco Get Message Options per richiamare il messaggio se un messaggio non è stato rimosso dalla coda.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQHO_UNUSABLE_HOBJ

CallbackArea (MQPTR) per MQCBC
Questo campo è disponibile per la funzione di callback da utilizzare.

Il gestore code non prende alcuna decisione in base al contenuto di questo campo e viene passato non modificato dal campo CallbackArea nella struttura MQCBD, che è un parametro sulla chiamata MQCB utilizzata per definire la funzione di callback.

Le modifiche a *CallbackArea* vengono conservate nelle chiamate della funzione di callback per un *HObj*. Questo campo non è condiviso con funzioni di callback per altri handle.

Questo è un campo di immissione / emissione per la funzione di callback. Il valore iniziale di questo campo è un puntatore null o byte null.

ConnectionArea (MQPTR) per MQCBC
Questo campo è disponibile per la funzione di callback da utilizzare.

Il gestore code non prende alcuna decisione in base al contenuto di questo campo e viene trasmesso non modificato dal campo ConnectionArea nella struttura MQCTLO, che è un parametro sulla chiamata MQCTL utilizzata per controllare la funzione di callback.

Tutte le modifiche apportate a questo campo dalle funzioni di callback vengono conservate nei richiami della funzione di callback. Questa area può essere utilizzata per passare informazioni che devono essere condivise da tutte le funzioni di callback. A differenza di *CallbackArea*, questa area è comune a tutte le richiamate per un handle di connessione.

Questo è un campo di immissione e di emissione. Il valore iniziale di questo campo è un puntatore null o byte null.

CompCode (MQLONG) per MQCBC
Questo campo è il codice di completamento. Indica se si sono verificati problemi durante l'utilizzo del messaggio.

Il valore è uno dei seguenti:

MQCC_OK
Completamento riuscito

MQCC_AVVERTENZA
Avvertenza (completamento parziale)

MQCC_NON RIUSCITO
Chiamata non riuscita

Questo è un campo di immissione. Il valore iniziale di questo campo è MQCC_OK.

Motivo (MQLONG) per MQCBC
Questo è il codice di errore che qualifica *CompCode*.

Questo è un campo di immissione. Il valore iniziale di questo campo è MQRC_NONE.

Stato (MQLONG) per MQCBC

Un'indicazione dello stato del consumatore corrente. Questo campo è di maggior valore per un'applicazione quando un codice di errore diverso da zero viene passato alla funzione consumer.

È possibile utilizzare questo campo per semplificare la programmazione dell'applicazione poiché non è necessario codificare il comportamento per ciascun codice di errore.

Questo è un campo di immissione. Il valore iniziale di questo campo è MQCS_NONE

Tabella 472.

Stato	**Azione gestore code**	**Valore della costante**
MQCS_NONE Questo codice di errore rappresenta una normale chiamata senza ulteriori informazioni sul motivo	Nessuna; questa è l'operazione normale.	0
MQCS_SUSPENDED_TEMPORARY Questi codici di errore rappresentano condizioni temporanee.	La routine di callback viene richiamata per segnalare la condizione e quindi sospesa. Dopo un periodo di tempo, il sistema potrebbe tentare nuovamente l'operazione, il che potrebbe portare alla riattivazione della stessa condizione.	1
MQCS_SUSPENDED_USER_ACTION Questi codici di errore rappresentano le condizioni in cui il callback deve eseguire un'azione per risolvere la condizione.	Il consumer è sospeso e la routine di callback viene richiamata per notificare la condizione. La routine di callback deve risolvere la condizione se possibile e RESUME o chiudere la connessione.	2
MQCS_SUSPENDED Questi codici di errore rappresentano errori che impediscono ulteriori callback del messaggio.	Il gestore code sospende automaticamente la funzione di callback. Se la funzione di richiamata viene ripresa, è probabile che riceva di nuovo lo stesso codice di errore.	3
MQCS_STOPPED Questi codici di errore rappresentano la fine del consumo del messaggio.	Consegnato al gestore delle eccezioni e ai callback che specificano MQCBDO_STOP_CALL. Non è possibile utilizzare ulteriori messaggi.	4

DataLength (MQLONG) per MQCBC

Questa è la lunghezza in byte dei dati dell'applicazione nel messaggio. Se il valore è zero, significa che il messaggio non contiene dati dell'applicazione.

Il campo DataLength contiene la lunghezza del messaggio, ma non necessariamente la lunghezza dei dati del messaggio passati al consumer. È possibile che il messaggio sia stato troncato. Utilizzare il campo ReturnedLength in MQGMO per determinare la quantità di dati effettivamente passati al consumer.

Se il codice di errore indica che il messaggio è stato troncato, è possibile utilizzare il campo DataLength per determinare la dimensione effettiva del messaggio. Ciò consente di determinare la dimensione del buffer richiesto per contenere i dati del messaggio ed emettere una chiamata MQCB per aggiornare la MaxMsgLength con un valore appropriato.

Se viene specificata l'opzione MQGMO_CONVERT, il messaggio convertito potrebbe essere maggiore del valore restituito per DataLength. In tali casi, è probabile che l'applicazione debba emettere una chiamata MQCB per aggiornare la MaxMsgLength in modo che sia maggiore del valore restituito dal gestore code per DataLength.

Per evitare problemi di troncamento dei messaggi, specificare MaxMsgLength come MQCBD_FULL_MSG_LENGTH. Ciò fa sì che il gestore code assegni un buffer per la lunghezza completa del messaggio dopo la conversione dei dati. Tenere presente, tuttavia, che anche se questa opzione viene specificata, è comunque possibile che non sia disponibile memoria sufficiente per elaborare correttamente la richiesta. Le applicazioni devono sempre controllare il codice motivo restituito. Ad esempio, se non è possibile assegnare memoria sufficiente per convertire il messaggio, i messaggi vengono restituiti all'applicazione non convertita.

Questo è un campo di input per la funzione del consumer del messaggio; non è rilevante per una funzione del gestore eventi.

BufferLength (MQLONG) per MQCBC

Questo campo è la lunghezza in byte del buffer di messaggi che è stato passato a questa funzione.

Il buffer può essere maggiore sia del valore di lunghezza MaxMsgdefinito per il consumer che del valore ReturnedLength in MQGMO.

La lunghezza effettiva del messaggio viene fornita nel campo DataLength .

L'applicazione può utilizzare l'intero buffer per i propri scopi per la durata della funzione di callback.

Questo è un campo di input per la funzione del destinatario del messaggio; non è rilevante per una funzione del gestore eccezioni.

Indicatori (MQLONG) per MQCBC

Indicatori contenenti informazioni su questo consumer.

È definita la seguente opzione:

MQCBCF_READA_BUFFER_EMPTY

Questo indicatore può essere restituito se una precedente chiamata MQCLOSE che utilizza l'opzione MQQUIESCE non è riuscita con un codice motivo MQRC_READ_AHEAD_MSGS.

Questo codice indica che viene restituito l'ultimo messaggio di lettura anticipata e che il buffer è ora vuoto. Se l'applicazione emette un'altra chiamata MQCLOSE utilizzando l'opzione MQCO_QUIESCE), l'operazione ha esito positivo.

Notare che non è garantito che a un'applicazione venga fornito un messaggio con questo indicatore impostato, poiché potrebbero essere ancora presenti messaggi nel buffer di lettura anticipata che non corrispondono ai criteri di selezione correnti. In questa istanza, la funzione consumer viene richiamata con il codice motivo MQRC_HOBJ_QUIESCED.

Se il buffer di lettura anticipata è completamente vuoto, il consumer viene richiamato con l'indicatore MQCBCF_READA_BUFFER_EMPTY e il codice motivo MQRC_HOBJ_QUIESCED_NO_MSGS.

Questo è un campo di input per la funzione del consumer del messaggio; non è rilevante per una funzione del gestore eventi.

ReconnectDelay (MQLONG) per MQCBC

ReconnectDelay indica il tempo di attesa del gestore code prima di tentare la riconnessione. Il campo può essere modificato da un gestore eventi per modificare il ritardo o arrestare completamente la riconnessione.

Utilizzare il campo `ReconnectDelay` solo se il valore del campo `Motivo` nel contesto di callback è `MQRC_RECONNECTING`.

All'immissione nel gestore eventi, il valore di `ReconnectDelay` è il numero di millisecondi che il gestore code attenderà prima di effettuare un tentativo di riconnessione. Tabella 473 a pagina 293 elenca i valori che è possibile impostare per modificare il comportamento del gestore code al ritorno dal gestore eventi.

Tabella 473. Valori di ReconnectDelay		
Nome	**Valore**	**Descrizione**
MQRD_NO_RECONNECT	-1	Non effettuare ulteriori tentativi di riconnessione. Viene restituito un errore all'applicazione.
MQRD_NO_DELAY	0	Provare a riconnettersi immediatamente.
Milliseconds	>0	Attendere questo numero di millisecondi prima di ritentare la connessione.

MQCBD - Descrittore di callback

La struttura MQCBD viene utilizzata per specificare una funzione di callback e le opzioni che ne controllano l'utilizzo da parte del gestore code. La struttura è un parametro di immissione nella chiamata MQCB.

Disponibilità

La struttura MQCBD è disponibile sulle piattaforme seguenti:

- AIX
- IBM i
- Linux
- Windows
- z/OS

e per IBM MQ MQI clients collegati a questi sistemi.

Versione

La versione corrente di MQCBD è MQCBD_VERSION_1.

Serie di caratteri e codifica

I dati in MQCBD devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da MQENC_NATIVE. Tuttavia, se l'applicazione è in esecuzione come client MQ MQI, la struttura deve essere nella serie di caratteri e nella codifica del client.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 474. Campi in MQCBD		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
StrucID (identificativo della struttura)	ID_STRUC_MQCBD	'CBD¬'
Versione (numero versione struttura)	MQCBD_VERSION_1	1

Tabella 474. Campi in MQCBD (Continua)

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
CallbackType (tipo di funzione di callback)	MQCBT_MESSAGE_CONSUMER	1
Opzioni (opzioni che controllano l'utilizzo dei messaggi)	MQCBDO_NONE	0
CallbackArea (campo per la funzione di callback da utilizzare)	Nessuna	Puntatore null o spazi null
CallbackFunction (se la funzione viene richiamata come chiamata API)	Nessuna	Puntatore null o spazi null
CallbackName (se la funzione viene richiamata come un programma collegato dinamicamente)	Nessuna	Stringa null o spazi vuoti
MaxMsgLength (lunghezza del messaggio più lungo che può essere letto)	MQCBD_FULL_MSG_LENGTH	-1

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Il valore Stringa nulla o spazi vuoti indica la stringa nulla nel linguaggio di programmazione C e i caratteri vuoti in altri linguaggi di programmazione.
3. Nel linguaggio di programmazione C, la variabile macroMQCBD_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQCBD MyCBD = {MQCBD_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQCBD

```
typedef struct tagMQCBD MQCBD;
 struct tagMQCBD {
   MQCHAR4    StrucId;                /* Structure identifier */
   MQLONG     Version;                /* Structure version number */
   MQLONG     CallBackType;           /* Callback function type */
   MQLONG     Options;                /* Options controlling message
                                         consumption */
   MQPTR      CallbackArea;           /* User data passed to the function */
   MQPTR      CallbackFunction;       /* Callback function pointer */
   MQCHAR128  CallbackName;           /* Callback name */
   MQLONG     MaxMsgLength;           /* Maximum message length */
 };
```

Dichiarazione COBOL per MQCBD

```
** MQCBCD structure
 10  MQCBD.
** Structure Identifier
 15  MQCBD-STRUCID                    PIC X(4).
** Structure Version
 15  MQCBD-VERSION                    PIC S9(9) BINARY.
 ** Callback Type
 15  MQCBD-CALLBACKTYPE               PIC S9(9) BINARY.
** Options
 15  MQCBD-OPTIONS                    PIC S9(9) BINARY.
```

```
** Callback User Area
 15  MQCBD-CALLBACKAREA                POINTER
** Callback Function Pointer
 15  MQCBD-CALLBACKFUNCTION            FUNCTION-POINTER
** Callback Program Name
 15  MQCBD-CALLBACKNAME                PIC X(128)
** Maximum Message Length
 15  MQCDB-MAXMSGLENGTH                PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQCBD

```
dcl
 1 MQCBD based,
   3 StrucId             char(4),       /* Structure identifier*/
   3 Version             fixed bin(31), /* Structure version*/
   3 CallbackType        fixed bin(31), /* Callback function type */
   3 Options             fixed bin(31), /* Options */
   3 CallbackArea        pointer,       /* User area passed to the function */
   3 CallbackFunction    pointer,       /* Callback Function Pointer */
   3 CallbackName        char(128),     /* Callback Program Name */
   3 MaxMsgLength        fixed bin(31); /* Maximum Message Length */
```

StrucId (MQCHAR4) per MQCBD

Questo è l'identificativo della struttura del descrittore callback. È sempre un campo di immissione. Il valore è MQCBD_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQCBD
Identificativo per la struttura del descrittore di callback.

Per il linguaggio di programmazione C, viene definita anche la costante MQCBD_STRUC_ID_ARRAY. Ha lo stesso valore di MQCBD_STRUC_ID, ma è un array di caratteri anziché una stringa.

Versione (MQLONG) per MQCBD

Questo è il numero di versione della struttura descrittore di callback. È sempre un campo di immissione.

Il valore deve essere:

MQCBD_VERSION_1
Struttura del descrittore di callback versione 1.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQCBD_CURRENT_
Versione corrente della struttura del descrittore di callback.

CallbackType (MQLONG) per MQCBD

Struttura descrittore callback - Campo CallbackType

È il tipo di funzione di callback. Il valore deve essere uno tra:

MQCBT_MESSAGE_CONSUMER
Definisce questo callback come funzione di consumatore di messaggi.

Una funzione di callback del destinatario del messaggio viene chiamata quando un messaggio, che soddisfa i criteri di selezione specificati, è disponibile su un handle dell'oggetto e la connessione viene avviata.

HANDLER EVENTO MQCBT_
Definisce questo callback come routine di eventi asincroni; non viene utilizzato per utilizzare i messaggi per un handle.

Hobj non è richiesto nella chiamata MQCB che definisce il gestore eventi e viene ignorato se specificato.

Il gestore eventi viene chiamato per le condizioni che influenzano l'intero ambiente del consumatore di messaggi. La funzione consumer viene richiamata senza un messaggio quando si verifica un evento,

ad esempio un gestore code o l'arresto della connessione o la sospensione. Non viene richiamata per le condizioni specifiche di un singolo utente di messaggi, ad esempio, MQRC_GET_INHIBITED.

Gli eventi vengono consegnati all'applicazione, indipendentemente dal fatto che la connessione sia stata avviata o arrestata, tranne che nei seguenti ambienti:

- Ambiente CICS su z/OS
- applicazioni senza thread

Se il chiamante non inoltra uno di questi valori, la chiamata ha esito negativo con un codice *Reason* di MQRC_CALLBACK_TYPE_ERROR

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQCBT_MESSAGE_CONSUMER.

Opzioni (MQLONG) per MQCBD

Struttura descrittore callback - Campo Opzioni

È possibile specificare una o più di queste opzioni. Per specificare più di un'opzione, aggiungere i valori insieme (non aggiungere la stessa costante più di una volta) o combinare i valori utilizzando l'operazione OR bit per bit (se il linguaggio di programmazione supporta le operazioni bit).

MQCBDO_FAIL_IF_QUIESCING
La chiamata MQCB ha esito negativo se il gestore code è in stato di inattività.

Su z/OS, questa opzione forza anche la mancata riuscita della chiamata MQCB se la connessione (per un'applicazione CICS o IMS) è in stato di inattività.

Specificare MQGMO_FAIL_IF_QUIESCING, nelle opzioni MQGMO trasmesse sulla chiamata MQCB, per causare la notifica ai consumatori di messaggi quando sono in fase di sospensione.

Opzioni di controllo: le seguenti opzioni controllano se la funzione di callback viene richiamata, senza un messaggio, quando lo stato del consumer cambia:

REGISTER_MQCBDO_CALL
La funzione callback viene richiamata con il tipo di chiamata MQCBCT_REGISTER_CALL.

MQCBDO_START_CALL
La funzione callback viene richiamata con il tipo di chiamata MQCBCT_START_CALL.

CALL MQCBDO_STOP_
La funzione callback viene richiamata con il tipo di chiamata MQCBCT_STOP_CALL.

MQCBDO_DEREGISTER_CALL
La funzione callback viene richiamata con il tipo di chiamata MQCBCT_DEREGISTER_CALL.

MQCBDO_EVENT_CALL
La funzione callback viene richiamata con il tipo di chiamata MQCBCT_EVENT_CALL.

MQCBDO_MC_EVENT_CALL
La funzione callback viene richiamata con il tipo di chiamata MQCBCT__MC_EVENT_CALL.

Per ulteriori dettagli su questi tipi di chiamata, consultare CallType .

Opzione predefinita: se non è necessaria alcuna delle opzioni descritte, utilizzare la seguente opzione:

MQCBDO_NONE
Utilizzare questo valore per indicare che non sono state specificate altre opzioni; tutte le opzioni assumono i propri valori predefiniti.

MQCBDO_NONE è definito per aiutare la documentazione del programma; non è previsto che questa opzione venga utilizzata con altre, ma poiché il valore è zero, tale utilizzo non può essere rilevato.

Questo è un campo di immissione. Il valore iniziale del campo di *Options* è MQCBDO_NONE.

CallbackArea (MQPTR) per MQCBD

Struttura descrittore callback - Campo CallbackArea

Questo è un campo disponibile per la funzione di callback da utilizzare.

Il gestore code non prende decisioni in base al contenuto di questo campo e viene trasmesso non modificato dal campo CallbackArea nella struttura MQCBC, che è un parametro nella dichiarazione della funzione di callback.

Il valore viene utilizzato solo su un *Operation* con un valore MQOP_REGISTER, senza callback attualmente definito, non sostituisce una definizione precedente.

Questo è un campo di input e output per la funzione di callback. Il valore iniziale di questo campo è un puntatore null o byte null.

CallbackFunction (MQPTR) per MQCBD

Struttura descrittore callback - Campo CallbackFunction

La funzione callback viene richiamata come una chiamata di funzione.

Utilizzare questo campo per specificare un puntatore alla funzione callback.

È necessario specificare *CallbackFunction* o *CallbackName*. Se si specificano entrambi, viene restituito il codice di errore MQRC_CALLBACK_ROUTINE_ERROR.

Se non è impostato né *CallbackName* né *CallbackFunction* , la chiamata ha esito negativo con il codice di errore MQRC_CALLBACK_ROUTINE_ERROR.

Questa opzione non è supportata nel seguente ambiente: linguaggi di programmazione e programmi di compilazione che non supportano riferimenti di puntatore di funzione. In tali situazioni, la chiamata ha esito negativo con il codice motivo MQRC_CALLBACK_ROUTINE_ERROR.

z/OS Su z/OS, la funzione deve essere richiamata con le convenzioni di collegamento del SO. Ad esempio, nel linguaggio di programmazione C, specificare:

```
#pragma linkage(MQCB_FUNCTION,OS)
```

Questo è un campo di immissione. Il valore iniziale di questo campo è un puntatore null o byte null.

Nota: Quando si utilizza CICS con IBM WebSphere MQ 7.0.1, l'utilizzo asincrono è supportato se:

- Apar PK66866 viene applicato a CICS TS 3.2
- Apar PK89844 viene applicato a CICS TS 4.1

CallbackName (MQCHAR128) per MQCBD

Struttura descrittore callback - Campo CallbackName

La funzione callback viene richiamata come un programma collegato dinamicamente.

È necessario specificare *CallbackFunction* o *CallbackName*. Se si specificano entrambi, viene restituito il codice di errore MQRC_CALLBACK_ROUTINE_ERROR.

Se non è impostato né *CallbackName* né *CallbackFunction* , la chiamata ha esito negativo con il codice di errore MQRC_CALLBACK_ROUTINE_ERROR.

Il modulo viene caricato quando viene registrata la prima routine di callback da utilizzare e scaricato quando viene annullata la registrazione dell'ultima routine di callback da utilizzare.

Tranne dove indicato nel testo seguente, il nome è giustificato a sinistra all'interno del campo, senza spazi incorporati; il nome stesso viene riempito con spazi vuoti fino alla lunghezza del campo. Nelle descrizioni che seguono, le parentesi quadre ([]) indicano informazioni facoltative:

IBM i

Il nome callback può essere uno dei formati seguenti:

- Programma "/" libreria
- Libreria "/" ServiceProgram "("FunctionName")"

Ad esempio, `MyLibrary/MyProgram(MyFunction)`.

Il nome della libreria può essere *LIBL. I nomi della libreria e del programma sono limitati ad un massimo di 10 caratteri.

AIX and Linux
Il nome callback è il nome di un modulo o libreria caricabile dinamicamente, con il suffisso del nome di una funzione che risiede in tale libreria. Il nome della funzione deve essere racchiuso tra parentesi. Il nome della libreria può essere facoltativamente preceduto da un percorso di directory:

```
[path]library(function)
```

Se il percorso non viene specificato, viene utilizzato il percorso di ricerca del sistema.

Il nome è limitato a un massimo di 128 caratteri.

Windows
Il nome di callback è il nome di una libreria di collegamento dinamico, con suffisso il nome di una funzione che risiede in tale libreria. Il nome della funzione deve essere racchiuso tra parentesi. Il nome della libreria può essere facoltativamente preceduto da un percorso di directory e unità:

```
[d:][path]library(function)
```

Se l'unità e il percorso non sono specificati, viene utilizzato il percorso di ricerca del sistema.

Il nome è limitato a un massimo di 128 caratteri.

z/OS
Il nome di callback è il nome di un modulo di caricamento valido per la specifica sul parametro EP della macro LINK o LOAD.

Il nome è limitato a un massimo di 8 caratteri.

z/OS CICS
Il nome di callback è il nome di un modulo di caricamento valido per la specificazione nel parametro PROGRAM della macro del comando EXEC CICS LINK.

Il nome è limitato a un massimo di 8 caratteri.

Il programma può essere definito come remoto utilizzando l'opzione REMOTESYTEM della definizione PROGRAM installata o tramite il programma di instradamento dinamico.

La regione CICS remota deve essere connessa a IBM MQ se il programma deve utilizzare le chiamate API IBM MQ . Si noti, tuttavia, che il campo Hobj nella struttura MQCBC non è valido in un sistema remoto.

Se si verifica un errore durante il tentativo di caricamento di *CallbackName*, all'applicazione viene restituito uno dei seguenti codici di errore:

- MQRC_MODULE_NOT_FOUND
- ID_NON_VALIDO MQRC_MODULE
- MQRC_MODULE_ENTRY_NOT_FOUND

Viene anche scritto un messaggio nel log degli errori contenente il nome del modulo per cui è stato tentato il caricamento e il codice di errore dal sistema operativo.

Questo è un campo di immissione. Il valore iniziale di questo campo è una stringa nulla o spazi vuoti.

MaxMsgLength (MQLONG) per MQCBD
Questa è la lunghezza in byte del messaggio più lungo che può essere letto dall'handle e fornito alla routine di callback. Struttura del descrittore di callback - MaxMsgCampo Lunghezza

Se un messaggio ha una lunghezza maggiore, la routine di callback riceve *MaxMsgLength* byte del messaggio e il codice di errore:

- MQRC_TRUNCATED_MSG_FAILED o

- MQRC_TRUNCATED_MSG_ACCEPTED se è stato specificato MQGMO_ACCEPT_TRUNCATED_MSG.

La lunghezza reale del messaggio viene fornita nel campo DataLength della struttura MQCBC.

Viene definito il seguente valore speciale:

MQCBD_FULL_MSG_LENGTH
La lunghezza del buffer viene regolata dal sistema per restituire i messaggi senza troncamento.

Se la memoria disponibile non è sufficiente per assegnare un buffer per ricevere il messaggio, il sistema richiama la funzione di callback con un codice motivo MQRC_STORAGE_NOT_AVAILABLE.

Se, ad esempio, si richiede la conversione dei dati e la memoria disponibile non è sufficiente per convertire i dati del messaggio, il messaggio non convertito viene passato alla funzione di callback.

Questo è un campo di immissione. Il valore iniziale del campo *MaxMsgLength* è MQCBD_FULL_MSG_LENGTH.

MQCHARV - Stringa a lunghezza variabile

Utilizzare la struttura MQCHARV per descrivere una stringa di lunghezza variabile.

Disponibilità

La struttura MQCHARV è disponibile sulle piattaforme seguenti:

- AIX
- IBM i
- Linux
- Windows

e per IBM MQ MQI clients collegati a questi sistemi.

Serie di caratteri e codifica

I dati in MQCHARV devono essere nella codifica del gestore code locale fornito da MQENC_NATIVE e la serie di caratteri del campo VSCCSID nella struttura. Se l'applicazione è in esecuzione come client MQ , la struttura deve essere nella codifica del client. Alcune serie di caratteri hanno una rappresentazione che dipende dalla codifica. Se VSCCSID è una di queste serie di caratteri, la codifica utilizzata è la stessa di quella degli altri campi in MQCHARV. La serie di caratteri identificata da VSCCSID può essere una serie di caratteri a doppio byte (DBCS).

Utilizzo

La struttura MQCHARV indirizza i dati che potrebbero essere discontigui con la struttura che li contiene. Per indirizzare questi dati, è possibile utilizzare i campi dichiarati con il tipo di dati puntatore. Tenere presente che COBOL non supporta il tipo di dati puntatore in tutti gli ambienti. Per questo motivo, i dati possono essere indirizzati anche utilizzando campi che contengano l'offset dei dati dall'inizio della struttura contenente MQCHARV.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 475. Campi in MQCHARV		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
VSPtr (puntatore alla stringa di lunghezza variabile)	Nessuna	Puntatore null o byte null.
VSOffset (offset in byte della stringa di lunghezza variabile dall'avvio della struttura che contiene questa struttura MQCHARV)	Nessuna	0
VSBufSize (dimensione in byte del buffer indirizzato dal campo VSPtr o VSOffset)	MQVS_UT_VSLENGTH	0
VSLength (lunghezza in byte della stringa di lunghezza variabile indirizzata dal campo VSPtr o VSOffset)	Nessuna	0
VSCCSID (identificativo della serie di caratteri della stringa di lunghezza variabile indirizzata dal campo VSPtr o VSOffset)	APPL MQCCSI	-3
Nota: Nel linguaggio di programmazione C, la variabile macro MQCHARV_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura: `MQCHARV MyVarStr = {MQCHARV_DEFAULT};`		

Dichiarazioni di lingua

Dichiarazione C per MQCHARV

```
typedef struct tagMQCHARV MQCHARV;
struct tagMQCHARV {
  MQPTR     VSPtr;              /* Address of variable length string */
  MQLONG    VSOffset;           /* Offset of variable length string */
  MQLONG    VSBufSize;          /* Size of buffer */
  MQLONG    VSLength;           /* Length of variable length string */
  MQLONG    VSCCSID;            /* CCSID of variable length string */
};
```

Dichiarazione COBOL per MQCHARV

```
** MQCHARV structure
 10  MQCHARV.
** Address of variable length string
 15  MQCHARV-VSPTR        POINTER.
** Offset of variable length string
 15  MQCHARV-VSOFFSET     PIC S9(9) BINARY.
** Size of buffer
 15  MQCHARV-VSBUFSIZE    PIC S9(9) BINARY.
** Length of variable length string
 15  MQCHARV-VSLENGTH     PIC S9(9) BINARY.
** CCSID of variable length string
 15  MQCHARV-VSCCSID      PIC S9(9) BINARY.
```

Nota: Se si desidera trasferire un'applicazione COBOL tra ambienti, è necessario scoprire se il tipo di dati del puntatore è disponibile in tutti gli ambienti previsti. In caso contrario, l'applicazione deve indirizzare i dati utilizzando i campi di scostamento invece dei campi puntatore. In ambienti in cui i puntatori non sono supportati, è possibile dichiarare i campi del puntatore come stringhe di byte della lunghezza appropriata, con il valore iniziale che è la stringa di byte all - null. Non modificare questo valore iniziale se si utilizzano

i campi di scostamento. Un modo per farlo senza modificare le copie fornite è quello di utilizzare quanto segue:

```
COPY CMQCHRVV REPLACING POINTER BY ==BINARY PIC S9(9)==.
```

dove CMQCHRVV può essere scambiato per il copy book da utilizzare.

Dichiarazione PL/I per MQCHARV

```
dcl
 1 MQCHARV based,
  3 VSPtr       pointer,       /* Address of variable length string */
  3 VSOffset    fixed bin(31), /* Offset of variable length string */
  3 VSBufSize   fixed bin(31), /* Size of buffer */
  3 VSLength    fixed bin(31), /* Length of variable length string */
  3 VSCCSID     fixed bin(31); /* CCSID of variable length string */
```

Dichiarazione High Level Assembler per MQCHARV

```
MQCHARV             DSECT
MQCHARV_VSPTR       DS   F     Address of variable length string
MQCHARV_VSOFFSET    DS   F     Offset of variable length string
MQCHARV_VSBUFSIZE   DS   F     Size of buffer
MQCHARV_VSLENGTH    DS   F     Length of variable length string
MQCHARV_VSCCSID     DS   F     CCSID of variable length string
*
MQCHARV_LENGTH      EQU  *-MQCHARV
                    ORG  MQCHARV
MQCHARV_AREA        DS   CL(MQCHARV_LENGTH)
```

VSPtr (MQPTR) per MQCHARV

Questo è un puntatore alla stringa a lunghezza variabile.

È possibile utilizzare il campo VSPtr o VSOffset per specificare la stringa a lunghezza variabile, ma non entrambi.

Il valore iniziale di questo campo è un puntatore null o byte null.

VSOffset (MQLONG) per MQCHARV

L'offset può essere positivo o negativo. È possibile utilizzare il campo VSPtr o VSOffset per specificare la stringa a lunghezza variabile, ma non entrambi. L'offset in byte della stringa di lunghezza variabile dall'inizio di MQCHARV o dalla struttura che lo contiene.

Quando la struttura MQCHARV è integrata in un'altra struttura, questo valore è l'offset in byte della stringa di lunghezza variabile dall'inizio della struttura che contiene questa struttura MQCHARV. Quando la struttura MQCHARV non è incorporata in un'altra struttura, ad esempio, se viene specificata come parametro in una chiamata di funzione, l'offset è relativo all'avvio della struttura MQCHARV.

Il valore iniziale di questo campo è 0.

VSBufSize (MQLONG) per MQCHARV

Questa è la dimensione in byte del buffer indirizzato dal campo VSPtr o VSOffset.

Quando la struttura MQCHARV viene utilizzata come un campo di output in una chiamata di funzione, questo campo deve essere inizializzato con la lunghezza del buffer fornito. Se il valore di VSLength è maggiore di VSBufSize , solo VSBufSize byte di dati vengono restituiti al chiamante nel buffer.

Questo valore deve essere un valore maggiore o uguale a zero oppure il seguente valore speciale riconosciuto:

MQVS_UT_VSLENGTH
Quando viene specificato, la lunghezza del buffer viene presa dal campo VSLength nella struttura MQCHARV. Non utilizzare questo valore quando si utilizza la struttura come campo di output e viene fornito un buffer.

Questo è il valore iniziale di questo campo.

VSLength (MQLONG) per MQCHARV

La lunghezza in byte della stringa di lunghezza variabile indirizzata dal campo VSPtr o VSOffset.

Il valore iniziale di questo campo è 0. Il valore deve essere maggiore o uguale a zero o il seguente valore speciale riconosciuto:

MQVS_NULL_TERMINATED

Se MQVS_NULL_TERMINATED non è specificato, i byte VSLength sono inclusi come parte della stringa. Se sono presenti caratteri null, non delimitano la stringa.

Se viene specificato MQVS_NULL_TERMINATED, la stringa è delimitata dal primo valore null rilevato nella stringa. Il valore null non viene incluso come parte di tale stringa.

Nota: Il carattere null utilizzato per terminare una stringa se viene specificato MQVS_NULL_TERMINATED è un valore null dal codeset specificato da VSCCSID.

Ad esempio, in UTF-16 (CCSID 1200, 13488 e 17584), questa è la codifica Unicode a due byte in cui un valore null è rappresentato da un numero di 16 bit di tutti zeri. In UTF-16 è comune trovare singoli byte impostati su tutti zero che fanno parte di caratteri (ad esempio, caratteri ASCII a 7 bit), ma le stringhe termineranno con valore null solo quando due byte 'zero' si trovano su un limite di byte pari. È possibile ottenere due byte 'zero' su un limite dispari quando fanno parte di caratteri validi. Ad esempio x '01' x '00 x' 00 'x' 30 ' rappresenta due caratteri Unicode validi e non termina la stringa con un valore null.

VSCCSID (MQLONG) per MQCHARV

Questo è l'identificativo della serie di caratteri della stringa di lunghezza variabile indirizzata dal campo **VSPtr** o **VSOffset** .

Il valore iniziale di questo campo è *MQCCSI_APPL*, che è definito da MQ per indicare che deve essere modificato nell'identificativo della serie di caratteri true del processo corrente. Di conseguenza, il valore della costante *MQCCSI_APPL* non è mai associato a una stringa di lunghezza variabile.

Il valore iniziale di questo campo può essere modificato definendo un altro valore per la costante *MQCCSI_APPL* per l'unità di compilazione. Il modo in cui si esegue questa operazione dipende dal linguaggio di programmazione dell'applicazione.

z/OS Sui sistemi z/OS , l'applicazione predefinita CCSID utilizzata da *MQCCSI_APPL* è definita come segue:

- Per le applicazioni LE batch che utilizzano l'interfaccia DLL, il valore predefinito è CODESET associato alla locale corrente al momento dell'emissione di **MQCONN** (il valore predefinito è 1047).
- Per le applicazioni LE batch associate a uno degli stub MQ batch, il valore predefinito è CODESET associato alla locale corrente al momento della prima chiamata MQI emessa dopo **MQCONN** (il valore predefinito è 1047).
- Per le applicazioni non LE batch in esecuzione su un thread z/OS UNIX System Services , il valore predefinito è quello di THLICCSID al momento della prima chiamata MQI emessa dopo **MQCONN** (il valore predefinito è 1047).
- Per altre applicazioni batch, il valore predefinito è CCSID del gestore code.

Ridefinizione di MQCCSI_APPL

I seguenti esempi mostrano come è possibile sovrascrivere il valore di MQCCSI_APPL in vari linguaggi di programmazione. È possibile modificare il valore di MQCCSI_APPL, eliminando la necessità di impostare separatamente il VSCCSID per ciascuna stringa di lunghezza variabile. In questi esempi il CCSID è impostato su 1208; modificarlo nel valore richiesto. Questo diventa il valore predefinito, che è possibile sovrascrivere impostando VSCCSID in qualsiasi istanza specifica di MQCHARV.

Utilizzo C

```
#define MQCCSI_APPL 1208
#include <cmqc.h>
```

Utilizzo di COBOL

```
COPY CMQXYZV REPLACING -3 BY 1208.
```

utilizzo PL/I

```
%MQCCSI_APPL = '1208';
%include syslib(cmqp);
```

utilizzo di High Level Assembler

```
MQCCSI_APPL EQU 1208
CMQA LIST=NO
```

Intestazione MQCIH - CICS bridge

La struttura MQCIH descrive le informazioni di intestazione per un messaggio inviato a CICS attraverso CICS bridge.

Per qualsiasi piattaforma supportata da IBM MQ è possibile creare e trasmettere un messaggio che include la struttura MQCIH, ma solo un gestore code IBM MQ for z/OS può utilizzare CICS bridge. Pertanto, perché il messaggio arrivi a CICS da un gestore code nonz/OS , la rete del gestore code deve includere almeno un gestore code z/OS attraverso il quale il messaggio può essere instradato.

Tutte le versioni CICS supportate da IBM MQ 9.0.0e successive utilizzano la versione fornita da CICS del bridge. Per ulteriori informazioni sulla configurazione dell'adattatore IBM MQ CICS e dei componenti di IBM MQ CICS bridge , consultare la sezione Configurazione delle connessioni a MQ della documentazione di CICS .

Disponibilità

La struttura MQCIH è disponibile sulle piattaforme seguenti:

- 

AIX
- 

Linux
- 

Windows
- 

z/OS

e per IBM MQ MQI clients collegati a questi sistemi.

Nome formato

MQFMT_CICS

Versione

La versione corrente di MQCIH è MQCIH_VERSION_2. I campi che esistono solo nella versione più recente della struttura vengono identificati come tali nelle descrizioni che seguono.

I file di intestazione, COPY e INCLUDE forniti per i linguaggi di programmazione supportati contengono la versione più recente di MQCIH, con il valore iniziale del campo *Version* impostato su MQCIH_VERSION_2.

Serie di caratteri e codifica

Le condizioni speciali si applicano alla serie di caratteri e alla codifica utilizzati per la struttura MQCIH e i dati del messaggio dell'applicazione:

- Le applicazioni che si connettono al gestore code che possiede la coda CICS bridge devono fornire una struttura MQCIH che si trova nella serie di caratteri e nella codifica del gestore code. Ciò si verifica perché la conversione dei dati della struttura MQCIH non viene eseguita in questo caso.
- Le applicazioni che si connettono ad altri gestori code possono fornire una struttura MQCIH che si trova in una qualsiasi delle serie di caratteri e delle codifiche supportate; l'agent del canale dei messaggi ricevente connesso al gestore code che possiede la coda CICS bridge converte la struttura MQCIH.
- I dati del messaggio dell'applicazione che seguono la struttura MQCIH devono essere nella stessa serie di caratteri e codifica della struttura MQCIH. Non è possibile utilizzare i campi *CodedCharSetId* e *Encoding* nella struttura MQCIH per specificare la serie di caratteri e la codifica dei dati del messaggio dell'applicazione.

  È necessario fornire un'uscita di conversione dati per convertire i dati del messaggio dell'applicazione se i dati non sono uno dei formati integrati supportati dal gestore code.

Utilizzo

Se l'applicazione richiede valori uguali ai valori iniziali mostrati in Tabella 477 a pagina 305e il bridge è in esecuzione con AUTH=LOCAL o AUTH=IDENTIFY, è possibile omettere la struttura MQCIH dal messaggio. In tutti gli altri casi, la struttura deve essere presente.

Il bridge accetta una struttura MQCIH version-1 o version-2 , ma per le transazioni 3270, è necessario utilizzare una struttura version-2 .

L'applicazione deve garantire che i campi documentati come campi di richiesta abbiano valori appropriati nel messaggio inviato al bridge; questi campi sono immessi nel bridge.

I campi documentati come campi di risposta vengono impostati da CICS bridge nel messaggio di replica che il bridge invia all'applicazione. Le informazioni sugli errori vengono restituite nei campi *ReturnCode*, *Function*, *CompCode*, *Reason* e *AbendCode* , ma non tutte sono impostate in tutti i casi. La seguente tabella mostra i campi impostati per i diversi valori di *ReturnCode*.

Tabella 476. Contenuto dei campi di informazioni sull'errore nella struttura MQCIH per MQCIH

ReturnCode	***Function***	***CompCode***	***Reason***	***AbendCode***
OK MQCRC	-	-	-	-
ERRORE MQCRC_BRIDGE_	-	-	MQFB_CICS_ *	-
MQCRC_MQ_API_ERROR MQCRC_BRIDGE_TIMEOUT	Nome chiamata MQ	MQ *CompCode*	MQ *Reason*	-
MQCRC_CICS_EXEC_ERROR MQCRC_SECURITY_ERROR MQCRC_PROGRAM_NOT_AVAILABLE MQCRC_TRANSID_NOT_AVAILABLE	EIBFN CICS	CICS EIBRESP	CICS EIBRESP2	-
MQCRC_BRIDGE_ABEND MQCRC_APPLICATION_ABEND	-	-	-	CICS CODICE

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 477. Campi in MQCIH per MQCIH

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQCIH	'CIH¬'
Versione (numero versione struttura)	MQCIH_VERSION_2	2
StrucLength (lunghezza della struttura MQCIH)	MQCIH_LENGTH_2	180
Codifica (riservato)	Nessuna	0
CodedCharSetId (riservato)	Nessuna	0
Formato (nome formatoMQ dei dati che seguono MQCIH)	MQFMT_NONE	Spazi
Indicatori (indicatori)	MQCIH_NONE	0
ReturnCode (codice di ritorno dal bridge)	OK MQCRC	0
CompCode (codice di completamentoMQ o CICS EIBRESP)	MQCC_OK	0
Motivo (codice motivo o feedback diMQ o CICS EIBRESP2)	MQRC_NONE	0
UOWControl (controllo unità di lavoro)	SOLO MQCUOWC_	273
GetWaitGetWait (intervallo di attesa per la chiamata MQGET emessa dall'attività bridge)	MQCGWI_PREDEFINITO	-2
LinkType (tipo di collegamento)	PROGRAMMA_MQCL	1
OutputDataLength (lunghezza dati COMMAREA di emissione)	MQCODL_AS_INPUT	-1
FacilityKeepTime (ora di rilascio della funzione bridge)	Nessuna	0
ADSDescriptor (invio / ricezione descrittore ADS)	MQCADSD_NONE	0
ConversationalTask (se l'attività può essere conversazionale)	NO MQCCT	0
TaskEndStato (stato alla fine dell'attività)	NOSYNC MQCTES	0
Funzione (token funzione bridge)	MQCFAC_NONE	Valori null
Funzione (nome della chiamataMQ o funzione CICS EIBFN)	MQCFUN_NONE	Spazi
AbendCode (codice abend)	Nessuna	Spazi
Programma di autenticazione (password o passticket)	Nessuna	Spazi
Reserved1 (riservato)	Nessuna	Spazi
ReplyToFormat (nome formatoMQ del messaggio di risposta)	MQFMT_NONE	Spazi

Tabella 477. Campi in MQCIH per MQCIH (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
RemoteSysId (ID sistema CICS remoto da utilizzare)	Nessuna	Spazi
IDRemoteTrans (CICS RTRANSID da utilizzare)	Nessuna	Spazi
TransactionId (transazione da collegare)	Nessuna	Spazi
FacilityLike (attributi emulati di terminale)	Nessuna	Spazi
AttentionId (tasto AID)	Nessuna	Spazi
StartCode (codice di avvio transazione)	MQCSC_NONE	Spazi
CancelCode (codice transazione di fine anomala)	Nessuna	Spazi
NextTransactionId (transazione successiva da collegare)	Nessuna	Spazi
Reserved2 (riservato)	Nessuna	Spazi
Reserved3 (riservato)	Nessuna	Spazi
Nota: I campi rimanenti non sono presenti se *Version* è inferiore a MQCIH_VERSION_2.		
CursorPosition (posizione cursore)	Nessuna	0
ErrorOffset (offset dell'errore nel messaggio)	Nessuna	0
InputItem (elemento di input)	Nessuna	0
Reserved4 (riservato)	Nessuna	0

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macroMQCIH_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQCIH MyCIH = {MQCIH_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQCIH

```
typedef struct tagMQCIH MQCIH;
struct tagMQCIH {
  MQCHAR4  StrucId;             /* Structure identifier */
  MQLONG   Version;             /* Structure version number */
  MQLONG   StrucLength;         /* Length of MQCIH structure */
  MQLONG   Encoding;            /* Reserved */
  MQLONG   CodedCharSetId;      /* Reserved */
  MQCHAR8  Format;              /* MQ format name of data that follows
                                   MQCIH */
  MQLONG   Flags;               /* Flags */
  MQLONG   ReturnCode;          /* Return code from bridge */
  MQLONG   CompCode;            /* MQ completion code or CICS EIBRESP */
  MQLONG   Reason;              /* MQ reason or feedback code, or CICS
                                   EIBRESP2 */
  MQLONG   UOWControl;          /* Unit-of-work control */
```

```
  MQLONG   GetWaitInterval;     /* Wait interval for MQGET call issued
                                   by bridge task */
  MQLONG   LinkType;            /* Link type */
  MQLONG   OutputDataLength;    /* Output COMMAREA data length */
  MQLONG   FacilityKeepTime;    /* Bridge facility release time */
  MQLONG   ADSDescriptor;       /* Send/receive ADS descriptor */
  MQLONG   ConversationalTask;  /* Whether task can be conversational */
  MQLONG   TaskEndStatus;       /* Status at end of task */
  MQBYTE8  Facility;            /* Bridge facility token */
  MQCHAR4  Function;            /* MQ call name or CICS EIBFN
                                   function */
  MQCHAR4  AbendCode;           /* Abend code */
  MQCHAR8  Authenticator;       /* Password or passticket */
  MQCHAR8  Reserved1;           /* Reserved */
  MQCHAR8  ReplyToFormat;       /* MQ format name of reply message */
  MQCHAR4  RemoteSysId;         /* Reserved */
  MQCHAR4  RemoteTransId;       /* Reserved */
  MQCHAR4  TransactionId;       /* Transaction to attach */
  MQCHAR4  FacilityLike;        /* Terminal emulated attributes */
  MQCHAR4  AttentionId;         /* AID key */
  MQCHAR4  StartCode;           /* Transaction start code */
  MQCHAR4  CancelCode;          /* Abend transaction code */
  MQCHAR4  NextTransactionId;   /* Next transaction to attach */
  MQCHAR8  Reserved2;           /* Reserved */
  MQCHAR8  Reserved3;           /* Reserved */
  MQLONG   CursorPosition;      /* Cursor position */
  MQLONG   ErrorOffset;         /* Offset of error in message */
  MQLONG   InputItem;           /* Reserved */
  MQLONG   Reserved4;           /* Reserved */
};
```

Dichiarazione COBOL per MQCIH

```
**   MQCIH structure
  10 MQCIH.
**    Structure identifier
   15 MQCIH-STRUCID            PIC X(4).
**    Structure version number
   15 MQCIH-VERSION            PIC S9(9) BINARY.
**    Length of MQCIH structure
   15 MQCIH-STRUCLENGTH        PIC S9(9) BINARY.
**    Reserved
   15 MQCIH-ENCODING           PIC S9(9) BINARY.
**    Reserved
   15 MQCIH-CODEDCHARSETID     PIC S9(9) BINARY.
**    MQ format name of data that follows MQCIH
   15 MQCIH-FORMAT             PIC X(8).
**    Flags
   15 MQCIH-FLAGS              PIC S9(9) BINARY.
**    Return code from bridge
   15 MQCIH-RETURNCODE         PIC S9(9) BINARY.
**    MQ completion code or CICS EIBRESP
   15 MQCIH-COMPCODE           PIC S9(9) BINARY.
**    MQ reason or feedback code, or CICS EIBRESP2
   15 MQCIH-REASON             PIC S9(9) BINARY.
**    Unit-of-work control
   15 MQCIH-UOWCONTROL         PIC S9(9) BINARY.
**    Wait interval for MQGET call issued by bridge task
   15 MQCIH-GETWAITINTERVAL    PIC S9(9) BINARY.
**    Link type
   15 MQCIH-LINKTYPE           PIC S9(9) BINARY.
**    Output COMMAREA data length
   15 MQCIH-OUTPUTDATALENGTH   PIC S9(9) BINARY.
**    Bridge facility release time
   15 MQCIH-FACILITYKEEPTIME   PIC S9(9) BINARY.
**    Send/receive ADS descriptor
   15 MQCIH-ADSDESCRIPTOR      PIC S9(9) BINARY.
**    Whether task can be conversational
   15 MQCIH-CONVERSATIONALTASK PIC S9(9) BINARY.
**    Status at end of task
   15 MQCIH-TASKENDSTATUS      PIC S9(9) BINARY.
**    Bridge facility token
   15 MQCIH-FACILITY           PIC X(8).
**    MQ call name or CICS EIBFN function
   15 MQCIH-FUNCTION           PIC X(4).
**    Abend code
   15 MQCIH-ABENDCODE          PIC X(4).
**    Password or passticket
   15 MQCIH-AUTHENTICATOR      PIC X(8).
```

```
**    Reserved
   15 MQCIH-RESERVED1          PIC X(8).
**    MQ format name of reply message
   15 MQCIH-REPLYTOFORMAT      PIC X(8).
**    Reserved
   15 MQCIH-REMOTESYSID        PIC X(4).
**    Reserved
   15 MQCIH-REMOTETRANSID      PIC X(4).
**    Transaction to attach
   15 MQCIH-TRANSACTIONID      PIC X(4).
**    Terminal emulated attributes
   15 MQCIH-FACILITYLIKE       PIC X(4).
**    AID key
   15 MQCIH-ATTENTIONID        PIC X(4).
**    Transaction start code
   15 MQCIH-STARTCODE          PIC X(4).
**    Abend transaction code
   15 MQCIH-CANCELCODE         PIC X(4).
**    Next transaction to attach
   15 MQCIH-NEXTTRANSACTIONID  PIC X(4).
**    Reserved
   15 MQCIH-RESERVED2          PIC X(8).
**    Reserved
   15 MQCIH-RESERVED3          PIC X(8).
**    Cursor position
   15 MQCIH-CURSORPOSITION     PIC S9(9) BINARY.
**    Offset of error in message
   15 MQCIH-ERROROFFSET        PIC S9(9) BINARY.
**    Reserved
   15 MQCIH-INPUTITEM          PIC S9(9) BINARY.
**    Reserved
   15 MQCIH-RESERVED4          PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQCIH

```
dcl
 1 MQCIH based,
  3 StrucId            char(4),       /* Structure identifier */
  3 Version            fixed bin(31), /* Structure version number */
  3 StrucLength        fixed bin(31), /* Length of MQCIH structure */
  3 Encoding           fixed bin(31), /* Reserved */
  3 CodedCharSetId     fixed bin(31), /* Reserved */
  3 Format             char(8),       /* MQ format name of data that
                                         follows MQCIH */
  3 Flags              fixed bin(31), /* Flags */
  3 ReturnCode         fixed bin(31), /* Return code from bridge */
  3 CompCode           fixed bin(31), /* MQ completion code or CICS
                                         EIBRESP */
  3 Reason             fixed bin(31), /* MQ reason or feedback code, or
                                         CICS EIBRESP2 */
  3 UOWControl         fixed bin(31), /* Unit-of-work control */
  3 GetWaitInterval    fixed bin(31), /* Wait interval for MQGET call
                                         issued by bridge task */
  3 LinkType           fixed bin(31), /* Link type */
  3 OutputDataLength   fixed bin(31), /* Output COMMAREA data length */
  3 FacilityKeepTime   fixed bin(31), /* Bridge facility release time */
  3 ADSDescriptor      fixed bin(31), /* Send/receive ADS descriptor */
  3 ConversationalTask fixed bin(31), /* Whether task can be
                                         conversational */
  3 TaskEndStatus      fixed bin(31), /* Status at end of task */
  3 Facility           char(8),       /* Bridge facility token */
  3 Function           char(4),       /* MQ call name or CICS EIBFN
                                         function */
  3 AbendCode          char(4),       /* Abend code */
  3 Authenticator      char(8),       /* Password or passticket */
  3 Reserved1          char(8),       /* Reserved */
  3 ReplyToFormat      char(8),       /* MQ format name of reply
                                         message */
  3 RemoteSysId        char(4),       /* Reserved */
  3 RemoteTransId      char(4),       /* Reserved */
  3 TransactionId      char(4),       /* Transaction to attach */
  3 FacilityLike       char(4),       /* Terminal emulated attributes */
  3 AttentionId        char(4),       /* AID key */
  3 StartCode          char(4),       /* Transaction start code */
  3 CancelCode         char(4),       /* Abend transaction code */
  3 NextTransactionId  char(4),       /* Next transaction to attach */
  3 Reserved2          char(8),       /* Reserved */
  3 Reserved3          char(8),       /* Reserved */
  3 CursorPosition     fixed bin(31), /* Cursor position */
```

```
  3 ErrorOffset        fixed bin(31), /* Offset of error in message */
  3 InputItem          fixed bin(31), /* Reserved */
  3 Reserved4          fixed bin(31); /* Reserved */
```

Dichiarazione High Level Assembler per MQCIH

```
MQCIH                     DSECT
MQCIH_STRUCID             DS   CL4  Structure identifier
MQCIH_VERSION             DS   F    Structure version number
MQCIH_STRUCLENGTH         DS   F    Length of MQCIH structure
MQCIH_ENCODING            DS   F    Reserved
MQCIH_CODEDCHARSETID      DS   F    Reserved
MQCIH_FORMAT              DS   CL8  MQ format name of data that follows
*                                   MQCIH
MQCIH_FLAGS               DS   F    Flags
MQCIH_RETURNCODE          DS   F    Return code from bridge
MQCIH_COMPCODE            DS   F    MQ completion code or CICS EIBRESP
MQCIH_REASON              DS   F    MQ reason or feedback code, or CICS
*                                   EIBRESP2
MQCIH_UOWCONTROL          DS   F    Unit-of-work control
MQCIH_GETWAITINTERVAL     DS   F    Wait interval for MQGET call issued
*                                   by bridge task
MQCIH_LINKTYPE            DS   F    Link type
MQCIH_OUTPUTDATALENGTH    DS   F    Output COMMAREA data length
MQCIH_FACILITYKEEPTIME    DS   F    Bridge facility release time
MQCIH_ADSDESCRIPTOR       DS   F    Send/receive ADS descriptor
MQCIH_CONVERSATIONALTASK  DS   F    Whether task can be conversational
MQCIH_TASKENDSTATUS       DS   F    Status at end of task
MQCIH_FACILITY            DS   XL8  Bridge facility token
MQCIH_FUNCTION            DS   CL4  MQ call name or CICS EIBFN function
MQCIH_ABENDCODE           DS   CL4  Abend code
MQCIH_AUTHENTICATOR       DS   CL8  Password or passticket
MQCIH_RESERVED1           DS   CL8  Reserved
MQCIH_REPLYTOFORMAT       DS   CL8  MQ format name of reply message
MQCIH_REMOTESYSID         DS   CL4  Reserved
MQCIH_REMOTETRANSID       DS   CL4  Reserved
MQCIH_TRANSACTIONID       DS   CL4  Transaction to attach
MQCIH_FACILITYLIKE        DS   CL4  Terminal emulated attributes
MQCIH_ATTENTIONID         DS   CL4  AID key
MQCIH_STARTCODE           DS   CL4  Transaction start code
MQCIH_CANCELCODE          DS   CL4  Abend transaction code
MQCIH_NEXTTRANSACTIONID   DS   CL4  Next transaction to attach
MQCIH_RESERVED2           DS   CL8  Reserved
MQCIH_RESERVED3           DS   CL8  Reserved
MQCIH_CURSORPOSITION      DS   F    Cursor position
MQCIH_ERROROFFSET         DS   F    Offset of error in message
MQCIH_INPUTITEM           DS   F    Reserved
MQCIH_RESERVED4           DS   F    Reserved
*
MQCIH_LENGTH              EQU  *-MQCIH
                          ORG  MQCIH
MQCIH_AREA                DS   CL(MQCIH_LENGTH)
```

Dichiarazione Visual Basic per MQCIH

```
Type MQCIH
  StrucId            As String*4 'Structure identifier'
  Version            As Long     'Structure version number'
  StrucLength        As Long     'Length of MQCIH structure'
  Encoding           As Long     'Reserved'
  CodedCharSetId     As Long     'Reserved'
  Format             As String*8 'MQ format name of data that follows'
                                 'MQCIH'
  Flags              As Long     'Flags'
  ReturnCode         As Long     'Return code from bridge'
  CompCode           As Long     'MQ completion code or CICS EIBRESP'
  Reason             As Long     'MQ reason or feedback code, or CICS'
                                 'EIBRESP2'
  UOWControl         As Long     'Unit-of-work control'
  GetWaitInterval    As Long     'Wait interval for MQGET call issued'
                                 'by bridge task'
  LinkType           As Long     'Link type'
  OutputDataLength   As Long     'Output COMMAREA data length'
  FacilityKeepTime   As Long     'Bridge facility release time'
  ADSDescriptor      As Long     'Send/receive ADS descriptor'
  ConversationalTask As Long     'Whether task can be conversational'
  TaskEndStatus      As Long     'Status at end of task'
```

```
   Facility           As MQBYTE8  'Bridge facility token'
   Function           As String*4 'MQ call name or CICS EIBFN function'
   AbendCode          As String*4 'Abend code'
   Authenticator      As String*8 'Password or passticket'
   Reserved1          As String*8 'Reserved'
   ReplyToFormat      As String*8 'MQ format name of reply message'
   RemoteSysId        As String*4 'Reserved'
   RemoteTransId      As String*4 'Reserved'
   TransactionId      As String*4 'Transaction to attach'
   FacilityLike       As String*4 'Terminal emulated attributes'
   AttentionId        As String*4 'AID key'
   StartCode          As String*4 'Transaction start code'
   CancelCode         As String*4 'Abend transaction code'
   NextTransactionId  As String*4 'Next transaction to attach'
   Reserved2          As String*8 'Reserved'
   Reserved3          As String*8 'Reserved'
   CursorPosition     As Long     'Cursor position'
   ErrorOffset        As Long     'Offset of error in message'
   InputItem          As Long     'Reserved'
   Reserved4          As Long     'Reserved'
End Type
```

StrucId (MQCHAR4) per MQCIH

Questo è l'identificativo della struttura dell'intestazione delle informazioni CICS . È sempre un campo di immissione. Il valore è MQCIH_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQCIH
Identificativo per la struttura dell'intestazione delle informazioni CICS .

Per il linguaggio di programmazione C, viene definita anche la costante MQCIH_STRUC_ID_ARRAY. Ha lo stesso valore di MQCIH_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQCIH

Questo campo è un campo di richiesta. Il valore iniziale è MQCIH_VERSION_2.

Il valore deve essere uno dei seguenti.

MQCIH_VERSION_1
Version-1 CICS struttura dell'intestazione delle informazioni.

MQCIH_VERSION_2
Version-2 CICS struttura dell'intestazione delle informazioni.

I campi esistenti solo nella versione più recente della struttura vengono identificati come tali nelle descrizioni dei campi. La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQCIH_CURRENT_
Versione corrente della struttura dell'intestazione delle informazioni CICS .

StrucLength (MQLONG) per MQCIH

Questo è un campo di richiesta, con un valore iniziale di MQCIH_LENGTH_2.

Il valore deve essere uno dei seguenti.

MQCIH_LENGTH_1
Lunghezza della struttura dell'intestazione delle informazioni version-1 CICS .

MQCIH_LENGTH_2
Lunghezza della struttura dell'intestazione delle informazioni version-2 CICS .

La seguente costante specifica la lunghezza della versione corrente:

MQCIH_CURRENT_LENGTH
La lunghezza della versione corrente della struttura dell'intestazione delle informazioni CICS .

Codifica (MQLONG) per MQCIH

Questo campo è un campo riservato; il suo valore non è significativo. Il valore iniziale è 0.

La codifica per le strutture supportate che seguono una struttura MQCIH è uguale alla codifica della struttura MQCIH stessa e presa da qualsiasi intestazione IBM MQ precedente.

CodedCharSetId (MQLONG) per MQCIH

CodedCharSetId è un campo riservato; il suo valore non è significativo. Il valore iniziale di questo campo è 0.

L'ID della serie di caratteri per le strutture supportate che seguono una struttura MQCIH è uguale all'ID della serie di caratteri della struttura MQCIH stessa e viene preso da qualsiasi intestazione IBM MQ precedente.

Formato (MQCHAR8) per MQCIH

Questo campo mostra il nome formato IBM MQ dei dati che seguono la struttura MQCIH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono le stesse regole per la codifica del campo *Format* in MQMD.

Questo nome di formato viene utilizzato anche per il messaggio di risposta, se il campo *ReplyToFormat* ha il valore MQFMT_NONE.

- Per le richieste DPL, *Format* deve essere il nome formato della COMMAREA.
- Per le richieste 3270, *Format* deve essere CSQCBDCIe il bridge imposta il formato su CSQCBDC0 per i messaggi di risposta.

Le uscite di conversione dati per questi formati devono essere installate sul gestore code in cui devono essere eseguite.

Se il messaggio di richiesta genera un messaggio di risposta di errore, il messaggio di risposta di errore ha un nome formato MQFMT_STRING.

Questo campo è un campo di richiesta. La lunghezza di questo campo è fornita da MQ_FORMAT_LENGTH. Il valore iniziale di questo campo è MQFMT_NONE.

Indicatori (MQLONG) per MQCIH

Questo campo è un campo di richiesta. Il valore iniziale di questo campo è MQCIH_NONE.

Il valore deve essere:

MQCIH_NONE
Nessun indicatore.

SCADENZA_PASS_MQCIH
Il messaggio di risposta contiene:

- Le stesse opzioni del report di scadenza del messaggio di richiesta.
- Il tempo di scadenza rimanente dal messaggio di richiesta senza alcuna regolazione effettuata per il tempo di elaborazione del bridge.

Se si omette questo valore, la scadenza viene impostata su *unlimited*.

MQCIH_REPLY_WITHOUT_NULLS
La lunghezza del messaggio di risposta di una richiesta del programma CICS DPL viene regolata per escludere i valori null finali (X'00 ') alla fine della COMMAREA restituita dal programma DPL. Se questo valore non è impostato, i valori null potrebbero essere significativi e viene restituita la COMMAREA completa.

RETURN MQCIH_SYNC_ON
Il link CICS per le richieste DPL utilizza l'opzione SYNCONRETURN, facendo sì che CICS prenda un punto di sincronizzazione quando il programma viene completato se viene spedito in un'altra regione CICS . Il bridge non specifica a quale regione CICS spedire la richiesta; ciò è controllato dalla definizione del programma CICS o dalle funzioni di bilanciamento del carico di lavoro.

ReturnCode (MQLONG) per MQCIH
Il valore di questo campo è il codice di ritorno da CICS bridge che descrive il risultato dell'elaborazione effettuata dal bridge. Questo campo è un campo di risposta, con un valore iniziale di MQCRC_OK.

I campi *Function*, *CompCode*, *Reason* e *AbendCode* potrebbero contenere ulteriori informazioni (consultare Tabella 476 a pagina 304). Il valore è uno dei seguenti:

FINE ANOMALA APPLICAZIONE MQCR
(5, X'005 ') L'applicazione è terminata in modo anomalo.

MQCRC_BRIDGE_ABEND
(4, X'004 ') CICS bridge terminato in modo anomalo.

ERRORE MQCRC_BRIDGE_
(3, X'003 ') CICS bridge ha rilevato un errore.

MQCRC_BRIDGE_TIMEOUT
(8, X'008 ') Il secondo o il successivo messaggio nell'unità di lavoro corrente non è stato ricevuto entro il tempo specificato.

ERRORE MQCRC_CICS_EXEC
(1, X'001 ') L'istruzione EXEC CICS ha rilevato un errore.

ERRORE MQCR_MQ_API
(2, X'002 ') La chiamata MQ ha rilevato un errore.

OK MQCRC
(0, X'000 ') Nessun errore.

PROGRAMMA_MQCR_NOT_AVAILABLE
(7, X'007 ') Programma non disponibile.

ERRORE MQCR_SECURITY_ERROR
(6, X'006 ') Si è verificato un errore di sicurezza.

MQCRC_TRANSID_NOT_AVAILABLE
(9, X'009 ') Transazione non disponibile.

CompCode (MQLONG) per MQCIH
Questo campo è un campo di risposta. Il valore iniziale è MQCC_OK

Il valore restituito in questo campo dipende da *ReturnCode* ; consultare Tabella 476 a pagina 304.

Motivo (MQLONG) per MQCIH
Questo campo è un campo di risposta. Il valore iniziale è MQRC_NONE.

Il valore restituito in questo campo dipende da *ReturnCode* ; consultare Tabella 476 a pagina 304.

UOWControl (MQLONG) per MQCIH
Questo campo è un campo di richiesta che controlla l'elaborazione dell'unità di lavoro eseguita da CICS bridge. Il valore iniziale di questo campo è MQCUOWC_ONLY.

È possibile richiedere al bridge di eseguire una singola transazione o uno o più programmi all'interno di un'unità di lavoro. Il campo indica se CICS bridge avvia un'unità di lavoro, esegue la funzione richiesta all'interno dell'unità di lavoro corrente o termina l'unità di lavoro eseguendone il commit o il backout. Sono supportate varie combinazioni, per ottimizzare i flussi di trasmissione dati.

Il valore deve essere uno dei seguenti.

SOLO MQCUOWC_
Avviare l'unità di lavoro, eseguire la funzione, quindi eseguire il commit dell'unità di lavoro.

MQCUOWC_CONTINUA
Ulteriori dati per l'unità di lavoro corrente (solo 3270).

MQCUOWC_FIRST
Avviare l'unità di lavoro ed eseguire la funzione.

MQCUOWC_MEDIO
Esegui funzione all'interno dell'unità di lavoro corrente

LAST MQCUOWC
Eseguire la funzione, quindi eseguire il commit dell'unità di lavoro.

COMMIT MQCUOWC
Eseguire il commit dell'unità di lavoro (solo DPL).

BACKOUT MQCUOWC
Ripristinare l'unità di lavoro (solo DPL).

Intervallo GetWait(MQLONG) per MQCIH
Questo campo è un campo di richiesta. Il suo valore iniziale è MQCGWI_DEFAULT.

Questo campo è valido solo quando *UOWControl* ha il valore MQCUOWC_FIRST. Consente all'applicazione di invio di specificare il tempo approssimativo, in millisecondi, in cui le chiamate MQGET emesse dal bridge attenderanno il secondo e i successivi messaggi di richiesta per l'unità di lavoro avviata da questo messaggio. Questa funzione sovrascrive l'intervallo di attesa predefinito utilizzato dal bridge. È possibile utilizzare i seguenti valori speciali:

MQCGWI_DEFAULT
Intervallo di attesa predefinito.

Questo valore fa sì che CICS bridge attenda l'ora specificata quando è stato avviato il bridge.

MQWI_ILLIMITATO
Intervallo di attesa illimitato.

LinkType (MQLONG) per MQCIH
Questo campo è un campo di richiesta. Il valore iniziale è MQCLT_PROGRAM.

Questo valore indica il tipo di oggetto che il bridge tenta di collegare. Deve essere uno dei seguenti valori:

PROGRAMMA_MQCL
Programma DPL.

TRANSAZIONE MQCLT
Transazione 3270.

Lunghezza OutputData(MQLONG) per MQCIH
Questo campo è un campo di richiesta utilizzato solo per programmi DPL. Il valore iniziale è MQCODL_AS_INPUT.

Questo valore è la lunghezza dei dati dell'utente da restituire al client in un messaggio di risposta. Tale lunghezza comprende il nome di programma a 8-byte. La lunghezza della COMMAREA inoltrata al programma collegato è il massimo di questo campo e la lunghezza dei dati utente nel messaggio di richiesta, meno 8.

Nota: La lunghezza dei dati utente in un messaggio è la lunghezza del messaggio escludendo la struttura MQCIH.

Se la lunghezza dei dati dell'utente nel messaggio di richiesta è inferiore a *OutputDataLength*, viene utilizzata l'opzione DATALENGTH del comando LINK , consentendo a LINK di essere spedito in modo efficiente in un'altra regione CICS .

È possibile utilizzare il seguente valore speciale:

MQCODL_AS_INPUT
La lunghezza di emissione è uguale alla lunghezza di immissione.

Questo valore potrebbe essere necessario anche se non viene richiesta alcuna risposta, per garantire che la COMMAREA passata al programma collegato abbia una dimensione sufficiente.

FacilityKeepTime (MQLONG) per MQCIH

FacilityKeepIl tempo è il periodo di tempo, in secondi, in cui la funzione bridge viene conservata dopo la fine della transazione utente.

Per le transazioni pseudo - conversazionali, specificare un valore che corrisponda alla durata prevista di una pseudo - conversazione; specificare zero per l'ultima transazione di una pseudo - conversazione e per gli altri tipi di transazione specificare zero.

Questo campo è un campo di richiesta utilizzato solo per transazioni 3270. Il valore iniziale di questo campo è 0.

ADSDescriptor (MQLONG) per MQCIH

Questo campo è un indicatore che specifica se inviare descrittori ADS su richieste SEND e RECEIVE BMS.

Vengono definiti i seguenti valori:

MQCADSD_NONE
Non inviare o ricevere descrittori ADS.

MQCADSD_INVIA
Inviare i descrittori ADS.

MQCADSD_RECV
Ricevere i descrittori ADS.

MQCADSD_MSGFORMATO
Utilizzare il formato del messaggio per i descrittori ADS.

Questo invia o riceve i descrittori ADS utilizzando la forma estesa del descrittore ADS. Il modulo lungo ha campi allineati su limiti di 4 byte.

Impostare il campo *ADSDescriptor* come segue:

- Se non si utilizzano descrittori ADS, impostare il campo su MQCADSD_NONE.
- Se si utilizzano descrittori ADS con lo *stesso* CCSID in ogni ambiente, impostare il campo sulla somma di MQCADSD_SEND e MQCADSD_RECV.
- Se si utilizzano descrittori ADS con CCSID *differenti* in ciascun ambiente, impostare il campo sulla somma di MQCADSD_SEND, MQCADSD_RECV e MQCADSD_MSGFORMAT.

Questo è un campo di richiesta utilizzato solo per le transazioni 3270. Il valore iniziale di questo campo è MQCADSD_NONE.

ConversationalTask (MQLONG) per MQCIH

Questo campo è un indicatore che specifica se consentire all'attività di emettere richieste di ulteriori informazioni o di arrestare l'attività ed emettere un messaggio di interruzione.

Il valore deve essere una delle seguenti opzioni:

SÌ MQCC
L'attività è conversazionale.

MQCCT_NO
L'attività non è conversazionale.

Questo campo è un campo di richiesta utilizzato solo per transazioni 3270. Il valore iniziale di questo campo è MQCCT_NO.

Stato TaskEnd(MQLONG) per MQCIH

Questo campo è un campo di risposta, che mostra lo stato della transazione utente alla fine dell'attività. Il campo viene utilizzato solo per le transazioni 3270 e il valore iniziale è MQCTES_NOSYNC.

Viene restituito uno dei seguenti valori:

NOSYNC MQCTES
Non sincronizzato.

La transazione utente non è stata ancora completata e non è stata sincronizzata. Il campo *MsgType* in MQMD è MQMT_REQUEST in questo caso.

COMMIT MQCTES
Commit unità di lavoro.

La transazione utente non è stata ancora completata, ma ha sincronizzato la prima unità di lavoro. Il campo *MsgType* in MQMD è MQMT_DATAGRAM in questo caso.

BACKOUT MQCTES
Backout unità di lavoro.

La transazione utente non è stata ancora completata. L'unità di lavoro corrente viene ripristinata. Il campo *MsgType* in MQMD è MQMT_DATAGRAM in questo caso.

ENDTASK MQCTES
Termina attività.

La transazione utente è terminata (o è terminata in modo anomalo). Il campo *MsgType* in MQMD è MQMT_REPLY in questo caso.

Funzione (MQBYTE8) per MQCIH

Questo campo mostra il token della funzione bridge a 8 byte.

Un token della funzione bridge consente a più transazioni in una pseudo conversazione di utilizzare la stessa funzione bridge (terminale 3270 virtuale). Nel primo, o unico, messaggio in una pseudo - conversazione, impostare un valore di MQCFAC_NONE. Questo valore indica a CICS di allocare una nuova funzione bridge per questo messaggio. Un token della funzione bridge viene restituito nei messaggi di risposta quando viene specificato un *FacilityKeepTime* diverso da zero sul messaggio di input. I messaggi di input successivi all'interno di una pseudo - conversazione devono quindi utilizzare lo stesso token della funzione bridge.

Viene definito il seguente valore speciale:

MQCFAC_NONE
Nessun token di funzione specificato.

Per il linguaggio di programmazione C, viene definita anche la costante MQCFAC_NONE_ARRAY, che ha lo stesso valore di MQCFAC_NONE, ma è un array di caratteri invece di una stringa.

Questo campo è sia una richiesta che un campo di risposta utilizzato solo per le transazioni 3270. La lunghezza di questo campo è fornita da MQ_FACILITY_LENGTH. Il valore iniziale di questo campo è MQCFAC_NONE.

Funzione (MQCHAR4) per MQCIH

Questo campo è un campo di risposta. La lunghezza di questo campo è fornita da MQ_FUNCTION_LENGTH. Il valore iniziale di questo campo è MQCFUNC_NONE.

Il valore restituito in questo campo dipende da *ReturnCode* ; consultare Tabella 476 a pagina 304. I seguenti valori sono possibili quando *Function* contiene un nome chiamata IBM MQ :

MQCFUN_MQCONN
Chiamata MQCONN.

MQCFUN_MQGET
Chiamata MQGET.

MQCFUN_MQINQ
Chiamata MQINQ.

MQCFUN_MQOPEN
Chiamata MQOPEN.

MQCFUN_MQPUT
Chiamata MQPUT.

MQCFUNC_MQPUT1
Chiamata MQPUT1 .

MQCFUN_NONE
Nessuna chiamata.

In tutti i casi, per il linguaggio di programmazione C vengono definite anche le costanti MQCFUNC_ * _ARRAY; queste costanti hanno gli stessi valori delle costanti MQCFUNC_ * corrispondenti, ma sono array di caratteri invece di stringhe.

AbendCode (MQCHAR4) per MQCIH
AbendCode è un campo di risposta. La lunghezza di questo campo è fornita da MQ_ABEND_CODE_LENGTH. Il valore iniziale di questo campo è di 4 caratteri vuoti.

Il valore restituito in questo campo è significativo solo se il valore del campo *ReturnCode* è MQCRC_APPLICATION_ABEND o MQCRC_BRIDGE_ABEND. In caso contrario, *AbendCode* contiene il valore ABCODE di CICS .

Programma di autentificazione (MQCHAR8) per MQCIH
Il valore di questo campo è la password o il passticket.

Se l'autenticazione dell'identificativo utente è attiva per CICS bridge, *Authenticator* viene utilizzato con l'identificativo utente nel contesto di identità MQMD per autenticare il mittente del messaggio.

Questo è un campo di richiesta. La lunghezza di questo campo è fornita da MQ_AUTHENTICATOR_LENGTH. Il valore iniziale di questo campo è di 8 spazi.

Reserved1 (MQCHAR8) per MQCIH
Questo campo è riservato. Il valore deve essere di 8 spazi.

Formato ReplyTo(MQCHAR8) per MQCIH
Il valore di questo campo è il nome formato IBM MQ del messaggio di risposta inviato in risposta al messaggio corrente.

Le regole per la codifica di questo campo sono uguali a quelle per la codifica del campo *Format* in MQMD.

Questo campo è un campo di richiesta utilizzato solo per programmi DPL. La lunghezza di questo campo è fornita da MQ_FORMAT_LENGTH. Il valore iniziale di questo campo è MQFMT_NONE.

ID RemoteSys(MQCHAR4) per MQCIH
Questo campo mostra l'identificativo di sistema CICS del sistema CICS che elabora la richiesta.

Se questo campo è vuoto, la richiesta di sistema CICS viene elaborata sullo stesso sistema CICS del monitor bridge. Il SYSID utilizzato viene restituito nel messaggio di risposta.

Per una pseudo - conversazione 3270, tutti i messaggi successivi nella conversazione devono specificare il SYSID remoto restituito nella risposta iniziale. Se specificato, il SYSID deve:

- Essere attivi.
- Avere accesso alla coda di richiesta IBM MQ .
- Essere accessibili dai link ISC CICS dal sistema CICS del monitor bridge.

ID RemoteTrans(MQCHAR4) per MQCIH
Questo campo è un campo Richiesta facoltativo. La lunghezza di questo campo è fornita da MQ_TRANSACTION_ID_LENGTH.

Se specificato, il campo viene utilizzato come valore RTRANSID di CICS START.

TransactionId (MQCHAR4) per MQCIH
Questo campo è un campo di richiesta. La sua lunghezza è fornita da MQ_TRANSACTION_ID_LENGTH. Il valore iniziale di questo campo è di quattro spazi.

Se *LinkType* ha il valore MQCLT_TRANSACTION, *TransactionId* è l'identificativo della transazione dell'utente da eseguire; in questo caso, specificare un valore non vuoto.

Se *LinkType* ha il valore MQCLT_PROGRAM, *TransactionId* è il codice transazione con cui devono essere eseguiti tutti i programmi all'interno dell'unità di lavoro. Se si specifica un valore vuoto, viene utilizzato il CKBP (CICS DPL bridge default transaction code). Se il valore non è vuoto, è necessario che sia stato definito in CICS come transazione locale con un programma iniziale CSQCBP00. Questo campo si applica solo quando *UOWControl* ha il valore MQCUOWC_FIRST o MQCUOWC_ONLY.

FacilityLike (MQCHAR4) per MQCIH

FacilityLike è il nome di un terminale installato da utilizzare come un modello per la funzione bridge.

Un valore vuoto indica che *FacilityLike* viene preso dalla definizione del profilo di transazione bridge oppure viene utilizzato un valore predefinito.

Questo campo è un campo di richiesta utilizzato solo per transazioni 3270. La lunghezza di questo campo è fornita da MQ_FACILITY_LIKE_LENGTH. Il valore iniziale di questo campo è di quattro spazi.

AttentionId (MQCHAR4) per MQCIH

Il valore in questo campo determina il valore iniziale della chiave AID quando viene avviata la transazione. È un valore di 1 byte, allineato a sinistra.

AttentionId è un campo di richiesta utilizzato solo per transazioni 3270. La lunghezza di questo campo è data da MQ_ATTENION_ID_LENGTH. Il valore iniziale di questo campo è di quattro spazi.

StartCode (MQCHAR4) per MQCIH

Il valore di questo campo è un indicatore che specifica se il bridge emula una transazione terminale o una transazione avviata con START.

Il valore deve essere uno dei seguenti.

INIZIO_MQCSC
Avvio.

DATI STAR MQCSC
Avviare i dati.

MQCSC_TERMINPUT
Input terminale.

MQCSC_NONE
Nessuna.

In tutti i casi, per il linguaggio di programmazione C vengono definite anche le costanti MQCSC_ * _ARRAY; queste costanti hanno gli stessi valori delle costanti MQCSC_ * corrispondenti, ma sono array di caratteri invece di stringhe.

Nella risposta dal bridge, questo campo è impostato sul codice di avvio appropriato per l'ID transazione successivo contenuto nel campo *NextTransactionId* . I seguenti codici di avvio sono possibili nella risposta:

- INIZIO_MQCSC
- DATI STAR MQCSC
- MQCSC_TERMINPUT

Per CICS Transaction Server 1.2, questo campo è solo un campo di richiesta; il suo valore nella risposta non è definito.

Per CICS Transaction Server 1.3 e release successive, questo campo è sia un campo di richiesta che un campo di risposta.

Questo campo viene utilizzato solo per transazioni 3270. La lunghezza di questo campo è fornita da MQ_START_CODE_LENGTH. Il valore iniziale di questo campo è MQCSC_NONE.

CancelCode (MQCHAR4) per MQCIH
Il valore in questo campo è il codice di abend da utilizzare per terminare la transazione (normalmente una transazione conversazionale che richiede più dati). Altrimenti, questo campo è impostato su spazi vuoti.

Questo campo è un campo di richiesta utilizzato solo per transazioni 3270. La lunghezza di questo campo è fornita da MQ_CANCEL_CODE_LENGTH. Il valore iniziale di questo campo è di quattro spazi.

ID NextTransaction(MQCHAR4) per MQCIH
Questo valore è il nome della transazione successiva restituita dalla transazione utente (di solito da EXEC CICS RETURN TRANSID). Se non vi è alcuna transazione successiva, questo campo viene impostato su spazi vuoti.

Questo campo è un campo di risposta utilizzato solo per transazioni 3270. La lunghezza di questo campo è fornita da MQ_TRANSACTION_ID_LENGTH. Il valore iniziale di questo campo è di quattro spazi.

Reserved2 (MQCHAR8) per MQCIH
Questo campo è riservato. Il valore deve essere di 8 spazi.

Reserved3 (MQCHAR8) per MQCIH
Questo campo è riservato. Il valore deve essere di 8 spazi.

CursorPosition (MQLONG) per MQCIH
Il valore in questo campo mostra la posizione iniziale del cursore quando la transazione viene avviata. Per le transazioni interattive, la posizione del cursore è nel vettore RECEIVE.

Questo campo è un campo di richiesta utilizzato solo per transazioni 3270. Il valore iniziale di questo campo è 0. Questo campo non è presente se *Version* è minore di MQCIH_VERSION_2.

ErrorOffset (MQLONG) per MQCIH
Il campo ErrorOffset mostra la posizione dei dati non validi rilevati dall'uscita bridge. Questo campo fornisce lo scostamento dall'inizio del messaggio alla posizione dei dati non validi.

ErrorOffset è un campo di risposta utilizzato solo per transazioni 3270. Il valore iniziale di questo campo è 0. Questo campo non è presente se *Version* è minore di MQCIH_VERSION_2.

InputItem (MQLONG) per MQCIH
Questo campo è riservato. Il valore deve essere 0.

Questo campo non è presente se *Version* è minore di MQCIH_VERSION_2.

Reserved4 (MQLONG) per MQCIH
Questo campo è riservato. Il valore deve essere 0.

Questo campo non è presente se *Version* è minore di MQCIH_VERSION_2.

MQCMHO - Opzioni di creazione dell'handle del messaggio

La struttura **MQCMHO** consente alle applicazioni di specificare le opzioni che controllano il modo in cui vengono creati gli handle del messaggio. La struttura è un parametro di immissione sulla chiamata **MQCRTMH** .

Disponibilità
La struttura **MQCMHO** è disponibile sulle piattaforme seguenti:

- AIX
- IBM i
- Linux
- Windows
- z/OS

e per IBM MQ MQI clients collegati a questi sistemi.

Serie di caratteri e codifica

I dati in **MQCMHO** devono essere nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (**MQENC_NATIVE**).

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 478. Campi in MQCMHO

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQCMHO_	'CMHO'
Versione (numero versione struttura)	MQCMHO_VERSION_1	1
Opzioni (opzioni)	MQCMHO_DEFAULT_VALIDATION	0

Note:

1. Nel linguaggio di programmazione C, la variabile macroMQCMHO_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura:

```
MQCMHO MyCMHO = {MQCMHO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQCMHO

```
struct tagMQCMHO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of MQCRTMH */
};
```

Dichiarazione COBOL per MQCMHO

```
**   MQCMHO structure
  10 MQCMHO.
**    Structure identifier
   15 MQCMHO-STRUCID        PIC X(4).
**    Structure version number
   15 MQCMHO-VERSION        PIC S9(9) BINARY.
```

```
**    Options that control the action of MQCRTMH
   15 MQCMHO-OPTIONS        PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQCMHO

```
dcl
 1 MQCMHO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action of MQCRTMH */
```

Dichiarazione High Level Assembler per MQCMHO

```
MQCMHO                DSECT
MQCMHO_STRUCID        DS   CL4   Structure identifier
MQCMHO_VERSION        DS   F     Structure version number
MQCMHO_OPTIONS        DS   F     Options that control the action of
*                                MQCRTMH
MQCMHO_LENGTH         EQU  *-MQCMHO
MQCMHO_AREA           DS   CL(MQCMHO_LENGTH)
```

StrucId (MQCHAR4) per MQCMHO

Questo è l'identificativo della struttura delle opzioni di creazione dell'handle del messaggio. È sempre un campo di immissione. Il suo valore è MQCMHO_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQCMHO_
Identificativo per la struttura delle opzioni di gestione del messaggio.

Per il linguaggio di programmazione C, viene definita anche la costante **MQCMHO_STRUC_ID_ARRAY** . Ha lo stesso valore di **MQCMHO_STRUC_ID**, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQCMHO

Questo campo è sempre un campo di input. Il suo valore iniziale è MQCMHO_VERSION_1.

Questo è il numero di versione della struttura; il valore deve essere:

MQCMHO_VERSION_1
Version-1 creare la struttura di opzioni dell'handle del messaggio.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQCMHO_CURRENT_
Versione corrente della struttura delle opzioni di creazione dell'handle del messaggio.

Opzioni (MQLONG) per MQCMHO

Questo campo è sempre un campo di input. Il valore iniziale è MQCMHO_DEFAULT_VALIDATION.

È possibile specificare una delle seguenti opzioni:

VALIDATE MQCMHO_

Quando **MQSETMP** viene chiamato per impostare una proprietà in questo handle del messaggio, il nome della proprietà viene convalidato per garantire che:

- non contiene caratteri non validi.
- non inizia con JMS o usr.JMS ad eccezione dei seguenti:
  - JMSCorrelationID
  - JMSReplyTo

  - JMSType
  - JMSXGroupID
  - JMSXGroupSeq

  Questi nomi sono riservati per proprietà JMS .
- non è una delle seguenti parole chiave, in qualsiasi combinazione di maiuscolo o minuscolo:
  - E
  - TRA
  - ESCAPE
  - FALSO
  - IN
  - È
  - SIMILE A
  - NON
  - NULL
  - OPPURE
  - VERO
- non inizia il body. o Root. (tranne per Root.MQMD.).

Se la proprietà è definita da MQ(mq. *) e il nome viene riconosciuto, i campi descrittore della proprietà sono impostati sui valori corretti per la proprietà. Se la proprietà non è riconosciuta, il campo *Support* del descrittore della proprietà è impostato su **MQPD_OPTIONAL**.

MQCMHO_DEFAULT_VALIDATION

Questo valore specifica che si verifica il livello di convalida predefinito dei nomi delle proprietà.

Il livello predefinito di convalida è equivalente al livello specificato da **MQCMHO_VALIDATE**.

Questo è il valore predefinito.

MQCMHO_NO_VALIDATION

Non viene eseguita alcuna convalida sul nome della proprietà. Consultare la descrizione di **MQCMHO_VALIDATE**.

Opzione predefinita: se nessuna delle opzioni precedenti descritte è richiesta, è possibile utilizzare la seguente opzione:

MQCMHO_NONE

Tutte le opzioni assumono i valori predefiniti. Utilizzare questo valore per indicare che non sono state specificate altre opzioni. **MQCMHO_NONE** supporta la documentazione del programma; non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

MQCNO - Opzioni di connessione

La struttura MQCNO consente all'applicazione di specificare le opzioni relative alla connessione al gestore code. La struttura è un parametro di input / output sulla chiamata MQCONNX.

Per ulteriori informazioni sull'utilizzo degli handle condivisi e sulla chiamata MQCONNX, consultare Connessioni condivise (indipendenti dal thread) con MQCONNX.

Disponibilità

La struttura MQCNO è disponibile sulle piattaforme seguenti:

- AIX

-  IBM i
-  Linux
-  Windows

e per IBM MQ MQI clients collegati a questi sistemi.

Versione

I file di intestazione, COPY e INCLUDE forniti per i linguaggi di programmazione supportati contengono la versione più recente di MQCNO, ma con il valore iniziale del campo *Version* impostato su MQCNO_VERSION_1. Per utilizzare i campi che non sono presenti nella struttura version-1 , l'applicazione deve impostare il campo *Version* sul numero di versione richiesto.

Serie di caratteri e codifica

I dati in MQCNO devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da MQENC_NATIVE. Tuttavia, se l'applicazione è in esecuzione come IBM MQ MQI client, la struttura deve essere nella serie di caratteri e nella codifica del client.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 479. Campi in MQCNO per MQCNO

Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
StrucId (identificativo della struttura)	ID_STRUC_MQCNO	'CNO¬'
Versione (numero versione struttura)	MQCNO_VERSION_1	1
Opzioni (opzioni che controllano l'azione di MQCONNX)	MQCNO_NONE	0
Nota: I restanti campi vengono ignorati se *Version* è minore di MQCNO_VERSION_2.		
ClientConnOffset (offset della struttura MQCD per la connessione client)	Nessuna	0
ClientConnPtr (indirizzo della struttura MQCD per la connessione client)	Nessuna	Puntatore null o byte null
Nota: I restanti campi vengono ignorati se *Version* è minore di MQCNO_VERSION_3.		
ConnTag (tag di connessione del gestore code)	MQCT_NONE	Valori null
Nota: I restanti campi vengono ignorati se *Version* è minore di MQCNO_VERSION_4.		
SSLConfigPtr (indirizzo della struttura MQSCO per la connessione client)	Nessuna	Puntatore null o byte null
SSLConfigOffset (offset della struttura MQSCO per la connessione client)	Nessuna	0
Nota: I restanti campi vengono ignorati se *Version* è inferiore a MQCNO_VERSION_5.		

Tabella 479. Campi in MQCNO per MQCNO (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
ConnectionId (ID connessione univoco)	Nessuna	Puntatore null o byte null
SecurityParmsSecurityParms (offset della struttura MQSCO per i parametri di protezione)	Nessuna	Puntatore null o byte null
SecurityParmsPtr (indirizzo della struttura MQSCO per i parametri di sicurezza)	Nessuna	Puntatore null o byte null
Nota: I restanti campi vengono ignorati se *Version* è minore di MQCNO_VERSION_6.		
Riservato (campo riservato)	Nessuna	Campo riservato per inserire la struttura in un limite a 64 bit.
CCDTUrlLength (lunghezza URL CCDT)	Nessuna	Lunghezza della stringa identificata da *CCDTUrlPtr* o *CCDTUrlOffset*
CCDTUrlPtr (puntatore URL CCDT)	Nessuna	Puntatore a una stringa che contiene un URL, per identificare l'ubicazione della tabella del canale di connessione client da utilizzare per la connessione.
CCDTUrlOffset (offset URL CCDT)	Nessuna	Offset in byte da una stringa che contiene un URL che identifica l'ubicazione della tabella del canale di connessione client da utilizzare per la connessione.
Nota: I restanti campi vengono ignorati se *Version* è inferiore a MQCNO_VERSION_7.		
ApplName (nome impostato dall'applicazione)	Nessuna	Nome impostato dall'applicazione per identificare la connessione al gestore code
Reserved2 (campo riservato)	Nessuna	Campo riservato per inserire la struttura in un limite a 64 bit.
Nota: I campi rimanenti vengono ignorati se *Version* è minore di MQCNO_VERSION_8.		
BalanceParms| Offset	Nessuna	Offset in byte per la struttura MQBNO

Tabella 479. Campi in MQCNO per MQCNO (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
Ptr BalanceParms	Nessuna	Puntatore all'ubicazione della struttura MQBNO

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macroMQCNO_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQCNO MyCNO = {MQCNO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQCNO

```
typedef struct tagMQCNO MQCNO;
struct tagMQCNO {
  MQCHAR4    StrucId;           /* Structure identifier */
  MQLONG     Version;           /* Structure version number */
  MQLONG     Options;           /* Options that control the action of
                                   MQCONNX */
  MQLONG     ClientConnOffset;  /* Offset of MQCD structure for client
                                   connection */
  MQPTR      ClientConnPtr;     /* Address of MQCD structure for client
                                   connection */
  MQBYTE128  ConnTag;           /* Queue managerconnection tag */
  PMQSCO     SSLConfigPtr;      /* Address of MQSCO structure for client
                                   connection */
  MQLONG     SSLConfigOffset;   /* Offset of MQSCO structure for client
                                   connection */
  MQBYTE24   ConnectionId;      /* Unique connection identifier */
  MQLONG     SecurityParmsOffset /* Security fields */
  PMQCSP     SecurityParmsPtr   /* Security parameters */
  MQLONG     CCDTUrlLength      /* Length of string identified by Ptr or offset */
  MQLONG     CCDTUrlOffset      /* Offset in bytes to URL of client connection channel */
  PMQURL     CCDTUrlPtr         /* Address of string containing URL */
  MQBYTE4    Reserved           /* Reserved field to pad out to 64 bit boundary */
  MQCHAR28   ApplName           /* Name set by the application to identify the connection to
                                   the queue manager */
  MQBYTE4    Reserved2          /* Reserved field to pad out to 64 bit boundary */
  MQLONG     BalanceParmsOffset /* Offset of the MQBMO structure */
  PMQBMO     BalanceParmsPtr    /* Address of the location of the MQBMO structure */
};
```

Dichiarazione COBOL per MQCNO

```
**   MQCNO structure
  10 MQCNO.
**    Structure identifier
   15 MQCNO-STRUCID          PIC X(4).
**    Structure version number
   15 MQCNO-VERSION          PIC S9(9) BINARY.
**    Options that control the action of MQCONNX
   15 MQCNO-OPTIONS          PIC S9(9) BINARY.
**    Offset of MQCD structure for client connection
   15 MQCNO-CLIENTCONNOFFSET PIC S9(9) BINARY.
**    Address of MQCD structure for client connection
   15 MQCNO-CLIENTCONNPTR    POINTER.
**    Queue manager connection tag
   15 MQCNO-CONNTAG          PIC X(128).
**    Address of MQSCO structure for client connection
   15 MQCNO-SSLCONFIGPTR     POINTER.
```

```
**    Offset of MQSCO structure for client connection
   15 MQCNO-SSLCONFIGOFFSET  PIC S9(9) BINARY.
**    Unique connection identifier
   15 MQCNO-CONNECTIONID     PIC X(24).
**    Offset of MQCSP structure for security parameters
   15 MQCNO-SECURITYPARMSOFFSET PIC S9(9) BINARY.
**    Address of MQCSP structure for security parameters
   15 MQCNO-SECURITYPARMSPTR POINTER.
**    Length of string identified by CCDTURLOFFSET or CCDTURLPTR
   15 MQCNO-CCDTURLLENGTH
**    Pointer to a string which contains a URL, to identify the location of the client
connection channel
   15 MQCNO-CCDTURLPTR
**    Address of string which contains a URL that identifies the location of the client
connection channel table
   15 MQCNO-CCDTURLOFFSET
**    Reserved field to pad to 64 bit boundary
   15 MQCNO-RESERVED
**    Name set by the application to identify the connection to the queue manager
   15 MQCNO-APPLNAME
**    Reserved field to pad to 64 bit boundary
   15 MQCNO-RESERVED2
**    Address of the MQBMO structure
   15 MQCNO-BALANCEPARMSOFFSET
**    Pointer to the MQBMO structure
   15 MQCNO-BALANCEPARMSPTR
```

Dichiarazione PL/I per MQCNO

```
dcl
 1 MQCNO based,
  3 StrucId            char(4),        /* Structure identifier */
  3 Version            fixed bin(31),  /* Structure version number */
  3 Options            fixed bin(31),  /* Options that control the action
                                          of MQCONNX */
  3 ClientConnOffset   fixed bin(31),  /* Offset of MQCD structure for
                                          client connection */
  3 ClientConnPtr      pointer,        /* Address of MQCD structure for
                                          client connection */
  3 ConnTag            char(128),      /* Queue managerconnection tag */
  3 SSLConfigPtr       pointer,        /* Address of MQSCO structure for
                                          client connection */
  3 SSLConfigOffset    fixed bin(31),  /* Offset of MQSCO structure for
                                          client connection */
  3 ConnectionId       char(24),       /* Unique connection identifier
  3 SecurityParmsOffset fixed bin(31)  /* Offset of MQCSP structure for
                                          security parameters */
  3 SecurityParmsPtr   pointer,        /* Address of MQCSP structure for
                                          security parameters */
  3 CCDTUrlLength      fixed bin(31)   /* Length of string identified by CCDTUrlPtr
                                          or CCDTUrlOffset */
  3 CCDTUrlOffset      fixed bin(31)   /* Offset in bytes to URL of client connection channel */
  3 CCDTUrlPtr         pointer         /* Pointer to string containing URL */
  3 Reserved           char(4)         /* Reserved field to pad out to 64 bit boundary */
  3 ApplName           char(28)        /* Name set by the application to identify the
connection to
                                          the queue manager */
  3 Reserved2          char(4)         /* Reserved field to pad out to 64 bit boundary */
  3 BalanceParmsOffset fixed bin(31)   /* Offset of the MQBMO structure */
  3 BalanceParmsPtr    pointer         /* Address of the MQBMO structure */
```

Dichiarazione High Level Assembler per MQCNO

```
MQCNO                    DSECT
MQCNO_STRUCID            DS   CL4    Structure identifier
MQCNO_VERSION            DS   F      Structure version number
MQCNO_OPTIONS            DS   F      Options that control the action of
*                                     MQCONNX
MQCNO_CLIENTCONNOFFSET   DS   F      Offset of MQCD structure for client
*                                     connection
MQCNO_CLIENTCONNPTR      DS   F      Address of MQCD structure for client
*                                     connection
MQCNO_CONNTAG            DS   XL128  Queue manager connection tag
MQCNO_CONNECTIONID       DS   XL24   Unique connection identifier
MQCNO_SSLCONFIGOFFSET    DS   F      Offset of MQCSP structure for security
*                                     parameters
```

```
MQCNO_SSLCONFIGPTR        DS   F      Address of MQCSP structure for security
*                                     parameters
MQCNO_LENGTH              EQU  *-MQCNO
                          ORG  MQCNO
MQCNO_AREA                DS   CL(MQCNO_LENGTH)
MQCNO_CCDTURLLENGTH       DS   F      Length of string identified by CCDTURLPTR or
*                                     CCDTURLOFFSET
MQCNO_CCDTURLOFFSET       DS   F      Offset in bytes to URL of client connection channel
MQCNO_CCDTURLPTR          DS   F      Pointer to string containing URL
RESERVED                  DS   XL4    Reserved field to pad out to 64 bit boundary
APPLNAME                  DS   CL28   Name set by the application to identify the connection to
*                                     the queue manager
RESERVED2                 DS   XL4    Reserved field to pad out to 64 bit boundary
MQCNO_BALANCEPARMSOFFSET  DS   F      Offset of the MQBMO structure
MQCNO_BALANCEPARMSPTR     DS   F      Address of the MQBMO structure
```

Dichiarazione di Visual Basic per MQCNO

```
Type MQCNO
  StrucId             As String*4  'Structure identifier'
  Version             As Long      'Structure version number'
  Options             As Long      'Options that control the action of'
                                   'MQCONNX'
  ClientConnOffset    As Long      'Offset of MQCD structure for client'
                                   'connection'
  ClientConnPtr       As MQPTR     'Address of MQCD structure for client'
                                   'connection'
  ConnTag             As MQBYTE128 'Queue manager connection tag'
  SSLConfigPtr        As MQPTR     'Address of MQSCO structure for client'
                                   'connection'
  SSLConfigOffset     As Long      'Offset of MQSCO structure for client'
                                   'connection'
  ConnectionId        As MQBYTE24  'Unique connection identifier'
  SecurityParmsOffset As Long      'Offset of MQCSP structure for security'
                                   'parameters'
  SecurityParmsPtr    As MQPTR     'Address of MQCSP structure for security'
                                   'parameters'
  CCDTUrlLength       As Long      'Length of string identified by CCDTUrlPtr'
                                   'or CCDTUrlOffset'
  CCDTUrlOffset       As Long      'Offset in bytes to URL of client connection channel'
  CCDTUrlPtr          As MQPTR     'Pointer to string containing URL'
  Reserved            As MQBYTE4   'Reserved field to pad out to 64 bit boundary'
  ApplName            As String*28 'Name set by the application to identify the connection to'
                                   'the queue manager'
  Reserved2           As MQBYTE4   'Reserved field to pad out to 64 bit boundary'
  BalanceParmsOffset  As Long      'Offset in bytes to MQBNO structure'
  BalanceParmsPtr     As MQPTR     'Address of MQBNO structure'
  End Type
```

Attività correlate

Utilizzo di MQCONNX

StrucId (MQCHAR4) per MQCNO

Questo è l'identificativo della struttura delle opzioni di connessione. È sempre un campo di immissione. Il valore è MQCNO_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQCNO
Identificativo per la struttura di opzioni di collegamento.

Per il linguaggio di programmazione C, viene definita anche la costante MQCNO_STRUC_ID_ARRAY. Questa costante ha lo stesso valore di MQCNO_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQCNO

La versione è sempre un campo di input. Il valore iniziale è MQCNO_VERSION_1.

Il valore deve essere uno dei seguenti.

MQCNO_VERSION_1
Struttura delle opzioni di connessione Version-1 .

MQCNO_VERSION_2
Struttura Version-2 connect - options.

MQCNO_VERSION_3
Version-3 struttura connect - options.

MQCNO_VERSION_4
Version-4 struttura delle opzioni di connessione.

MQCNO_VERSION_5
Version-5 struttura delle opzioni di connessione.

MQCNO_VERSION_6
Version-6 struttura connect - options.

MQCNO_VERSION_7
Version-7 struttura delle opzioni di connessione.

MQCNO_VERSION_8
Version-8 struttura delle opzioni di connessione.

I campi che esistono solo nelle versioni più recenti della struttura vengono identificati come tali nelle descrizioni dei campi. La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQCNO_CURRENT_
Versione corrente della struttura delle opzioni di connessione.

Opzioni (MQLONG) per MQCNO

Opzioni che controllano l'azione di MQCONNX.

Opzioni di contabilizzazione

Le seguenti opzioni controllano il tipo di account se l'attributo del Gestore code **AccountingConnOverride** è impostato su MQMON_ENABLED:

MQCNO_ACCOUNTING_MQI_ENABLED
Quando la raccolta dati di controllo è disabilitata nella definizione del gestore code impostando l'attributo **MQIAccounting** su MQMON_OFF, l'impostazione di questo indicatore abilita la raccolta dati di account MQI.

MQCNO_ACCOUNTING_MQI_DISABLED
Quando il monitoraggio della raccolta dati è disabilitato nella definizione del gestore code impostando l'attributo **MQIAccounting** su MQMON_OFF, l'impostazione di questo indicatore arresta la raccolta dei dati di account MQI.

MQCNO_ACCOUNTING_Q_ENABLED
Quando la raccolta dei dati di account della coda è disabilitata nella definizione del gestore code impostando l'attributo **MQIAccounting** su MQMON_OFF, l'impostazione di questo indicatore abilita la raccolta dei dati di account per le code che specificano un gestore code nel campo *MQIAccounting* della relativa definizione di coda.

MQCNO_ACCOUNTING_Q_DISABLED
Quando la raccolta dati di account della coda è disabilitata nella definizione del gestore code impostando l'attributo **MQIAccounting** su MQMON_OFF, l'impostazione di questo indicatore disattiva la raccolta dati di account per le code che specificano un gestore code nel campo *MQIAccounting* della relativa definizione della coda.

Se non viene definito nessuno di questi indicatori, l'account per la connessione è quello definito negli attributi del gestore code.

Opzioni di bind

Le seguenti opzioni controllano il tipo di bind IBM MQ da utilizzare. Specificare solo una delle seguenti opzioni:

MQCNO_STANDARD_BINDING
L'applicazione e l'agent del gestore code locale (il componente che gestisce le operazioni di accodamento) vengono eseguiti in unità di esecuzione separate (generalmente, in processi separati). Questa disposizione mantiene l'integrit ... del gestore code, vale a dire, protegge il gestore code dai programmi erranti.

Se il gestore code supporta più tipi di bind e si imposta MQCNO_STANDARD_BINDING, il gestore code utilizza l'attributo **DefaultBindType** nella stanza `Connection` nel file `qm.ini` per selezionare il tipo effettivo di bind. Se questa stanza non è definita o il valore non può essere utilizzato o non è appropriato per l'applicazione, il gestore code seleziona un tipo di bind appropriato. Il gestore code imposta il tipo di bind effettivo utilizzato nelle opzioni di connessione.

Utilizzare MQCNO_STANDARD_BINDING in situazioni in cui l'applicazione potrebbe non essere stata completamente verificata o potrebbe essere inaffidabile o non affidabile. MQCNO_STANDARD_BINDING è il valore predefinito.

Questa opzione è supportata in tutti gli ambienti.

Se si sta effettuando il collegamento alla libreria `mqm` , viene tentata prima una connessione al server standard utilizzando il tipo di collegamento predefinito. Se non è stato possibile caricare la libreria del server sottostante, viene tentata una connessione client.

- Per modificare il comportamento di MQCONN (o MQCONNX se è specificato MQCNO_STANDARD_BINDING), impostare la variabile di ambiente MQ_CONNECT_TYPE su una delle seguenti opzioni. Notare che è presente un'eccezione: se MQCNO_FASTPATH_BINDING viene specificato con MQ_CONNECT_TYPE impostato su LOCAL o STANDARD, le connessioni fastpath possono essere declassate dall'amministratore senza una modifica correlata all'applicazione.

Tabella 480. Valori per MQ_CONNECT_TYPE che modificano il comportamento di MQCONN o MQCONNX

Valore	**Significato**
CLIENTE	Viene tentato solo un collegamento client.
Percorso veloce	Questo valore era supportato nei release precedenti, ma viene ora ignorato se specificato.
LOCALE	Viene tentata solo una connessione server. Le connessioni Fastpath vengono ridotte a una connessione server standard.
STANDARD	Supportato per la compatibilità con le release precedenti. Questo valore viene ora considerato come LOCAL.

- Se la variabile di ambiente MQ_CONNECT_TYPE non è impostata quando viene richiamato MQCONNX, viene tentata una connessione server standard che utilizza il tipo di collegamento predefinito. Se il caricamento della libreria del server non riesce, viene tentata una connessione client.

MQCNO_FASTPATH_BINDING
L'applicazione e il gestore code locale fanno parte della stessa unità di esecuzione. Ciò è in contrasto con il metodo tipico di bind, in cui l'applicazione e l'agent del gestore code locale vengono eseguiti in unità di esecuzione separate.

MQCNO_FASTPATH_BINDING viene ignorato se il gestore code non supporta questo tipo di bind; l'elaborazione continua come se l'opzione non fosse stata specificata.

MQCNO_FASTPATH_BINDING può essere vantaggioso in situazioni in cui più processi consumano più risorse rispetto alla risorsa complessiva utilizzata dall'applicazione.Un'applicazione che utilizza il collegamento fastpath è nota come *applicazione sicura*.

Considerare i seguenti punti importanti quando si decide se utilizzare il collegamento fastpath:

- L'uso dell'opzione MQCNO_FASTPATH_BINDING non impedisce a un'applicazione di modificare o corrompere i messaggi e altre aree dati appartenenti al gestore code. Utilizzare questa opzione solo in situazioni in cui questi problemi sono stati completamente valutati.
- L'applicazione non deve utilizzare segnali asincroni o interruzioni del timer (come `sigkill`) con MQCNO_FASTPATH_BINDING. Esistono anche delle limitazioni sull'utilizzo dei segmenti di memoria condivisa.
- L'applicazione deve utilizzare la chiamata MQDISC per disconnettersi dal gestore code.
- L'applicazione deve essere terminata prima di terminare il gestore code con il comando `endmqm` .
- IBM i Su IBM i, il lavoro deve essere eseguito con un profilo utente appartenente al gruppo `QMQMADM` . Inoltre, il programma non deve arrestarsi in modo anomalo, altrimenti possono verificarsi risultati imprevedibili.
- Linux AIX Su AIX and Linux, l'identificativo utente `mqm` deve essere l'ID utente effettivo e l'identificativo gruppo `mqm` deve essere l'identificativo gruppo effettivo. Per eseguire l'applicazione in questo modo, configurare il programma in modo che appartenga all'identificativo utente `mqm` e all'identificativo gruppo `mqm` , quindi impostare i bit di autorizzazione `setuid` e `setgid` sul programma.

  IBM MQ OAM (Object Authority Manager) utilizza ancora l'ID utente reale per il controllo dell'autorizzazione.
- Windows Su Windows, il programma deve essere un membro del gruppo `mqm` . Il collegamento Fastpath non è supportato per le applicazioni a 64 bit.

L'opzione MQCNO_FASTPATH_BINDING è supportato nei seguenti ambienti:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Windows Windows

z/OS Su z/OS, l'opzione è accettata ma ignorata.

Per ulteriori informazioni sulle implicazioni dell'utilizzo di applicazioni attendibili, consultare Limitazioni per le applicazioni attendibili.

MQCNO_SHARED_BINDING

Con MQCNO_SHARED_BINDING, le applicazioni e l'agente del gestore code locale condividono alcune risorse. MQCNO_SHARED_BINDING viene ignorato se il gestore code non supporta questo tipo di bind. L'elaborazione continua come se l'opzione non fosse stata specificata.

MQCNO_ISOLATED_BINDING

In tal caso, il processo dell'applicazione e l'agente del gestore code locale sono isolati l'uno dall'altro in quanto non condividono risorse. MQCNO_ISOLATED_BINDING viene ignorato se il gestore code non supporta questo tipo di bind. L'elaborazione continua come se l'opzione non fosse stata specificata.

MQCNO_CLIENT_BINDING

Specificare questa opzione per rendere il tentativo dell'applicazione solo una connessione client. Questa opzione ha i seguenti limiti:

- z/OS MQCNO_CLIENT_BINDING viene ignorato su z/OS.
- MQCNO_CLIENT_BINDING viene rifiutato con MQRC_OPTIONS_ERROR se è specificato con qualsiasi opzione di bind MQCNO diversa da MQCNO_STANDARD_BINDING.
- MQCNO_CLIENT_BINDING non è disponibile per Java o .NET poiché dispongono di meccanismi propri per la selezione del tipo di bind.

BINDING MQCNO_LOCAL_

Specificare questa opzione per fare in modo che l'applicazione tenti una connessione server. Se è stato specificato anche MQCNO_FASTPATH_BINDING, MQCNO_ISOLATED_BINDING o MQCNO_SHARED_BINDING, la connessione è di quel tipo ed è documentata in questa sezione. Altrimenti viene tentata una connessione al server standard utilizzando il tipo di bind predefinito. MQCNO_LOCAL_BINDING ha i seguenti limiti:

- z/OS MQCNO_LOCAL_BINDING viene ignorato su z/OS.
- MQCNO_LOCAL_BINDING viene rifiutato con MQRC_OPTIONS_ERROR se specificato con qualsiasi opzione di riconnessione MQCNO diversa da MQCNO_RECONNECT_AS_DEF.
- MQCNO_LOCAL_BINDING non è disponibile per Java o .NET in quanto dispongono di meccanismi propri per la scelta del tipo di bind.

Sulle seguenti piatteforme, è possibile utilizzare la variabile di ambiente `MQ_CONNECT_TYPE` con il tipo di collegamento specificato dal campo `Options` , per controllare il tipo di collegamento utilizzato.

- AIX AIX
- Linux Linux
- Windows Windows

Se si specifica questa variabile di ambiente, deve avere il valore `FASTPATH` o `STANDARD` ; se ha un valore diverso, viene ignorato. Il valore della variabile di ambiente è sensibile al maiuscolo / minuscolo; per ulteriori informazioni, consultare Variabile di ambiente MQCONNX .

La variabile di ambiente e il campo *`Options`* interagiscono come segue:

- Se si omette la variabile di ambiente o si fornisce un valore non supportato, l'utilizzo del bind del percorso rapido viene determinato esclusivamente dal campo `Options` .
- Se si fornisce alla variabile di ambiente un valore supportato, il collegamento fastpath viene utilizzato solo se la variabile di ambiente e il campo `Options` specificano il collegamento fastpath.

Opzioni tag di connessione

Le seguenti opzioni controllano l'uso della tag di collegamento `ConnTag`. Sono supportati solo durante la connessione a un gestore code IBM MQ for z/OS . Non sono validi se `Version` è minore di MQCNO_VERSION_3. Specificare solo una di queste opzioni. Se ci si connette da un gestore code Multiplatforms, specificare *MQCNO_GENERATE_CONN_TAG*.

Multi **MQCNO_GENERATE_CONN_TAG**

Restituisce la tag di connessione che il gestore code ha associato a questa connessione, nella struttura MQCNO di output.

La tag di connessione restituita è uguale per tutte le connessioni che il gestore code considera come una sola istanza dell'applicazione.

z/OS **MQCNO_SERIALIZE_CONN_TAG_Q_MGR**

Questa opzione richiede l'uso esclusivo della tag di connessione all'interno del gestore code locale. Se la tag di connessione è già in uso nel gestore code locale, la chiamata MQCONNX ha esito negativo con codice motivo MQRC_CONN_TAG_IN_USE. Il risultato della chiamata non viene influenzato dall'utilizzo della tag di connessione altrove nel gruppo di condivisione code a cui appartiene il gestore code locale.

z/OS **MQCNO_SERIALIZE_CONN_TAG_QSG**

Questa opzione richiede l'uso esclusivo della tag di connessione all'interno del gruppo di condivisione code a cui appartiene il gestore code locale. Se la tag di connessione è già in uso

nel gruppo di condivisione code, la chiamata MQCONNX ha esito negativo con codice motivo MQRC_CONN_TAG_IN_USE.

z/OS MQCNO_RESTRICT_CONN_TAG_Q_MGR

Questa opzione richiede l'uso condiviso della tag di connessione all'interno del gestore code locale. Se la tag di connessione è già in uso nel gestore code locale, la chiamata MQCONNX può avere esito positivo se l'applicazione richiedente è in esecuzione nello stesso ambito di elaborazione dell'utente esistente della tag. Se questa condizione non viene soddisfatta, la chiamata MQCONNX ha esito negativo con codice motivo MQRC_CONN_TAG_IN_USE. Il risultato della chiamata non è influenzato dall'utilizzo della tag di connessione altrove nel gruppo di condivisione code a cui appartiene il gestore code locale.

- Le applicazioni devono essere eseguite all'interno dello stesso spazio di indirizzo MVS per condividere la tag di connessione. Se l'applicazione che utilizza la tag di collegamento è un'applicazione client, MQCNO_RESTRICT_CONN_TAG_Q_MGR non è consentito.

z/OS MQCNO_RESTRICT_CONN_TAG_QSG

Questa opzione richiede l'uso condiviso della tag di connessione all'interno del gruppo di condivisione code a cui appartiene il gestore code locale. Se la tag di connessione è già in uso nel gruppo di condivisione code, la chiamata MQCONNX può essere eseguita correttamente se l'applicazione richiedente è in esecuzione nello stesso ambito di elaborazione ed è connessa allo stesso gestore code, come l'utente esistente della tag.

Se queste condizioni non vengono soddisfatte, la chiamata MQCONNX ha esito negativo con codice motivo MQRC_CONN_TAG_IN_USE.

- Le applicazioni devono essere eseguite all'interno dello stesso spazio di indirizzo MVS per condividere la tag di connessione. Se l'applicazione che utilizza la tag di connessione è un'applicazione client, MQCNO_RESTRICT_CONN_TAG_QSG non è consentito.

Se non viene specificata alcuna opzione, `ConnTag` non viene utilizzato.

Gestione delle opzioni di condivisione

Multi

Queste opzioni sono supportate nei seguenti ambienti:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Windows Windows

Controllano la condivisione di handle tra thread differenti (unità di elaborazione parallele) all'interno dello stesso processo. È possibile specificare solo una delle seguenti opzioni:

MQCNO_HANDLE_SHARE_NONE

Questa opzione indica che gli handle della connessione e dell'oggetto possono essere utilizzati solo dal thread che ha causato l'allocazione dell'handle (ovvero, il thread che ha emesso la chiamata MQCONN, MQCONNX o MQOPEN). Le maniglie non possono essere utilizzate da altri thread appartenenti allo stesso processo.

MQCNO_HANDLE_SHARE_BLOCK

Questa opzione indica che gli handle di connessione e di oggetto assegnati da un sottoprocesso di un processo possono essere utilizzati da altri sottoprocessi appartenenti allo stesso processo. Tuttavia, solo un thread alla volta può utilizzare un determinato handle; ovvero, è consentito solo l'utilizzo seriale di un handle. Se un sottoprocesso tenta di utilizzare una gestione che è già utilizzata da un altro sottoprocesso, la chiamata blocca (attende) fino a quando la gestione non diventa disponibile.

MQCNO_HANDLE_SHARE_NO_BLOCK

È uguale a MQCNO_HANDLE_SHARE_BLOCK, ad eccezione del fatto che se l'handle è utilizzato da un altro thread, la chiamata viene completata immediatamente con MQCC_FAILED e MQRC_CALL_IN_PROGRESS invece di bloccare fino a quando l'handle non diventa disponibile.

Un thread può avere zero o un handle non condiviso:

- Ogni chiamata MQCONN o MQCONNX che specifica MQCNO_HANDLE_SHARE_NONE restituisce un nuovo handle non condiviso sulla prima chiamata e lo stesso handle non condiviso sulla seconda e sulle successive chiamate (presumendo che non sia presente alcuna chiamata MQDISC). Il codice motivo è MQRC_ALREADY_CONNECTED per le seconde e successive chiamate.
- Ogni chiamata MQCONNX che specifica MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK restituisce un nuovo handle condiviso per ciascuna chiamata.

I gestori oggetti ereditano le stesse proprietà di condivisione del gestore connessioni specificato nella chiamata MQOPEN che ha creato il gestore oggetti. Inoltre, le unità di lavoro ereditano le stesse proprietà di condivisione dell'handle di connessione utilizzato per avviare l'unità di lavoro; se l'unità di lavoro viene avviata in un thread utilizzando un handle condiviso, l'unità di lavoro può essere aggiornata in un altro thread utilizzando lo stesso handle.

Se non si specifica un'opzione di condivisione della gestione, l'impostazione predefinita è determinata dall'ambiente:

- Windows Nell'ambiente Microsoft Transaction Server (MTS), il valore predefinito è lo stesso di MQCNO_HANDLE_SHARE_BLOCK.
- In altri ambienti, il valore predefinito è uguale a MQCNO_HANDLE_SHARE_NONE.

Opzioni di riconnessione

Le opzioni di riconnessione determinano se una connessione è ricollegabile. Solo le connessioni client sono ricollegabili.

MQCNO_RECONNECT_AS_DEF

L'opzione di riconnessione viene risolta nel valore predefinito. Se non è impostato alcun valore predefinito, il valore di questa opzione si risolve in DISABLED. Il valore dell'opzione viene passato al server e può essere sottoposto a query da PCF e MQSC.

MQCNO_RECONNECT

L'applicazione può essere riconnessa a qualsiasi gestore code congruente con il valore del parametro **QmgrName** di MQCONNX. Utilizzare l'opzione MQCNO_RECONNECT solo se non vi è alcuna affinità tra l'applicazione client e il gestore code con cui ha inizialmente stabilito una connessione. Il valore dell'opzione viene passato al server e può essere sottoposto a query da PCF e MQSC.

MQCNO_RECONNECT_DISABLED

Impossibile ricollegare l'applicazione. Il valore dell'opzione non viene passato al server.

MQCNO_RECONNECT_Q_MGR

L'applicazione può essere riconnessa solo al gestore code con cui si è originariamente connessa. Utilizzare questo valore se un client può essere riconnesso, ma c'è un'affinità tra l'applicazione client e il gestore code con cui ha originariamente stabilito una connessione. Selezionare questo valore se si desidera che un client si riconnetta automaticamente all'istanza in standby di un gestore code altamente disponibile. Il valore dell'opzione viene passato al server e può essere sottoposto a query da PCF e MQSC.

Utilizzare le opzioni MQCNO_RECONNECT, MQCNO_RECONNECT_DISABLED e MQCNO_RECONNECT_Q_MGR solo per connessioni client. Se le opzioni vengono utilizzate per una connessione di collegamento, MQCONNX ha esito negativo con il codice di completamento MQCC_FAILED e

il codice motivo MQRC_OPTIONS_ERROR. La riconnessione automatica del client non è supportata da IBM MQ classes for Java

Opzioni di condivisione conversazione

Le seguenti opzioni si applicano solo alle connessioni client TCP/IP. Per canali SNA, SPX e NetBios , questi valori vengono ignorati e il canale viene eseguito come nelle versioni precedenti del prodotto

MQCNO_NO_CONV_SHARING

Questa opzione non permette la condivisione della conversazione.

È possibile utilizzare MQCNO_NO_CONV_SHARING in situazioni in cui le conversazioni sono pesantemente caricate e, quindi, in cui il conflitto è una possibilità sull'estremità di connessione server dell'istanza del canale su cui esistono le conversazioni di condivisione. MQCNO_NO_CNV_SHARING funziona come SHARECNV (1) quando è connesso a un canale che supporta la condivisione della conversazione e SHARECNV (0) quando è connesso a un canale che non supporta la condivisione della conversazione.

MQCNO_ALL_CONVS_SHARE

Questa opzione consente la condivisione della conversazione; l'applicazione non pone alcun limite al numero di connessioni sull'istanza del canale. Questa opzione rappresenta il valore predefinito.

Se l'applicazione indica che l'istanza del canale può condividere, ma la definizione *SharingConversations* (SHARECNV) sull'estremità della connessione server del canale è impostata su uno, non si verifica alcuna condivisione e non viene fornita alcuna avvertenza all'applicazione.

Allo stesso modo, se l'applicazione indica che la condivisione è consentita ma la definizione di connessione server *SharingConversations* è impostata su zero, non viene fornita alcuna avvertenza e l'applicazione presenta lo stesso comportamento di un client nelle versioni precedenti a IBM WebSphere MQ 7.0; l'impostazione dell'applicazione relativa alle conversazioni di condivisione viene ignorata.

MQCNO_NO_CONV_SHARING e MQCNO_ALL_CONVS_SHARE si escludono a vicenda. Se entrambe le opzioni sono specificate su una connessione particolare, la connessione viene rifiutata con un codice motivo MQRC_OPTIONS_ERROR.

Opzioni di definizione del canale

Le opzioni seguenti controllano l'utilizzo della struttura di definizione del canale inoltrata in MQCNO:

MQCNO_CD_FOR_OUTPUT_ONLY

Questa opzione consente alla struttura di definizione del canale in MQCNO di essere utilizzata solo per restituire il nome del canale utilizzato su una chiamata MQCONNX riuscita.

Se non viene fornita una struttura di definizione di canale valida, la chiamata non riesce con il codice motivo MQRC_CD_ERROR.

Se l'applicazione non è in esecuzione come client, l'opzione viene ignorata.

Il nome canale restituito può essere utilizzato su una chiamata MQCONNX successiva utilizzando l'opzione MQCNO_USE_CD_SELECTION per riconnettersi utilizzando la stessa definizione di canale. Ciò può essere utile quando ci sono più definizioni di canale applicabili nella tabella del canale client.

MQCNO_USE_CD_SELECTION

Questa opzione permette alla chiamata MQCONNX di connettersi utilizzando il nome del canale contenuto nella struttura di definizione del canale inoltrata in MQCNO.

Se la variabile di ambiente MQSERVER è impostata, viene utilizzata la sua definizione di canale. Se MQSERVER non è impostato, viene utilizzata la tabella del canale client.

Se non viene trovata una definizione di canale con il nome del canale e il nome del gestore code corrispondenti, la chiamata non riesce con codice motivo MQRC_Q_MGR_NAME_ERROR.

Se non viene fornita una struttura di definizione di canale valida, la chiamata non riesce con il codice motivo MQRC_CD_ERROR.

Se l'applicazione non è in esecuzione come client, l'opzione viene ignorata.

Opzione predefinita

Se non è richiesta alcuna delle opzioni descritte in precedenza, è possibile utilizzare la seguente opzione:

MQCNO_NONE

Non è stata specificata alcuna opzione.

Utilizzare MQCNO_NONE per la documentazione del programma. Non è previsto che questa opzione venga utilizzata con altre opzioni MQCNO_ *, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Offset ClientConn(MQLONG) per MQCNO

L'offset ClientConnè l'offset in byte di una struttura di definizione di canale MQCD dall'inizio della struttura MQCNO. L'offset può essere positivo o negativo. Questo campo è un campo di input con un valore iniziale di 0.

Utilizzare *ClientConnOffset* solo quando l'applicazione che emette la chiamata MQCONNX è in esecuzione come IBM MQ MQI client. Per informazioni su come utilizzare questo campo, consultare la descrizione del campo *ClientConnPtr* .

Questo campo viene ignorato se *Version* è minore di MQCNO_VERSION_2.

ClientConnPtr (MQPTR) per MQCNO

ClientConnPtr è un campo di input. Il suo valore iniziale è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e una stringa di byte all - null in caso contrario.

Utilizzare *ClientConnOffset* e *ClientConnPtr* solo quando l'applicazione che emette la chiamata MQCONNX è in esecuzione come IBM MQ MQI client. Specificando uno o l'altro di questi campi, l'applicazione può controllare la definizione del canale di connessione client fornendo una struttura di definizione del canale MQCD che contiene i valori richiesti.

Se l'applicazione è in esecuzione come IBM MQ MQI client, ma non fornisce una struttura MQCD, la variabile di ambiente MQSERVER viene utilizzata per selezionare la definizione del canale. Se MQSERVER non è impostato, viene utilizzata la tabella del canale client.

Se l'applicazione non è in esecuzione come IBM MQ MQI client, *ClientConnOffset* e *ClientConnPtr* vengono ignorati.

Se l'applicazione fornisce una struttura MQCD, impostare i campi elencati sui valori richiesti; gli altri campi in MQCD vengono ignorati. È possibile riempire le stringhe di caratteri con spazi vuoti fino alla lunghezza del campo o terminarle con un carattere null. Consultare "Campi" a pagina 1522 per ulteriori informazioni sui campi nella struttura MQCD.

Tabella 481. Campi in MQCD

Campo in MQCD	Valore
ChannelName	Nome canale.
Version	Numero di versione della struttura. Non deve essere inferiore a MQCD_VERSION_7.
TransportType	Qualsiasi tipo di trasporto supportato.
ModeName	Nome modalità LU 6.2 .

Tabella 481. Campi in MQCD (Continua)

Campo in MQCD	Valore
TpName	Nome del programma di transazione LU 6.2 .
SecurityExit	Nome dell'uscita di sicurezza del canale.
SendExit	Nome dell'uscita di invio del canale.
ReceiveExit	Nome dell'uscita di ricezione del canale.
MaxMsgLength	La lunghezza massima in byte dei messaggi che possono essere inviati sul canale di connessione client.
SecurityUserData	Dati utente per uscita di sicurezza.
SendUserData	Dati utente per uscita di invio.
ReceiveUserData	Dati utente per uscita ricezione.
UserIdentifier	Identificativo utente da utilizzare per stabilire una sessione LU 6.2 .
Password	Password da utilizzare per stabilire una sessione LU 6.2 .
ConnectionName	Nome connessione.
HeartbeatInterval	Tempo in secondi tra i flussi di heartbeat.
StrucLength	Lunghezza della struttura MQCD.
ExitNameLength	Lunghezza dei nomi di uscita indirizzati da *SendExitPtr* e *ReceiveExitPtr*. Deve essere maggiore di zero se *SendExitPtr* o *ReceiveExitPtr* è impostato su un valore che non è il puntatore null.
ExitDataLength	Lunghezza dei dati di uscita indirizzati da *SendUserDataPtr* e *ReceiveUserDataPtr*. Deve essere maggiore di zero se *SendUserDataPtr* o *ReceiveUserDataPtr* è impostato su un valore che non è il puntatore null.
SendExitsDefined	Numero di uscite di invio indirizzate da *SendExitPtr*. Se zero, *SendExit* e *SendUserData* forniscono il nome e i dati di uscita. Se maggiore di zero, *SendExitPtr* e *SendUserDataPtr* forniscono i nomi e i dati di uscita e *SendExit* e *SendUserData* devono essere vuoti.
ReceiveExitsDefined	Numero di uscite di ricezione indirizzate da *ReceiveExitPtr*. Se zero, *ReceiveExit* e *ReceiveUserData* forniscono il nome e i dati di uscita. Se maggiore di zero, *ReceiveExitPtr* e *ReceiveUserDataPtr* forniscono i nomi e i dati di uscita e *ReceiveExit* e *ReceiveUserData* devono essere vuoti.
SendExitPtr	Indirizzo del nome della prima uscita di invio.
SendUserDataPtr	Indirizzo dei dati per la prima uscita di invio.
ReceiveExitPtr	Indirizzo del nome della prima uscita di ricezione.
ReceiveUserDataPtr	Indirizzo dei dati per la prima uscita di ricezione.
LongRemoteUserIdLength	Lunghezza dell'identificativo utente remoto lungo.
LongRemoteUserIdPtr	Indirizzo dell'identificativo utente remoto lungo.
RemoteSecurityId	Identificativo di sicurezza remoto.
SSLCipherSpec	CipherSpecTLS.
SSLPeerNamePtr	Indirizzo del nome peer TLS.
SSLPeerNameLength	Lunghezza del nome peer TLS.

Tabella 481. Campi in MQCD (Continua)

Campo in MQCD	Valore
KeepAliveInterval	Valore passato allo stack di comunicazioni per il tempo keepalive per il canale
LocalAddress	L'indirizzo di comunicazione locale, incluso l'indirizzo IP dell'adattatore di rete locale da utilizzare e un intervallo di porte da utilizzare per le connessioni in uscita.

Fornire la struttura di definizione del canale in uno dei modi seguenti:

- Utilizzando il campo offset *ClientConnOffset*

  In questo caso, l'applicazione deve dichiarare una struttura composta contenente un MQCNO seguito dalla struttura di definizione del canale MQCD e impostare *ClientConnOffset* sull'offset della struttura di definizione del canale dall'inizio di MQCNO. Verificare che questo offset sia corretto. *ClientConnPtr* deve essere impostato sul puntatore null o sui byte null.

  Utilizzare *ClientConnOffset* per i linguaggi di programmazione che non supportano il tipo di dati puntatore o che implementano il tipo di dati puntatore in un modo non portabile in ambienti differenti (ad esempio, il linguaggio di programmazione COBOL).

  Per il linguaggio di programmazione Visual Basic, una struttura composta denominataMQCNOCD viene fornito nel file di intestazione CMQXB.BAS; questa struttura contiene una struttura di MQCNO seguita da una struttura MQCD. Inizializza MQCNOCD richiamando la sottoroutine MQCNOCD_DEFAULTS. MQCNOCD viene utilizzato conMQCONNXQualsiasi variante della chiamata MQCONNX; per ulteriori informazioni, consultare la descrizione della chiamata MQCONNX.
- Utilizzando il campo puntatore *ClientConnPtr*

  In tal caso, l'applicazione può dichiarare la struttura di definizione del canale separatamente dalla struttura MQCNO e impostare *ClientConnPtr* sull'indirizzo della struttura di definizione del canale. Impostare *ClientConnOffset* su zero.

  Utilizzare *ClientConnPtr* per i linguaggi di programmazione che supportano il tipo di dati puntatore in un modo che sia portabile in ambienti differenti (ad esempio, il linguaggio di programmazione C).

  Nel linguaggio di programmazione C, è possibile utilizzare la variabile macro MQCD_CLIENT_CONN_DEFAULT per fornire i valori iniziali per la struttura più adatti per l'utilizzo nella chiamata MQCONNX rispetto ai valori iniziali forniti da MQCD_DEFAULT.

Indipendentemente dalla tecnica scelta, è possibile utilizzare solo una delle opzioni *ClientConnOffset* e *ClientConnPtr* ; la chiamata non riesce con codice motivo MQRC_CLIENT_CONN_ERROR se entrambi sono diversi da zero.

Una volta completata la chiamata MQCONNX, non si fa più riferimento alla struttura MQCD.

Questo campo viene ignorato se *Version* è minore di MQCNO_VERSION_2.

Nota: Sulle piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata, con il valore iniziale che è la stringa di byte null.

Multi *ConnTag (MQBYTE128) per MQCNO su Multiplatforms*

Una tag di connessione è concettualmente simile a un identificativo di connessione, ma potrebbe estendersi a più connessioni correlate, identificandole come una sola istanza dell'applicazione. Su Multiplatforms, la tag di collegamento viene generata dal gestore code al momento della connessione.

Per ulteriori informazioni, vedi identificativo di connessione e istanza dell'applicazione.

I tag di connessione generati sono `semi` leggibili. Ovvero, possono essere visualizzati e filtrati in MQSC come se fossero stringhe nella serie di caratteri locale. Le connessioni note a IBM MQ come

correlate vengono assegnate automaticamente alla stessa tag di connessione. Questa assegnazione è particolarmente importante per il bilanciamento dell'applicazione.

La tag di connessione generata è visibile in tre modi:

- Nella struttura MQCNO di output su una chiamata MQCONNX, quando viene specificato MQCNO_GENERATE_CONN_TAG .
- Nell'output da DISPLAY CONN (o equivalenti programmatici).
- Nell'output da DISPLAY APSTATUS (o equivalenti).

La tag cessa di essere valida quando l'applicazione termina o emette la chiamata MQDISC.

Riferimenti correlati

"ConnTag (MQBYTE128) for MQCNO on IBM MQ for z/OS" a pagina 337
A connection tag is conceptually similar to a connection identifier, but might span multiple related connections, identifying them as a single application instance. On IBM MQ for z/OS, the connection tag is an input field, provided by the application and used in conjunction with MQCNO_*_CONN_TAG options to serialize connections from that application instance

z/OS *ConnTag (MQBYTE128) for MQCNO on IBM MQ for z/OS*

A connection tag is conceptually similar to a connection identifier, but might span multiple related connections, identifying them as a single application instance. On IBM MQ for z/OS, the connection tag is an input field, provided by the application and used in conjunction with MQCNO_*_CONN_TAG options to serialize connections from that application instance

Where there are multiple instances of an application that are intended to be simultaneously connected, they must each supply a unique value for this field. See the descriptions of these connection tag options for further details.

Notes:

- On IBM MQ for z/OS, there is no way to administratively determine the connection tag associated with an application at runtime.
- Connection tag values beginning with MQ in upper, lower, or mixed case in either ASCII or EBCDIC are reserved for use by IBM products. Do not use connection tag values beginning with these letters.

Use the following special value if you require no tag:

MQCT_NONE
The value is binary zero for the length of the field.

For the C programming language, the constant MQCT_NONE_ARRAY is also defined; this constant has the same value as MQCT_NONE, but is an array of characters instead of a string.

The ConnTag field is used when connecting to a z/OS queue manager.

The length of this field is given by MQ_CONN_TAG_LENGTH. This field is ignored if *Version* is less than MQCNO_VERSION_3.

Multi See "ConnTag (MQBYTE128) per MQCNO su Multiplatforms" on page 336 for information on using the connection tag on IBM MQ for Multiplatforms.

SSLConfigPtr (PMQSCO) per MQCNO

SSLConfigPtr è un campo di immissione. Il suo valore iniziale è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e una stringa di byte all - null in caso contrario.

Utilizzare *SSLConfigPtr* e *SSLConfigOffset* solo quando l'applicazione che emette la chiamata MQCONNX è in esecuzione come IBM MQ MQI client e il protocollo di canale è TCP/IP. Se l'applicazione non è in esecuzione come client IBM MQ o se il protocollo del canale non è TCP/IP, *SSLConfigPtr* e *SSLConfigOffset* vengono ignorati.

Specificando *SSLConfigPtr* o *SSLConfigOffset*, più *ClientConnPtr* o *ClientConnOffset*, l'applicazione può controllare l'utilizzo di TLS per la connessione client. Quando le informazioni TLS

vengono specificate in questo modo, le variabili di ambiente MQSSLKEYR e MQSSLCRYP vengono ignorate; anche tutte le informazioni relative a TLS nella CCDT (client channel definition table) vengono ignorate.

Le informazioni TLS possono essere specificate solo su:

- La prima chiamata MQCONNX del processo client o
- Una chiamata MQCONNX successiva quando tutte le connessioni TLS precedenti al gestore code sono state concluse utilizzando MQDISC.

Questi sono gli unici stati in cui è possibile inizializzare l'ambiente TLS a livello di processo. Se viene emessa una chiamata MQCONNX specificando le informazioni TLS quando l'ambiente TLS già esiste, le informazioni TLS sulla chiamata vengono ignorate e la connessione viene effettuata utilizzando l'ambiente TLS esistente; la chiamata restituisce il codice di completamento MQCC_WARNING e il codice motivo MQRC_SSL_ALREADY_INITIALIZED in questo caso.

È possibile fornire la struttura MQSCO nello stesso modo della struttura MQCD, specificando un indirizzo in *SSLConfigPtr*o specificando un offset in *SSLConfigOffset* ; Per i dettagli su come eseguire questa operazione, consultare la descrizione di *ClientConnPtr* . Tuttavia, non è possibile utilizzare più di uno tra *SSLConfigPtr* e *SSLConfigOffset* ; la chiamata ha esito negativo con codice motivo MQRC_SSL_CONFIG_ERROR. se entrambi sono diversi da zero.

Una volta completata la chiamata MQCONNX, non si fa più riferimento alla struttura MQSCO.

Questo campo viene ignorato se *Version* è minore di MQCNO_VERSION_4.

Nota: Su piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata.

SSLConfigOffset (MQLONG) per MQCNO

SSLConfigOffset è l'offset, in byte, di una struttura MQSCO dall'inizio della struttura MQCNO. L'offset può essere positivo o negativo. Questo campo è un campo di input, con un valore iniziale di 0.

Utilizzare *SSLConfigOffset* solo quando l'applicazione che emette la chiamata MQCONNX è in esecuzione come IBM MQ MQI client. Per informazioni su come utilizzare questo campo, consultare la descrizione del campo *SSLConfigPtr* .

Questo campo viene ignorato se *Version* è minore di MQCNO_VERSION_4.

ConnectionId (MQBYTE24) per MQCNO

ConnectionId è un identificativo univoco a 24 byte che consente a IBM MQ di identificare in modo affidabile un'applicazione. Un'applicazione può utilizzare questo identificativo per la correlazione nelle chiamate PUT e GET. Questo parametro di emissione ha un valore iniziale di 24 byte null in tutti i linguaggi di programmazione.

Il gestore code assegna un ID univoco a tutte le connessioni, comunque siano stabilite. Se un MQCONNX stabilisce la connessione con MQCNO versione 5, l'applicazione può determinare l' ConnectionId dall'MQCNO restituito. L'identificativo assegnato è garantito come univoco tra tutti gli altri identificativi che IBM MQ genera, come CorrelId, MsgIDe GroupId.

Utilizzare ConnectionId per identificare le unità di lavoro di lunga durata utilizzando il comando PCF Interroga connessione o il comando MQSC DISPLAY CONN. L' ConnectionId utilizzato dai comandi MQSC (CONN) deriva dall' ConnectionId qui restituito. I comandi PCF Inquire e Arresta connessione possono utilizzare il ConnectionId qui restituito senza modifiche.

È possibile utilizzare ConnectionId per forzare la fine di un'unità di lavoro di lunga durata, specificando ConnectionId utilizzando il comando PCF Stop Connection o il comando MQSC STOP CONN. Consultare Arresta connessione e STOP CONN per ulteriori informazioni sull'utilizzo di questi comandi.

Questo campo non viene restituito se la versione è inferiore a MQCNO_VERSION_5.

La lunghezza di questo campo è fornita da MQ_CONNECTION_ID_LENGTH.

Offset SecurityParms(MQLONG) per MQCNO
SecurityParmsOffset è l'offset, in byte, della struttura MQCSP dall'inizio della struttura MQCNO. L'offset può essere positivo o negativo. Questo campo è un campo di input, con un valore iniziale di 0.

Questo campo viene ignorato se *Versione* è inferiore a MQCNO_VERSION_5.

La struttura MQCSP è definita in "MQCSP - Parametri di sicurezza" a pagina 340.

SecurityParmsPtr (PMQCSP) per MQCNO
SecurityParmsPtr è l'indirizzo della struttura MQCSP, utilizzato per specificare un ID utente e password per l'autenticazione da parte del servizio di autorizzazione. Questo campo è un campo di immissione e il suo valore iniziale è un puntatore null o byte null.

Questo campo viene ignorato se *Versione* è inferiore a MQCNO_VERSION_5.

La struttura MQCSP è definita in "MQCSP - Parametri di sicurezza" a pagina 340.

Riservato (MQBYTE4) per MQCNO
Un campo riservato per inserire la struttura in un limite a 64 bit. Il valore iniziale del campo è zero binario per la lunghezza del campo.

Questo campo viene ignorato se `Version` è minore di MQCNO_VERSION_6.

CCDTUrlLength (MQLONG) per MQCNO
CCDTUrlLength è la lunghezza della stringa identificata da CCDTUrlPtr o CCDTUrlOffset che contiene un URL che identifica l'ubicazione della tabella del canale di connessione client da utilizzare per la connessione. Il valore iniziale del campo è zero.

Utilizzare `CCDTUrlLength` solo quando l'applicazione che emette la chiamata MQCONNX è in esecuzione come IBM MQ MQI client.

Questa è un'alternativa programmatica all'impostazione delle variabili di ambiente MQCHLLIB e MQCHLTAB .

Se l'applicazione non è in esecuzione come client, `CCDTUrlLength` viene ignorato.

Questo campo viene ignorato se `Version` è minore di MQCNO_VERSION_6.

CCDTUrlPtr (PMQCHAR) per MQCNO
CCDTUrlPtr è un puntatore facoltativo a una stringa che contiene un URL per identificare l'ubicazione della tabella del canale di connessione client da utilizzare per la connessione. Questo campo è un campo di input, con un valore iniziale di un puntatore nullo nei linguaggi di programmazione che supportano i puntatori e una stringa di byte completamente nulla in caso contrario.

Utilizzare `CCDTUrlPtr` solo quando l'applicazione che emette la chiamata MQCONNX è in esecuzione come IBM MQ MQI client.

Importante: È possibile utilizzare solo uno tra `CCDTUrlPtr` e `CCDTUrlOffset`. La chiamata ha esito negativo con codice motivo MQRC_CCDT_URL_ERROR se entrambi i campi sono diversi da zero.

Questa è un'alternativa programmatica all'impostazione delle variabili di ambiente MQCHLLIB e MQCHLTAB .

Se l'applicazione non è in esecuzione come client, `CCDTUrlPtr` viene ignorato.

Questo campo viene ignorato se `Version` è minore di MQCNO_VERSION_6.

CCDTUrlOffset (MQLONG) per MQCNO
CCDTUrlOffset è l'offset in byte, dall'inizio della struttura MQCNO a una stringa che contiene un URL che identifica l'ubicazione della tabella del canale di connessione client da utilizzare per la connessione. L'offset può essere positivo o negativo e il valore iniziale del campo è zero.

Utilizzare `CCDTUrlOffset` solo quando l'applicazione che emette la chiamata MQCONNX è in esecuzione come IBM MQ MQI client.

Importante: È possibile utilizzare solo uno tra `CCDTUrlPtr` e `CCDTUrlOffset`. La chiamata ha esito negativo con codice motivo MQRC_CCDT_URL_ERROR se entrambi i campi sono diversi da zero.

Questa è un'alternativa programmatica all'impostazione delle variabili di ambiente MQCHLLIB e MQCHLTAB .

Se l'applicazione non è in esecuzione come client, `CCDTUrlOffset` viene ignorato.

Questo campo viene ignorato se `Version` è minore di MQCNO_VERSION_6.

ApplName (MQCHAR28) per MQCNO

Il nome impostato dall'applicazione per identificare il collegamento al gestore code. Il valore iniziale del campo è MQAN_NONE_ARRAY (caratteri vuoti).

Questo campo viene ignorato se `Version` è minore di MQCNO_VERSION_7o se il valore è impostato su spazi vuoti.

z/OS Non è possibile impostare questo campo su z/OS. Se si tenta di eseguire questa operazione, si riceve di nuovo il codice motivo MQRC_CNO_ERROR.

Reserved2 (MQBYTE4) per MQCNO

Un campo riservato per inserire la struttura in un limite a 64 bit. Il valore iniziale del campo è zero binario per la lunghezza del campo.

Questo campo viene ignorato se `Version` è minore di MQCNO_VERSION_7.

Offset BalanceParms(MQLONG) per MQCNO

L'ubicazione della memoria per una struttura di tipo MQBNO che contiene informazioni sul comportamento di bilanciamento dell'applicazione. La struttura viene ignorata completamente a meno che l'applicazione non si stia collegando su un canale client.

Questo campo viene ignorato se `Version` è minore di MQCNO_VERSION_8.

Per ulteriori informazioni, consultare MQBNO .

Se si fornisce questo campo, non è possibile fornire il campo “BalanceParmsPtr (MQPTR) per MQCNO” a pagina 340 . Se si tenta di fornire entrambi i campi, si riceve un messaggio MQRC_CNO_ERROR. Poiché questo campo è rilevante solo per le connessioni client, fornendo questo campo su qualsiasi altro tipo di connessione si ottiene anche MQRC_CNO_ERROR.

BalanceParmsPtr (MQPTR) per MQCNO

Puntatore all'ubicazione della memoria per una struttura di tipo MQBNO che contiene informazioni sul comportamento di bilanciamento dell'applicazione. La struttura viene ignorata completamente a meno che l'applicazione non si stia collegando su un canale client.

Questo campo viene ignorato se `Version` è minore di MQCNO_VERSION_8.

Per ulteriori informazioni, consultare MQBNO .

Se si fornisce questo campo, non è possibile fornire il campo “Offset BalanceParms(MQLONG) per MQCNO” a pagina 340 . Se si tenta di fornire entrambi i campi, si riceve un messaggio MQRC_CNO_ERROR. Poiché questo campo è rilevante solo per le connessioni client, fornendo questo campo su qualsiasi altro tipo di connessione si ottiene anche MQRC_CNO_ERROR.

MQCSP - Parametri di sicurezza

La struttura del parametro di sicurezza della connessione IBM MQ viene utilizzata dalle applicazioni per passare le informazioni di autenticazione su una chiamata MQCONNX al gestore code. È inoltre possibile utilizzarlo per fornire la chiave iniziale utilizzata con il sistema di protezione password IBM MQ che codifica i dati sensibili.

Impostare *AuthenticationType* su MQCSP_AUTH_USER_ID_AND_PWD per includere l'ID utente e la password dalla versione 1.

Quando si forniscono le informazioni sulla chiave iniziale alla versione 2, *AuthenticationType* assume il valore predefinito MQCSP_AUTH_NONE.

V 9.4.0 Da IBM MQ 9.3.4, utilizzare *AuthenticationType* per includere le informazioni sul token di autenticazione.

V 9.4.0 È possibile utilizzare MQCSP_AUTH_USER_ID_AND_PWD o MQCSP_AUTH_ID_TOKEN, ma non entrambi.

Avviso: In alcuni casi, la parola d'ordine o token di autenticazione in una struttura MQCSP per un'applicazione client viene inviata sulla rete in testo semplice. Per assicurarsi che le password delle applicazioni client e i token di autenticazione siano protetti in maniera appropriata, consultare MQCSP password protection.

Disponibilità

La struttura MQCSP è disponibile su tutte le piattaforme IBM MQ supportate.

Versione

I file di intestazione, COPY e INCLUDE forniti per i linguaggi di programmazione supportati contengono la versione più recente di MQCSP, ma con il valore iniziale del campo *Version* impostato su MQCSP_VERSION_1. Per utilizzare i campi che non sono presenti nella struttura version-1 , l'applicazione deve impostare il campo *Version* sul numero di versione richiesto.

Serie di caratteri e codifica

I dati in MQCSP devono essere nella serie di caratteri e nella codificazione del gestore code locale, forniti rispettivamente dall'attributo del gestore code **CodedCharSetId** e da MQENC_NATIVE.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 482. Campi in MQCSP per MQCSP

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQCSP	'CSP¬'
Versione (numero versione struttura)	MQCSP_VERSION_1	1
AuthenticationType (tipo di autenticazione)	Nessuna	MQCSP_AUTH_NONE
Reserved1 (richiesto per l'allineamento del puntatore su IBM i)	Nessuna	Stringa null o spazi vuoti
CSPUserIdPtr (indirizzo dell'ID utente)	Nessuna	Puntatore null o byte null
CSPUserIdCSPUserId (offset dell'ID utente)	Nessuna	0
CSPUserIdLunghezza (lunghezza dell'ID utente)	Nessuna	0
Reserved2 (richiesto per l'allineamento del puntatore su IBM i)	Nessuna	Stringa null o spazi vuoti

Tabella 482. Campi in MQCSP per MQCSP (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
CSPPasswordPtr (indirizzo della parola d'ordine)	Nessuna	Puntatore null o byte null
CSPPasswordOffset (offset della password)	Nessuna	0
CSPPasswordLength (lunghezza della parola d'ordine)	Nessuna	0
Nota: I restanti campi vengono ignorati se *Version* è minore di MQCSP_VERSION_2.		
Reserved3 (richiesto per l'allineamento del puntatore su IBM i)	Nessuna	Stringa null o spazi vuoti
InitialKeyPtr	Nessuna	Puntatore null o spazi vuoti
InitialKeyInitialKey (offset della chiave iniziale per il sistema di protezione della parola d'ordine)	Nessuna	0
InitialKeyLunghezza (lunghezza della chiave iniziale per il sistema di protezione della parola d'ordine)	Nessuna	0
Nota: I restanti campi vengono ignorati se *Version* è minore di MQCSP_VERSION_3.		
V 9.4.0 Reserved4 (richiesto per l'allineamento del puntatore su IBM i)	Nessuna	Stringa null o spazi vuoti
V 9.4.0 TokenPtr (indirizzo del token di autenticazione)	Nessuna	Puntatore null o spazi vuoti
V 9.4.0 TokenKeyTokenKey (offset del token di autenticazione)	Nessuna	0
V 9.4.0 TokenLength (lunghezza token di autenticazione)	Nessuna	0
Note: 1. Il simbolo ¬ rappresenta un singolo carattere vuoto. 2. Nel linguaggio di programmazione C, la variabile macroMQCSP_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura: `MQCSP MyCSP = {MQCSP_DEFAULT};`		

Dichiarazioni di lingua

Dichiarazione C per MQCSP

```
typedef struct tagMQCSP MQCSP;
struct tagMQCSP {
  MQCHAR4    StrucId;            /* Structure identifier */
  MQLONG     Version;            /* Structure version number */
```

```
  MQLONG     AuthenticationType; /* Type of authentication */
  MQBYTE4    Reserved1;          /* Required for IBM i pointer alignment */
  MQPTR      CSPUserIdPtr;       /* Address of user ID */
  MQLONG     CSPUserIdOffset;    /* Offset of user ID */
  MQLONG     CSPUserIdLength;    /* Length of user ID */
  MQBYTE8    Reserved2;          /* Required for IBM i pointer alignment */
  MQPTR      CSPPasswordPtr;     /* Address of password */
  MQLONG     CSPPasswordOffset;  /* Offset of password */
  MQLONG     CSPPasswordLength;  /* Length of password */
/* Ver:1 */

  MQBYTE8    Reserved3;          /* Required for IBM i pointer alignment */
  MQPTR      InitialKeyPtr;      /* Address of initial key */
  MQLONG     InitialKeyOffset;   /* Offset of initial key */
  MQLONG     InitialKeyLength;   /* Length of initial key */
/* Ver:2 */
  };
```

V 9.4.0

```
typedef struct tagMQCSP MQCSP;
struct tagMQCSP {
  MQCHAR4    StrucId;            /* Structure identifier */
  MQLONG     Version;            /* Structure version number */
  MQLONG     AuthenticationType; /* Type of authentication */
  MQBYTE4    Reserved1;          /* Required for IBM i pointer alignment */
  MQPTR      CSPUserIdPtr;       /* Address of user ID */
  MQLONG     CSPUserIdOffset;    /* Offset of user ID */
  MQLONG     CSPUserIdLength;    /* Length of user ID */
  MQBYTE8    Reserved2;          /* Required for IBM i pointer alignment */
  MQPTR      CSPPasswordPtr;     /* Address of password */
  MQLONG     CSPPasswordOffset;  /* Offset of password */
  MQLONG     CSPPasswordLength;  /* Length of password */
/* Ver:1 */

  MQBYTE8    Reserved3;          /* Required for IBM i pointer alignment */
  MQPTR      InitialKeyPtr;      /* Address of initial key */
  MQLONG     InitialKeyOffset;   /* Offset of initial key */
  MQLONG     InitialKeyLength;   /* Length of initial key */
/* Ver:2 */

  MQBYTE8    Reserved4;          /* Required for IBM i pointer alignment */
  MQPTR      TokenPtr;           /* Address of token */
  MQLONG     TokenOffset;        /* Offset of token */
  MQLONG     TokenLength;        /* Length of token */
/* Ver:3 */
  };
```

Dichiarazione COBOL per MQCSP

```
**   MQCSP structure
  10 MQCSP.
**    Structure identifier
   15 MQCSP-STRUCID           PIC X(4).
**    Structure version number
   15 MQCSP-VERSION           PIC S9(9) BINARY.
**    Type of authentication
   15 MQCSP-AUTHENTICATIONTYPE PIC S9(9) BINARY.
**    Required for IBM i pointer alignment
   15 MQCSP-RESERVED1         PIC X(4).
**    Address of user ID
   15 MQCSP-CSPUSERIDPTR      POINTER.
**    Offset of user ID
   15 MQCSP-CSPUSERIDOFFSET   PIC S9(9) BINARY.
**    Length of user ID
   15 MQCSP-CSPUSERIDLENGTH   PIC S9(9) BINARY.
**    Required for IBM i pointer alignment
   15 MQCSP-RESERVED2         PIC X(4).
**    Address of password
   15 MQCSP-CSPPASSWORDPTR    POINTER.
**    Offset of password
   15 MQCSP-CSPPASSWORDOFFSET PIC S9(9) BINARY.
**    Length of password
   15 MQCSP-CSPPASSWORDLENGTH PIC S9(9) BINARY.
** Ver:1 **

**    Reserved
   15 MQCSP-RESERVED3         PIC X(8).
**    Address of initial key
```

```
   15 MQCSP-INITIALKEYPTR     POINTER.
**    Offset of initial key
   15 MQCSP-INITIALKEYOFFSET  PIC S9(9) BINARY.
**    Length of initial key
   15 MQCSP-INITIALKEYLENGTH  PIC S9(9) BINARY.
** Ver:2 **
```

V 9.4.0

```
**   MQCSP structure
  10 MQCSP.
**    Structure identifier
   15 MQCSP-STRUCID           PIC X(4).
**    Structure version number
   15 MQCSP-VERSION           PIC S9(9) BINARY.
**    Type of authentication
   15 MQCSP-AUTHENTICATIONTYPE PIC S9(9) BINARY.
**    Required for IBM i pointer alignment
   15 MQCSP-RESERVED1         PIC X(4).
**    Address of user ID
   15 MQCSP-CSPUSERIDPTR      POINTER.
**    Offset of user ID
   15 MQCSP-CSPUSERIDOFFSET   PIC S9(9) BINARY.
**    Length of user ID
   15 MQCSP-CSPUSERIDLENGTH   PIC S9(9) BINARY.
**    Required for IBM i pointer alignment
   15 MQCSP-RESERVED2         PIC X(4).
**    Address of password
   15 MQCSP-CSPPASSWORDPTR    POINTER.
**    Offset of password
   15 MQCSP-CSPPASSWORDOFFSET PIC S9(9) BINARY.
**    Length of password
   15 MQCSP-CSPPASSWORDLENGTH PIC S9(9) BINARY.
** Ver:1 **

**    Reserved
   15 MQCSP-RESERVED3         PIC X(8).
**    Address of initial key
   15 MQCSP-INITIALKEYPTR     POINTER.
**    Offset of initial key
   15 MQCSP-INITIALKEYOFFSET  PIC S9(9) BINARY.
**    Length of initial key
   15 MQCSP-INITIALKEYLENGTH  PIC S9(9) BINARY.
** Ver:2 **

**    Reserved
   15 MQCSP-RESERVED4         PIC X(8).
**    Address of token
   15 MQCSP-TOKENPTR     POINTER.
**    Offset of token
   15 MQCSP-TOKENOFFSET  PIC S9(9) BINARY.
**    Length of token
   15 MQCSP-TOKENLENGTH  PIC S9(9) BINARY.
** Ver:3 **
```

Dichiarazione PL/I per MQCSP

```
dcl
 1 MQCSP based,
   3 StrucId           char(4),       /* Structure identifier */
   3 Version           fixed bin(31), /* Structure version number */
   3 AuthenticationType fixed bin(31),  /* Type of authentication */
   3 Reserved1         char(4),       /* Required for IBM i pointer
                                         alignment */
   3 CSPUserIdPtr      pointer,       /* Address of user ID */
   3 CSPUserIdOffset   fixed bin(31), /* Offset of user ID */
   3 CSPUserIdLength   fixed bin(31), /* Length of user ID */
   3 Reserved2         char(8),       /* Required for IBM i pointer
                                         alignment */
   3 CSPPasswordPtr    pointer,       /* Address of password */
   3 CSPPasswordOffset fixed bin(31), /* Offset of user ID */
   3 CSPPasswordLength fixed bin(31), /* Length of user ID */
/* Version 1 */

   3 Reserved3         char(8),       /* Reserved */
   3 InitialKeyPtr     pointer,       /* Address of initial key */
   3 InitialKeyOffset  fixed bin(31), /* Offset of initial key */
```

```
  3 InitialKeyLength  fixed bin(31); /* Length of initial key */
/* Version 2 */
```

V 9.4.0

```
dcl
 1 MQCSP based,
  3 StrucId           char(4),       /* Structure identifier */
  3 Version           fixed bin(31), /* Structure version number */
  3 AuthenticationType fixed bin(31),  /* Type of authentication */
  3 Reserved1         char(4),       /* Required for IBM i pointer
                                        alignment */
  3 CSPUserIdPtr      pointer,       /* Address of user ID */
  3 CSPUserIdOffset   fixed bin(31), /* Offset of user ID */
  3 CSPUserIdLength   fixed bin(31), /* Length of user ID */
  3 Reserved2         char(8),       /* Required for IBM i pointer
                                        alignment */
  3 CSPPasswordPtr    pointer,       /* Address of password */
  3 CSPPasswordOffset fixed bin(31), /* Offset of user ID */
  3 CSPPasswordLength fixed bin(31), /* Length of user ID */
/* Version 1 */

  3 Reserved3         char(8),       /* Reserved */
  3 InitialKeyPtr     pointer,       /* Address of initial key */
  3 InitialKeyOffset  fixed bin(31), /* Offset of initial key */
  3 InitialKeyLength  fixed bin(31); /* Length of initial key */
/* Version 2 */

  3 Reserved4         char(8),       /* Reserved */
  3 TokenPtr          pointer,       /* Address of Token */
  3 TokenOffset       fixed bin(31), /* Offset of Token */
  3 TokenLength       fixed bin(31); /* Length of Token */
/* Version 3 */
```

Dichiarazione di Visual Basic per MQCSP

```
Type MQCSP
  StrucId           As String*4   'Structure identifier'
  Version           As Long       'Structure version number'
  AuthenticationType As Long      'Type of authentication'
  Reserved1         As MQBYTE4    'Required for IBM i pointer'
                                  'alignment'
  CSPUserIdPtr      As MQPTR      'Address of user ID'
  CSPUserIdOffset   As Long       'Offset of user ID'
  CSPUserIdLength   As Long       'Length of user ID'
  Reserved2         As MQBYTE8    'Required for IBM i pointer'
                                  'alignment'
  CSPPasswordPtr    As MQPTR      'Address of password'
  CSPPasswordOffset As Long       'Offset of password'
  CSPPasswordLength As Long       'Length of password'
End Type
```

Concetti correlati

Utilizzo dei token di autenticazione

StrucId (MQCHAR4) per MQCSP

È l'identificativo della struttura dei parametri di sicurezza. È sempre un campo di immissione. Il suo valore è MQCSP_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQCSP
Identificativo per la struttura dei parametri di sicurezza.

Per il linguaggio di programmazione C, viene definita anche la costante MQCSP_STRUC_ID_ARRAY. Ha lo stesso valore di MQCSP_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQCSP

Numero di versione della struttura MQCSP.

Il valore deve essere:

MQCSP_VERSION_1
Struttura dei parametri di sicurezza Version-1 . Alla versione 1, è possibile includere un ID utente e una password nella struttura MQCSP per l'autenticazione con il gestore code.

MQCSP_VERSION_2
Struttura dei parametri di sicurezza Version-2 . Alla versione 2 è possibile includere un ID utente e una password per l'autenticazione con il gestore code e specificare la chiave iniziale utilizzata per proteggere le password.

V 9.4.0 **MQCSP_VERSION_3**
Struttura dei parametri di sicurezza Version-3 . Alla versione 3, è possibile includere un ID utente e una password o un token di autenticazione nella struttura MQCSP per l'autenticazione con il gestore code. È anche possibile specificare la chiave iniziale utilizzata per proteggere le parole d'ordine.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQCSP_CURRENT_
Versione corrente della struttura dei parametri di sicurezza.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQCSP_VERSION_3.

AuthenticationType (MQLONG) per MQCSP

AuthenticationType è un campo di input. Il valore iniziale è MQCSP_AUTH_NONE.

Questo è il tipo di autenticazione da eseguire. I valori validi sono:

MQCSP_AUTH_NONE
Non utilizzare i campi ID utente e password o token di autenticazione .

MQCSP_AUTH_USER_ID_AND_PWD
Eseguire l'autenticazione utilizzando l'ID utente e password nella struttura MQCSP.

V 9.4.0 **ID_AUTORE MQCSP_TOKEN**
Eseguire l'autenticazione utilizzando il token di autenticazione nella struttura MQCSP.

Il valore predefinito è MQCSP_AUTH_NONE. Con l'impostazione predefinita, non viene eseguita alcuna protezione con password.

Se si richiede l'autenticazione, è necessario impostare **MQCSP**.Da **AuthenticationType** a MQCSP_AUTH_USER_ID_AND_PWD o MQCSP_AUTH_ID_TOKEN.

Per ulteriori informazioni, consultare MQCSP password protection.

Concetti correlati
Utilizzo dei token di autenticazione

Reserved1 (MQBYTE4) per MQCSP

Un campo riservato, richiesto per l'allineamento del puntatore su IBM i.

Il valore iniziale di questo campo è null.

CSPUserIdPtr (MQPTR) per MQCSP

L'indirizzo per l'ID utente da utilizzare nell'autenticazione.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e, in caso contrario, una stringa di byte completamente null. Questo campo viene ignorato se *Version* è minore di MQCNO_VERSION_5.

Questo campo può contenere un ID utente del sistema operativo quando un **AUTHTYPE** di *IDPWOS* viene denominato nel campo CONNAUTH del gestore code.

Su Windows , questo può essere un ID utente del dominio completo.

Questo campo può contenere un ID utente LDAP quando un **AUTHTYPE** di *IDPWLDAP* viene denominato nel campo CONNAUTH del gestore code.

Offset CSPUserId(MQLONG) per MQCSP
L'offset in byte per l'ID utente da utilizzare nell'autenticazione. L'offset può essere positivo o negativo.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

CSPUserIdLunghezza (MQLONG) per MQCSP
La lunghezza dell'ID utente da utilizzare nell'autenticazione.

la lunghezza massima dell'ID utente dipende dalla piattaforma; consultare ID utente. Se la lunghezza dell'ID utente è maggiore della lunghezza massima consentita, la richiesta di autenticazione non riesce con MQRC_CSP_ERROR. Nelle versioni precedenti di IBM MQ, l'errore restituito era MQRC_NoT_AUTHORIZED.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

Reserved2 (MQBYTE8) per MQCSP
Un campo riservato, richiesto per l'allineamento del puntatore su IBM i.

Il valore iniziale di questo campo è null.

CSPPasswordPtr (MQPTR) per MQCSP
L'indirizzo per la password da utilizzare nell'autenticazione.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e, in caso contrario, una stringa di byte completamente null. Questo campo viene ignorato se *Version* è minore di MQCNO_VERSION_5.

Questo campo può contenere una password vuota che viene rifiutata dal sistema operativo o dal controllo della password LDAP, a seconda dell'impostazione, ma non viene rifiutata da IBM MQ prima che venga passata al metodo di autenticazione.

CSPPasswordOffset (MQLONG) per MQCSP
Questo è l'offset in byte per la password da utilizzare nell'autenticazione. L'offset può essere positivo o negativo.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

CSPPasswordLength (MQLONG) per MQCSP
La lunghezza della password da utilizzare nell'autenticazione.

La lunghezza massima della password è MQ_CSP_PASSWORD_LENGTH, che è di 256 caratteri. Se la lunghezza della password è maggiore della lunghezza massima consentita, la richiesta di autenticazione non riesce con MQRC_CSP_ERROR. Nelle versioni precedenti di IBM MQ, l'errore restituito era MQRC_NoT_AUTHORIZED.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

Reserved3 (MQBYTE8) per MQCSP
Un campo riservato, richiesto per l'allineamento del puntatore su IBM i.

Il valore iniziale di questo campo è null.

Multi *InitialKeyPtr (MQPTR) per MQCSP*
L'indirizzo per la chiave iniziale per il sistema di protezione password.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e, in caso contrario, una stringa di byte completamente null. Questo campo viene ignorato se *Version* è inferiore a MQCSP_VERSION_2.

Questo campo è rilevante solo per IBM MQ MQI clients in esecuzione su IBM i, AIX, Linux, and Windows sistemi.

IBM MQ MQI clients può fornire valori codificati per alcuni campi, utilizzando il sistema di protezione con password IBM MQ . Se è stata utilizzata una chiave iniziale per codificare la password per il repository chiavi specificato nella struttura MQCSO, assicurarsi di includere i campi chiave iniziali in MQCSP per la stessa applicazione client.

I client IBM MQ MQI possono fornire valori codificati per alcuni campi, utilizzando il sistema di protezione con password IBM MQ . Se è stata utilizzata una chiave iniziale per codificare la parola d'ordine per il repository chiavi specificato nella struttura MQCSO, assicurarsi di includere i campi chiave iniziali nella struttura MQCSP per la stessa applicazione client.

Una chiave iniziale viene utilizzata dall'algoritmo di codifica per codificare e decodificare questi valori. Se viene fornita una chiave iniziale quando i valori di questi campi vengono codificati utilizzando il programma di utilità **runmqicred** , la stessa chiave iniziale deve essere specificata dal client quando si connette al gestore code.

La chiave iniziale specificata utilizzando questo campo sovrascrive qualsiasi chiave iniziale specificata utilizzando la variabile di ambiente *MQS_MQI_KEYFILE* o la proprietà *MQIInitialKeyFile* nella stanza di sicurezza del file di configurazione del client.

È possibile utilizzare *InitialKeyOffset* o *InitialKeyPtr* per specificare la chiave iniziale, ma non entrambe.

Attività correlate

Fornitura di una chiave iniziale per il client IBM MQ MQI su AIX, Linux e Windows

Protezione delle password nei file di configurazione del componente IBM MQ

Riferimenti correlati

runmqicred (proteggi password client IBM MQ)

“KeyRepoPasswordPtr (MQPTR) per MQSCO” a pagina 581
Questo è l'indirizzo in byte della passphrase del repository chiavi TLS.

“Offset InitialKey(MQLONG) per MQCSP” a pagina 348
L'offset in byte per la chiave iniziale per il sistema di protezione password dall'inizio della struttura MQCSP. L'offset può essere positivo o negativo.

Multi *Offset InitialKey(MQLONG) per MQCSP*

L'offset in byte per la chiave iniziale per il sistema di protezione password dall'inizio della struttura MQCSP. L'offset può essere positivo o negativo.

È possibile utilizzare *InitialKeyOffset* o *InitialKeyPtr* per specificare la chiave iniziale, ma non entrambe. Per ulteriori informazioni, consultare la descrizione del campo *InitialKeyPtr* .

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è inferiore a MQCSP_VERSION_2.

Attività correlate

Protezione delle password nei file di configurazione del componente IBM MQ

Fornitura di una chiave iniziale per il client IBM MQ MQI su AIX, Linux e Windows

Riferimenti correlati

“InitialKeyPtr (MQPTR) per MQCSP” a pagina 347
L'indirizzo per la chiave iniziale per il sistema di protezione password.

Multi *Lunghezza InitialKey(MQLONG) per MQCSP*

La lunghezza della chiave iniziale per il sistema di protezione password.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è inferiore a MQCSP_VERSION_2.

Attività correlate

Protezione delle password nei file di configurazione del componente IBM MQ

Fornitura di una chiave iniziale per il client IBM MQ MQI su AIX, Linux e Windows

V 9.4.0 *Reserved4 (MQBYTE8) per MQCSP*

Un campo riservato, richiesto per l'allineamento del puntatore su IBM i.

Il valore iniziale di questo campo è null.

V 9.4.0 *TokenPtr (MQPTR) per MQCSP*

L'indirizzo del token di autenticazione utilizzato per l'autenticazione con il gestore code.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e una stringa di byte all - null in caso contrario. Questo campo viene ignorato se *Version* è minore di MQCSP_VERSION_3.

Questo campo è rilevante per IBM MQ MQI clients la connessione ai gestori code IBM MQ in esecuzione su sistemi AIX o Linux .

È possibile utilizzare *TokenOffset* o *TokenPtr* per specificare il token di autenticazione, ma non entrambi.

Per ulteriori informazioni, vedi Utilizzo dei token di autenticazione in un'applicazione.

Concetti correlati

Utilizzo dei token di autenticazione

Riferimenti correlati

"TokenOffset (MQLONG) per MQCSP" a pagina 349

Questo è l'offset in byte per il token di autenticazione dall'inizio della struttura MQCSP. L'offset può essere positivo o negativo.

V 9.4.0 *TokenOffset (MQLONG) per MQCSP*

Questo è l'offset in byte per il token di autenticazione dall'inizio della struttura MQCSP. L'offset può essere positivo o negativo.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è minore di MQCSP_VERSION_3.

È possibile utilizzare *TokenOffset* o *TokenPtr* per specificare il token, ma non entrambi. Per ulteriori informazioni, consultare la descrizione del campo *TokenPtr* .

Concetti correlati

Utilizzo dei token di autenticazione

Attività correlate

Utilizzo dei token di autenticazione in una applicazione

Riferimenti correlati

"TokenPtr (MQPTR) per MQCSP" a pagina 349

L'indirizzo del token di autenticazione utilizzato per l'autenticazione con il gestore code.

V 9.4.0 *TokenLength (MQLONG) per MQCSP*

Questa è la lunghezza del token di autenticazione utilizzato per l'autenticazione con il gestore code.

La lunghezza massima del token di autenticazione è MQ_CSP_TOKEN_LENGTH, che è 8192 byte. Se *TokenLength* è maggiore della lunghezza massima consentita, la richiesta di autenticazione ha esito negativo con MQRC_CSP_ERROR.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è minore di MQCSP_VERSION_3.

Per ulteriori informazioni, vedi Utilizzo dei token di autenticazione in un'applicazione.

Concetti correlati
Utilizzo dei token di autenticazione

MQCTLO - Struttura delle opzioni di callback di controllo

La struttura MQCTLO viene utilizzata per specificare le opzioni relative a una funzione di callback di controllo. La struttura è un parametro di input e output nella chiamata MQCTL.

Disponibilità

La struttura MQCTLO è disponibile sulle piattaforme seguenti:

- AIX
- IBM i
- Linux
- Windows
- z/OS

e per IBM MQ MQI clients collegati a questi sistemi.

Versione

La versione corrente di MQCTLO è MQCTLO_VERSION_1.

Serie di caratteri e codifica

I dati in MQCTLO devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da MQENC_NATIVE. Tuttavia, se l'applicazione è in esecuzione come client MQ MQI, la struttura deve essere nella serie di caratteri e nella codifica del client.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 483. Campi in MQCTLO

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucID (identificativo della struttura)	ID_STRUC_MQCTLO_	'CTLO'
Versione (numero versione struttura)	MQCTLO_VERSION_1	1
Opzioni (opzioni)	MQCTLO_NONE	Valori null
Opzioni (campo riservato)	Campo riservato	
ConnectionArea (campo per la funzione di callback da utilizzare)	Nessuna	Puntatore null o byte null

Tabella 483. Campi in MQCTLO (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**

Note:

1. Nel linguaggio di programmazione C, la variabile macroMQCTLO_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQCTLO MyCTLO = {MQCTLO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQCTLO

```
typedef struct tagMQCTLO MQCTLO;
 struct tagMQCTLO {
  MQCHAR4   StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQLONG    Options;        /* Options that control the action of MQCTL */
  MQLONG    Reserved;       /* Reserved field */


MQPTR     ConnectionArea; /* Connection work area passed to the function */
 };
```

Dichiarazione COBOL per MQCTLO

```
** MQCTLO structure
 10  MQCTLO.
** Structure Identifier
 15  MQCTLO-STRUCID                    PIC X(4).
** Structure Version
 15  MQCTLO-VERSION                    PIC S9(9) BINARY.
 ** Options
 15  MQCTLO-OPTIONS                    PIC S9(9) BINARY.
** Reserved
 15  MQCTLO-RESERVED                   PIC S9(9) BINARY.
** ConnectionArea
 15  MQCTLO-CONNECTIONAREA             POINTER
```

Dichiarazione PL/I per MQCTLO

```
dcl
 1 MQCTLO based,
  3 StrucId             char(4),       /* Structure identifier */
  3 Version             fixed bin(31), /* Structure version */
  3 Options             fixed bin(31), /* Options */
  3 Reserved            fixed bin(31),
  3 ConnectionArea      pointer;       /* Connection work area */
```

StrucId (MQCHAR4) per MQCTLO

È l'identificativo della struttura delle opzioni di controllo. È sempre un campo di immissione. Il valore è MQCTLO_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQCTLO_
Identificativo per la struttura delle opzioni di controllo.

Per il linguaggio di programmazione C, viene definita anche la costante MQCTLO_STRUC_ID_ARRAY. Ha lo stesso valore di MQCTLO_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQCTLO

Struttura delle opzioni di controllo - Campo Versione

Questo è il numero di versione della struttura; il valore deve essere:

MQCTLO_VERSION_1
Version-1 Struttura delle opzioni di controllo.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQCTLO_CURRENT_
La versione corrente della struttura delle opzioni di controllo.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQCTLO_VERSION_1.

Opzioni (MQLONG) per MQCTLO

Struttura delle opzioni di controllo - Campo Opzioni

Opzioni che controllano l'azione di MQCTL.

MQCTLO_FAIL_IF_QUIESCING
Forzare la chiamata MQCTL in modo che abbia esito negativo se il gestore code o la connessione sono in stato di inattività.

Specificare MQGMO_FAIL_IF_QUIESCING, nelle opzioni MQGMO trasmesse sulla chiamata MQCB, per causare la notifica ai consumatori di messaggi quando sono in fase di sospensione.

MQCTLO_THREAD_AFFINITY
Questa opzione informa il sistema che l'applicazione richiede che tutti i destinatari del messaggio, per lo stesso collegamento, vengano richiamati sullo stesso thread. Questo thread verrà utilizzato per tutte le chiamate dei consumer fino a quando la connessione non viene arrestata.

Opzione predefinita: se non è necessaria alcuna delle opzioni descritte, utilizzare la seguente opzione:

MQCTLO_NONE
Utilizzare questo valore per indicare che non sono state specificate altre opzioni; tutte le opzioni assumono i propri valori predefiniti. MQCTLO_NONE è definito per aiutare la documentazione del programma; non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Questo è un campo di immissione. Il valore iniziale del campo *Options* è MQCTLO_NONE.

Riservato (MQLONG) per MQCTLO

Questo è un campo riservato. Il valore deve essere zero.

ConnectionArea (MQPTR) per MQCTLO

Struttura delle opzioni di controllo - campo ConnectionArea

Questo è un campo disponibile per la funzione di callback da utilizzare.

Il gestore code non prende alcuna decisione in base al contenuto di questo campo e viene passato invariato al campo ConnectionArea della struttura MQCBC, che è un parametro di input per il callback.

Questo campo viene ignorato per tutte le operazioni diverse da MQOP_START e MQOP_START_WAIT.

Questo è un campo di input e output per la funzione di callback. Il valore iniziale di questo campo è un puntatore null o byte null.

MQDH - Intestazione di distribuzione

La struttura MQDH descrive i dati aggiuntivi presenti in un messaggio quando tale messaggio è un messaggio dell'elenco di distribuzione memorizzato in una coda di trasmissione. Un messaggio

dell'elenco di distribuzione è un messaggio inviato a più code di destinazione. I dati aggiuntivi sono costituiti dalla struttura MQDH seguita da un array di record MQOR e da un array di record MQPMR.Questa struttura viene utilizzata da applicazioni specializzate che inserano i messaggi direttamente nelle code di trasmissione o che rimuovono i messaggi dalle code di trasmissione (ad esempio, gli agent del canale dei messaggi). Le applicazioni che desiderano inserire messaggi negli elenchi di distribuzione non devono utilizzare questa struttura. Al contrario, devono utilizzare la struttura MQOD per definire le destinazioni nell'elenco di distribuzione e la struttura MQPMO per specificare le proprietà del messaggio o ricevere informazioni sui messaggi inviati alle singole destinazioni.

Disponibilità

La struttura MQDH è disponibile sulle piattaforme seguenti:

- 

AIX
- 

IBM i
- 

Linux
- 

Windows

e per IBM MQ MQI clients collegati a questi sistemi.

Nome formato

INTESTAZIONE_DIST_MQFM

Serie di caratteri e codifica

I dati in MQDH devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da MQENC_NATIVE.

Impostare la serie di caratteri e la codifica di MQDH nei campi *CodedCharSetId* e *Encoding* in:

- MQMD (se la struttura MQDH è all'inizio dei dati del messaggio) oppure
- La struttura dell'intestazione che precede la struttura MQDH (tutti gli altri casi).

Utilizzo

Quando un'applicazione inserisce un messaggio in un elenco di distribuzione e alcune o tutte le destinazioni sono remote, il gestore code antepone i dati del messaggio dell'applicazione alle strutture MQXQH e MQDH e inserisce il messaggio nella coda di trasmissione pertinente. I dati si verificano quindi nella seguente sequenza quando il messaggio si trova su una coda di trasmissione:

- Struttura MQXQH
- Struttura MQDH più array di record MQOR e MQPMR
- Dati messaggio applicazione

A seconda delle destinazioni, il gestore code può generare più di un messaggio di questo tipo e collocarlo in code di trasmissione differenti. In questo caso, le strutture MQDH in tali messaggi identificano sottoinsiemi differenti delle destinazioni definite dall'elenco di distribuzioni aperto dall'applicazione.

Un'applicazione che inserisce un messaggio dell'elenco di distribuzione direttamente su una coda di trasmissione deve essere conforme alla sequenza descritta in precedenza e deve garantire che la struttura MQDH sia corretta. Se la struttura MQDH non è valida, il gestore code può non riuscire a eseguire la chiamata MQPUT o MQPUT1 con codice motivo MQRC_DH_ERROR.

È possibile memorizzare i messaggi su una coda in formato elenco di distribuzione solo se la coda è stata definita come in grado di supportare i messaggi dell'elenco di distribuzione. Consultare l'attributo coda **DistLists** descritto in “Attributi per le code” a pagina 858. Se un'applicazione inserisce un messaggio dell'elenco di distribuzione direttamente su una coda che non supporta gli elenchi di distribuzione, il

gestore code suddivide il messaggio dell'elenco di distribuzione in singoli messaggi e inserisce invece tali messaggi nella coda.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 484. Campi in MQDH per MQDH

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQDH_	'DH¬¬'
Versione (numero versione struttura)	MQDH_VERSION_1	1
StrucLength (lunghezza della struttura MQDH più i seguenti record)	Nessuna	0
Codifica (codifica numerica dei dati che seguono l'array di record MQPMR)	Nessuna	0
CodedCharSetId (identificativo della serie di caratteri dei dati che seguono un array di record MQPMR)	MQCCSI_UNDEFINED	0
Formato (nome formato dei dati che seguono l'array di record MQPMR)	MQFMT_NONE	Spazi
Indicatori (indicatori generali)	MQDHF_NONE	0
PutMsgRecFields (indicatori che indicano quali campi MQPMR sono presenti)	MQPMRF_NONE	0
RecsPresent (numero di record oggetto presenti)	Nessuna	0
ObjectRecOffset (offset del primo record oggetto dall'inizio di MQDH)	Nessuna	0
PutMsgRecOffset (offset del primo record put - message dall'avvio di MQDH)	Nessuna	0

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macroMQDH_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQDH MyDH = {MQDH_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQDH

```
typedef struct tagMQDH MQDH;
struct tagMQDH {
  MQCHAR4  StrucId;          /* Structure identifier */
  MQLONG   Version;          /* Structure version number */
  MQLONG   StrucLength;      /* Length of MQDH structure plus following
                                MQOR and MQPMR records */
```

```
  MQLONG   Encoding;         /* Numeric encoding of data that follows
                                the MQOR and MQPMR records */
  MQLONG   CodedCharSetId;   /* Character set identifier of data that
                                follows the MQOR and MQPMR records */
  MQCHAR8  Format;           /* Format name of data that follows the
                                MQOR and MQPMR records */
  MQLONG   Flags;            /* General flags */
  MQLONG   PutMsgRecFields;  /* Flags indicating which MQPMR fields are
                                present */
  MQLONG   RecsPresent;      /* Number of MQOR records present */
  MQLONG   ObjectRecOffset;  /* Offset of first MQOR record from start
                                of MQDH */
  MQLONG   PutMsgRecOffset;  /* Offset of first MQPMR record from start
                                of MQDH */
};
```

Dichiarazione COBOL per MQDH

```
**   MQDH structure
  10 MQDH.
**    Structure identifier
   15 MQDH-STRUCID         PIC X(4).
**    Structure version number
   15 MQDH-VERSION         PIC S9(9) BINARY.
**    Length of MQDH structure plus following MQOR and MQPMR records
   15 MQDH-STRUCLENGTH     PIC S9(9) BINARY.
**    Numeric encoding of data that follows the MQOR and MQPMR records
   15 MQDH-ENCODING        PIC S9(9) BINARY.
**    Character set identifier of data that follows the MQOR and MQPMR
**    records
   15 MQDH-CODEDCHARSETID  PIC S9(9) BINARY.
**    Format name of data that follows the MQOR and MQPMR records
   15 MQDH-FORMAT          PIC X(8).
**    General flags
   15 MQDH-FLAGS           PIC S9(9) BINARY.
**    Flags indicating which MQPMR fields are present
   15 MQDH-PUTMSGRECFIELDS PIC S9(9) BINARY.
**    Number of MQOR records present
   15 MQDH-RECSPRESENT     PIC S9(9) BINARY.
**    Offset of first MQOR record from start of MQDH
   15 MQDH-OBJECTRECOFFSET PIC S9(9) BINARY.
**    Offset of first MQPMR record from start of MQDH
   15 MQDH-PUTMSGRECOFFSET PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQDH

```
dcl
 1 MQDH based,
  3 StrucId         char(4),       /* Structure identifier */
  3 Version         fixed bin(31), /* Structure version number */
  3 StrucLength     fixed bin(31), /* Length of MQDH structure plus
                                      following MQOR and MQPMR
                                      records */
  3 Encoding        fixed bin(31), /* Numeric encoding of data that
                                      follows the MQOR and MQPMR
                                      records */
  3 CodedCharSetId  fixed bin(31), /* Character set identifier of data
                                      that follows the MQOR and MQPMR
                                      records */
  3 Format          char(8),       /* Format name of data that follows
                                      the MQOR and MQPMR records */
  3 Flags           fixed bin(31), /* General flags */
  3 PutMsgRecFields fixed bin(31), /* Flags indicating which MQPMR
                                      fields are present */
  3 RecsPresent     fixed bin(31), /* Number of MQOR records present */
  3 ObjectRecOffset fixed bin(31), /* Offset of first MQOR record from
                                      start of MQDH */
  3 PutMsgRecOffset fixed bin(31); /* Offset of first MQPMR record from
                                      start of MQDH */
```

Dichiarazione Visual Basic per MQDH

```
Type MQDH
  StrucId         As String*4 'Structure identifier'
  Version         As Long     'Structure version number'
```

```
   StrucLength     As Long     'Length of MQDH structure plus following'
                               'MQOR and MQPMR records'
   Encoding        As Long     'Numeric encoding of data that follows'
                               'the MQOR and MQPMR records'
   CodedCharSetId  As Long     'Character set identifier of data that'
                               'follows the MQOR and MQPMR records'
   Format          As String*8 'Format name of data that follows the'
                               'MQOR and MQPMR records'
   Flags           As Long     'General flags'
   PutMsgRecFields As Long     'Flags indicating which MQPMR fields are'
                               'present'
   RecsPresent     As Long     'Number of MQOR records present'
   ObjectRecOffset As Long     'Offset of first MQOR record from start'
                               'of MQDH'
   PutMsgRecOffset As Long     'Offset of first MQPMR record from start'
                               'of MQDH'
End Type
```

StrucId (MQCHAR4) per MQDH

Si tratta dell'identificativo della struttura dell'intestazione di distribuzione. È sempre un campo di immissione. Il valore è MQDH_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQDH_
Identificativo per la struttura dell'intestazione di distribuzione.

Per il linguaggi di programmazione C, viene definita anche la costante MQDH_STRUC_ID_ARRAY. Questo ha lo stesso valore di MQDH_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQDH

Il valore deve essere:

MQDH_VERSION_1
Numero di versione per la struttura dell'intestazione di distribuzione.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQDH_CURRENT_
Versione corrente della struttura dell'intestazione di distribuzione.

Il valore iniziale di questo campo è MQDH_VERSION_1.

StrucLength (MQLONG) per MQDH

Questo è il numero di byte dall'inizio della struttura MQDH all'inizio dei dati dei messaggi che seguono gli array di record MQOR e MQPMR. I dati si trovano nella seguente sequenza:

- struttura MQDH
- Array di record di MQOR
- Array di record MQPMR
- Dati messaggio

Gli array dei record MQOR e MQPMR sono indirizzati dagli offset contenuti nella struttura MQDH. Se questi offset risultano in byte inutilizzati tra uno o più della struttura MQDH, gli array di record e i dati del messaggio, tali byte inutilizzati devono essere inclusi nel valore di *StrucLength*, ma il contenuto di tali byte non viene conservato dal gestore code. È valido che l'array di record MQPMR preceda l'array di record MQOR.

Il valore iniziale di questo campo è 0.

Codifica (MQLONG) per MQDH

Questa è la codifica numerica dei dati che seguono gli array di record MQOR e MQPMR; non si applica ai dati numerici nella stessa struttura MQDH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati.

Il valore iniziale di questo campo è 0.

CodedCharSetId (MQLONG) per MQDH

Questo è l'identificativo della serie di caratteri dei dati che seguono gli array di record MQOR e MQPMR; non si applica ai dati carattere nella struttura MQDH stessa.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. È possibile utilizzare il seguente valore speciale:

MQCCSI_INHERIT
Eredita l'identificativo della serie di caratteri di questa struttura.

I dati carattere nei dati *che seguono* questa struttura si trovano nella stessa serie di caratteri di questa struttura.

Il gestore code modifica questo valore nella struttura inviata nel messaggio nell'effettivo identificativo della serie di caratteri della struttura. Se non si verifica alcun errore, la chiamata MQGET non restituisce il valore MQCCSI_INHERIT.

Non è possibile utilizzare MQCCSI_INHERIT se il valore del campo *PutApplType* in MQMD è MQAT_BROKER.

Questo valore è supportato nei seguenti ambienti:

- 

AIX
- 

IBM i
- 

Linux
- 

Windows

e per i client IBM MQ connessi a questi sistemi.

Il valore iniziale di questo campo è MQCCSI_UNDEFINED.

Formato (MQCHAR8) per MQDH

Questo è il nome del formato dei dati che seguono gli array dei record MQOD e MQPMR (a seconda di quale si verifica per ultimo).

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono le stesse del campo *Format* in MQMD.

Il valore iniziale di questo campo è MQFMT_NONE.

Indicatori (MQLONG) per MQDH

È possibile specificare il seguente flag:

ID_MSG_NEW_MQDHF
Generare un nuovo identificativo di messaggio per ogni destinazione nell'elenco di distribuzione. Impostare questa opzione solo quando non sono presenti record di messaggi di inserimento o quando i record sono presenti ma non contengono il campo *MsgId* .

L'utilizzo di questo indicatore differisce la creazione degli identificatori del messaggio fino al momento in cui il messaggio dell'elenco di distribuzione viene finalmente suddiviso in singoli messaggi. Ciò riduce al minimo la quantità di informazioni di controllo che devono essere trasmesse con il messaggio dell'elenco di distribuzione.

Quando un'applicazione inserisce un messaggio in un elenco di distribuzione, il gestore code imposta MQDHF_NEW_MSG_IDS in MQDH che genera quando entrambe le seguenti istruzioni sono vere:

- Non ci sono record put - message forniti dall'applicazione o i record forniti non contengono il campo *MsgId* .
- Il campo *MsgId* in MQMD è MQMI_NONE o il campo *Options* in MQPMO include MQPMO_NEW_MSG_ID

Se non sono necessari indicatori, specificare quanto segue:

MQDHF_NONE
Non è stato specificato alcun indicatore. MQDHF_NONE è definito per aiutare la documentazione del programma. Non è previsto che questa costante venga utilizzata con altre, ma poiché il valore è zero, tale utilizzo non può essere rilevato.

Il valore iniziale di questo campo è MQDHF_NONE.

PutMsgRecFields (MQLONG) per MQDH

È possibile specificare nessuno o più dei seguenti indicatori:

ID_MSG_MQPMRF
Il campo identificativo messaggio è presente.

ID CORREL_MQPMRF
Il campo Identificativo correlazione è presente.

ID_GROUP_MQPMRF
Il campo identificativo gruppo è presente.

MQPMRF_FEEDBACK
Il campo Feedback è presente.

MQPMRF_ACCOUNTING_TOKEN
Il campo Accounting - token è presente.

Se non è presente alcun campo MQPMR, specificare quanto segue:

MQPMRF_NONE
Non sono presenti campi record di messaggi di immissione. MQPMRF_NONE è definito per aiutare la documentazione del programma. Non è previsto che questa costante venga utilizzata con altre, ma poiché il valore è zero, tale utilizzo non può essere rilevato.

Il valore iniziale di questo campo è MQPMRF_NONE.

RecsPresent (MQLONG) per MQDH

Questo è il numero di destinazioni. Un elenco di distribuzione deve contenere sempre almeno una destinazione, quindi *RecsPresent* deve essere sempre maggiore di zero.

Il valore iniziale di questo campo è 0.

Offset ObjectRec(MQLONG) per MQDH

Fornisce l'offset in byte del primo record nell'array di record di oggetti MQOR contenenti i nomi delle code di destinazione. Ci sono *RecsPresent* record in questo array. Questi record (più eventuali byte ignorati tra il primo record oggetto e il campo precedente) sono inclusi nella lunghezza fornita dal campo *StrucLength* .

Un elenco di distribuzione deve contenere sempre almeno una destinazione, quindi *ObjectRecOffset* deve essere sempre maggiore di zero.

Il valore iniziale di questo campo è 0.

PutMsgRecOffset (MQLONG) per MQDH

Fornisce l'offset in byte del primo record nell'array dei record di messaggi di inserimento MQPMR contenenti le proprietà del messaggio. Se presente, ci sono *RecsPresent* record in questo array. Questi record (più eventuali byte saltati tra il record del primo messaggio di inserimento e il campo precedente) sono inclusi nella lunghezza fornita dal campo *StrucLength* .

I record di inserimento messaggi sono facoltativi; se non viene fornito alcun record, *PutMsgRecOffset* è zero e *PutMsgRecFields* ha il valore MQPMRF_NONE.

Il valore iniziale di questo campo è 0.

MQDLH - Intestazione lettera non instradabile

La struttura MQDLH descrive le informazioni che precedano i dati del messaggio dell'applicazione dei messaggi sulla coda dei messaggi non recapitabili (non recapitati). Un messaggio può arrivare sulla DLQ (dead letter queue) perché il gestore code o l'agente del canale dei messaggi lo ha reindirizzato alla coda oppure perché un'applicazione ha inserito il messaggio direttamente nella coda.

Nome formato

MQFMT_DEAD_LETTER_HEADER

Serie di caratteri e codifica

I campi nella struttura MQDLH si trovano nella serie di caratteri e nella codifica fornita dai campi *CodedCharSetId* e *Encoding* . Questi sono specificati nella struttura dell'intestazione che precede MQDLH o nella struttura MQMD se MQDLH si trova all'inizio dei dati del messaggio dell'applicazione.

La serie di caratteri deve essere una serie di caratteri a byte singolo per i caratteri validi nei nomi coda.

Se si utilizzano le classi IBM MQ per Java/JMSe la codepage definita in MQMD non è supportata dalla macchina virtuale Java , MQDLH viene scritto nella serie di caratteri UTF-8 .

Utilizzo

Le applicazioni che inserano i messaggi direttamente nella coda di messaggi non recapitabili devono anteporre ai dati del messaggio una struttura MQDLH e inizializzare i campi con valori appropriati. Tuttavia, il gestore code non richiede che sia presente una struttura MQDLH o che siano specificati valori validi per i campi.

Se un messaggio è troppo lungo per essere inserito nella coda di messaggi non recapitabili, l'applicazione deve effettuare una delle seguenti operazioni:

- Troncare i dati del messaggio per adattarli alla coda dei messaggi non recapitabili.
- Registrare il messaggio nella memoria ausiliaria e inserire un messaggio di report di eccezione nella coda dei messaggi non recapitabili.
- Eliminare il messaggio e restituire un errore al relativo creatore. Se il messaggio è (o potrebbe essere) un messaggio critico, eseguire questa operazione solo se si sa che il mittente ha ancora una copia del messaggio; ad esempio, un messaggio ricevuto da un MCA (message channel agent) da un canale di comunicazione.

Quale delle azioni precedenti è appropriata (se presente) dipende dalla progettazione dell'applicazione.

Il gestore code esegue un'elaborazione speciale quando un messaggio che è un segmento viene inserito con una struttura MQDLH nella parte anteriore; consultare la descrizione della struttura MQMDE per ulteriori dettagli.

Inserimento di messaggi nella coda di messaggi non recapitabili

Quando un messaggio viene inserito nella DLQ (dead letter queue), la struttura MQMD utilizzata per la chiamata MQPUT o MQPUT1 deve essere identica all'MQMD associato al messaggio (generalmente l'MQMD restituito dalla chiamata MQGET), ad eccezione di quanto segue:

- Impostare i campi *CodedCharSetId* e *Encoding* su qualsiasi serie di caratteri e codifica utilizzati per i campi nella struttura MQDLH.
- Impostare il campo *Format* su MQFMT_DEAD_LETTER_HEADER per indicare che i dati iniziano con una struttura MQDLH.

- Impostare i campi di contesto (*AccountingToken*, *ApplIdentityData*, *ApplOriginData*, *PutApplName*, *PutApplType*, *PutDate*, *PutTime*, *UserIdentifier*) utilizzando un'opzione di contesto appropriata alle circostanze:
  - Un'applicazione che inserisce nella coda di messaggi non recapitabili un messaggio non correlato ad alcun messaggio precedente deve utilizzare l'opzione MQPMO_DEFAULT_CONTEXT; ciò fa sì che il gestore code imposti tutti i campi di contesto nel descrittore del messaggio sui relativi valori predefiniti.
  - Un'applicazione server che inserisce nella DLQ (dead letter queue) un messaggio appena ricevuto deve utilizzare l'opzione MQPMO_PASS_ALL_CONTEXT per preservare le informazioni di contesto originali.
  - Un'applicazione server che inserisce nella DLQ (dead letter queue) una *risposta* a un messaggio che ha appena ricevuto deve utilizzare l'opzione MQPMO_PASS_IDENTITY_CONTEXT; questa opzione conserva le informazioni di identità ma imposta le informazioni di origine come quelle dell'applicazione server.
  - Un MCA (message channel agent) che inserisce nella DLQ (dead letter queue) un messaggio ricevuto dal proprio canale di comunicazione deve utilizzare l'opzione MQPMO_SET_ALL_CONTEXT per preservare le informazioni di contesto originali.

Nella struttura MQDLH stessa, impostare i campi come segue:

- Impostare i campi *CodedCharSetId*, *Encoding*e *Format* sui valori che descrivono i dati che seguono la struttura MQDLH, generalmente i valori del descrittore del messaggio originale.
- Impostare i campi di contesto *PutApplType*, *PutApplName*, *PutDate*e *PutTime* sui valori appropriati per l'applicazione che inserisce il messaggio nella coda di messaggi non instradabili; questi valori non sono correlati al messaggio originale.
- Impostare altri campi come appropriato.

Assicurarsi che tutti i campi abbiano valori validi e che i campi carattere siano riempiti con spazi vuoti fino alla lunghezza definita del campo; non terminare prematuramente i dati carattere utilizzando un carattere null, poiché il gestore code non converte i caratteri null e successivi in spazi vuoti nella struttura MQDLH.

Richiamo dei messaggi dalla coda dei messaggi non recapitabili

Le applicazioni che ricevono i messaggi dalla coda dei messaggi non recapitabili devono verificare che i messaggi inizino con una struttura MQDLH. L'applicazione può stabilire se è presente una struttura MQDLH esaminando il campo *Format* nel descrittore del messaggio MQMD; se il campo ha il valore MQFMT_DEAD_LETTER_HEADER, i dati del messaggio iniziano con una struttura MQDLH. Tenere presente, inoltre, che i messaggi che le applicazioni ricevono dalla coda dei messaggi non recapitabili potrebbero essere troncati se in origine erano troppo lunghi per la coda.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 485. Campi in MQDLH per MQDLH

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQDLH	'DLH¬'
Versione (numero versione struttura)	MQDLH_VERSION_1	1
Motivo (messaggio di errore arrivato nella coda di messaggi non recapitabili)	MQRC_NONE	0

Tabella 485. Campi in MQDLH per MQDLH (Continua)

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
DestQName (nome della coda di destinazione originale)	Nessuna	Stringa null o spazi vuoti
DestQMgrNome (nome del gestore code di destinazione originale)	Nessuna	Stringa null o spazi vuoti
Codifica (codifica numerica dei dati che seguono MQDLH)	Nessuna	0
CodedCharSetId (identificativo della serie di caratteri dei dati che seguono MQDLH)	MQCCSI_UNDEFINED	0
Formato (nome formato dei dati che seguono MQDLH)	MQFMT_NONE	Spazi
PutApplTipo (tipo di applicazione che inserisce il messaggio nella coda di messaggi non recapitabili)	Nessuna	0
PutApplNome (nome dell'applicazione che inserisce il messaggio nella coda di messaggi non instradabili)	Nessuna	Stringa null o spazi vuoti
PutDate (data in cui il messaggio è stato inserito nella coda di messaggi non instradabili)	Nessuna	Stringa null o spazi vuoti
PutTime (ora in cui il messaggio è stato inserito nella coda dei messaggi non instradabili)	Nessuna	Stringa null o spazi vuoti

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.
3. Nel linguaggio di programmazione C, la variabile macroMQDLH_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQDLH MyDLH = {MQDLH_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQDLH

```
typedef struct tagMQDLH MQDLH;
struct tagMQDLH {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Reason;          /* Reason message arrived on dead-letter
                                (undelivered-message) queue */
  MQCHAR48  DestQName;       /* Name of original destination queue */
  MQCHAR48  DestQMgrName;    /* Name of original destination queue
                                manager */
  MQLONG    Encoding;        /* Numeric encoding of data that follows
                                MQDLH */
  MQLONG    CodedCharSetId;  /* Character set identifier of data that
                                follows MQDLH */
  MQCHAR8   Format;          /* Format name of data that follows
                                MQDLH */
```

```
  MQLONG    PutApplType;     /* Type of application that put message on
                                dead-letter (undelivered-message)
                                queue */
  MQCHAR28  PutApplName;     /* Name of application that put message on
                                dead-letter (undelivered-message)
                                queue */
  MQCHAR8   PutDate;         /* Date when message was put on dead-letter
                                (undelivered-message) queue */
  MQCHAR8   PutTime;         /* Time when message was put on the
                                dead-letter (undelivered-message)
                                queue */
};
```

Dichiarazione COBOL per MQDLH

```
**   MQDLH structure
  10 MQDLH.
**    Structure identifier
   15 MQDLH-STRUCID        PIC X(4).
**    Structure version number
   15 MQDLH-VERSION        PIC S9(9) BINARY.
**    Reason message arrived on dead-letter (undelivered-message) queue
   15 MQDLH-REASON         PIC S9(9) BINARY.
**    Name of original destination queue
   15 MQDLH-DESTQNAME      PIC X(48).
**    Name of original destination queue manager
   15 MQDLH-DESTQMGRNAME   PIC X(48).
**    Numeric encoding of data that follows MQDLH
   15 MQDLH-ENCODING       PIC S9(9) BINARY.
**    Character set identifier of data that follows MQDLH
   15 MQDLH-CODEDCHARSETID PIC S9(9) BINARY.
**    Format name of data that follows MQDLH
   15 MQDLH-FORMAT         PIC X(8).
**    Type of application that put message on dead-letter
**    (undelivered-message) queue
   15 MQDLH-PUTAPPLTYPE    PIC S9(9) BINARY.
**    Name of application that put message on dead-letter
**    (undelivered-message) queue
   15 MQDLH-PUTAPPLNAME    PIC X(28).
**    Date when message was put on dead-letter (undelivered-message)
**    queue
   15 MQDLH-PUTDATE        PIC X(8).
**    Time when message was put on the dead-letter (undelivered-message)
**    queue
   15 MQDLH-PUTTIME        PIC X(8).
```

Dichiarazione PL/I per MQDLH

```
dcl
 1 MQDLH based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Reason         fixed bin(31), /* Reason message arrived on
                                     dead-letter (undelivered-message)
                                     queue */
  3 DestQName      char(48),      /* Name of original destination
                                     queue */
  3 DestQMgrName   char(48),      /* Name of original destination queue
                                     manager */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
                                     follows MQDLH */
  3 CodedCharSetId fixed bin(31), /* Character set identifier of data
                                     that follows MQDLH */
  3 Format         char(8),       /* Format name of data that follows
                                     MQDLH */
  3 PutApplType    fixed bin(31), /* Type of application that put
                                     message on dead-letter
                                     (undelivered-message) queue */
  3 PutApplName    char(28),      /* Name of application that put
                                     message on dead-letter
                                     (undelivered-message) queue */
  3 PutDate        char(8),       /* Date when message was put on
                                     dead-letter (undelivered-message)
                                     queue */
  3 PutTime        char(8);       /* Time when message was put on the
```

```
                                    dead-letter (undelivered-message)
                                    queue */
```

Dichiarazione High Level Assembler per MQDLH

```
MQDLH                 DSECT
MQDLH_STRUCID         DS   CL4   Structure identifier
MQDLH_VERSION         DS   F     Structure version number
MQDLH_REASON          DS   F     Reason message arrived on dead-letter
*                                (undelivered-message) queue
MQDLH_DESTQNAME       DS   CL48  Name of original destination queue
MQDLH_DESTQMGRNAME    DS   CL48  Name of original destination queue
*                                manager
MQDLH_ENCODING        DS   F     Numeric encoding of data that follows
*                                MQDLH
MQDLH_CODEDCHARSETID  DS   F     Character set identifier of data that
*                                follows MQDLH
MQDLH_FORMAT          DS   CL8   Format name of data that follows MQDLH
MQDLH_PUTAPPLTYPE     DS   F     Type of application that put message on
*                                dead-letter (undelivered-message) queue
MQDLH_PUTAPPLNAME     DS   CL28  Name of application that put message on
*                                dead-letter (undelivered-message) queue
MQDLH_PUTDATE         DS   CL8   Date when message was put on
*                                dead-letter (undelivered-message) queue
MQDLH_PUTTIME         DS   CL8   Time when message was put on the
*                                dead-letter (undelivered-message) queue
*
MQDLH_LENGTH          EQU  *-MQDLH
                      ORG  MQDLH
MQDLH_AREA            DS   CL(MQDLH_LENGTH)
```

Dichiarazione di Visual Basic per MQDLH

```
Type MQDLH
  StrucId        As String*4  'Structure identifier'
  Version        As Long      'Structure version number'
  Reason         As Long      'Reason message arrived on dead-letter'
                              '(undelivered-message) queue'
  DestQName      As String*48 'Name of original destination queue'
  DestQMgrName   As String*48 'Name of original destination queue'
                              'manager'
  Encoding       As Long      'Numeric encoding of data that follows'
                              'MQDLH'
  CodedCharSetId As Long      'Character set identifier of data that'
                              'follows MQDLH'
  Format         As String*8  'Format name of data that follows MQDLH'
  PutApplType    As Long      'Type of application that put message on'
                              'dead-letter (undelivered-message) queue'
  PutApplName    As String*28 'Name of application that put message on'
                              'dead-letter (undelivered-message) queue'
  PutDate        As String*8  'Date when message was put on dead-letter'
                              '(undelivered-message) queue'
  PutTime        As String*8  'Time when message was put on the'
                              'dead-letter (undelivered-message) queue'
End Type
```

StrucId (MQCHAR4) per MQDLH

Questo è l'identificativo della struttura dell'intestazione dei messaggi non recapitabili. È sempre un campo di immissione. Il suo valore è MQDLH_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQDLH
Identificativo per la struttura dell'intestazione dei messaggi non recapitabili.

Per il linguaggio di programmazione C, viene definita anche la costante MQDLH_STRUC_ID_ARRAY. Ha lo stesso valore di MQDLH_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQDLH

Versione è il numero di versione della struttura.

Il valore deve essere:

MQDLH_VERSION_1
Numero di versione per la struttura dell'intestazione dei messaggi non recapitabili.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQDLH_CURRENT_
La versione corrente della struttura dell'intestazione dei messaggi non instradabili.

Il valore iniziale di questo campo è MQDLH_VERSION_1.

Motivo (MQLONG) per MQDLH
Il campo Motivo identifica il motivo per cui il messaggio è stato collocato nella coda di messaggi non instradabili anziché nella coda di destinazione originale.

Ciò identifica il motivo per cui il messaggio è stato inserito nella coda di messaggi non instradabili invece che nella coda di destinazione originale. Deve essere uno dei valori MQFB_ * o MQRC_ * (ad esempio, MQRC_Q_FULL). Consultare la descrizione del campo *Feedback* in "MQMD - Descrittore messaggi" a pagina 431 per i dettagli dei valori MQFB_ * comuni che possono verificarsi.

Se il valore è compreso nell'intervallo tra MQFB_IMS_FIRST e MQFB_IMS_LAST, il codice di errore IMS effettivo può essere determinato sottraendo MQFB_IMS_ERROR dal valore del campo *Reason* .

Alcuni valori MQFB_* vengono applicati solo in questo campo. Sono correlati ai messaggi del repository, ai messaggi di trigger o ai messaggi della coda di trasmissione che sono stati trasferiti alla coda di messaggi non recapitabili. Essi sono:

MQFB_APPL_CANNOT_BE_STARTED (X'00000109')
Un'applicazione che elabora un messaggio di trigger non può avviare l'applicazione indicata nel campo *ApplId* del messaggio di trigger (consultare "MQTM - Messaggio trigger" a pagina 616).

Su z/OS, la transazione CKTI CICS è un esempio di un'applicazione che elabora i messaggi trigger.

MQFB_APPL_TYPE_ERROR (X'0000010B')
Un'applicazione che elabora un messaggio di trigger non può avviare l'applicazione perché il campo *ApplType* del messaggio di trigger non è valido (consultare "MQTM - Messaggio trigger" a pagina 616).

Su z/OS, la transazione CKTI CICS è un esempio di un'applicazione che elabora i messaggi trigger.

MQFB_BIND_OPEN_CLUSRCVR_DEL (X'00000119')
Il messaggio era sul SISTEMA SYSTEM.CLUSTER.TRANSMIT.QUEUE per una coda cluster aperta con l'opzione MQOO_BIND_ON_OPEN, ma il canale ricevente del cluster remoto da utilizzare per trasmettere il messaggio alla coda di destinazione è stato eliminato prima che il messaggio potesse essere inviato. Poiché è stato specificato MQOO_BIND_ON_OPEN, solo il canale selezionato quando è stata aperta la coda può essere utilizzato per trasmettere il messaggio. Poiché questo canale non è più disponibile, il messaggio viene inserito nella coda di messaggi non recapitabili.

MQFB_NOT_A_REPOSITORY_MSG (X'00000118')
Il messaggio non è un messaggio del repository.

MQFB_STOPPED_BY_CHAD_EXIT (X'00000115')
Il messaggio è stato arrestato dall'uscita di definizione automatica del canale.

MQFB_STOPPED_BY_MSG_EXIT (X'0000010D')
Il messaggio è stato arrestato dall'uscita del messaggio del canale.

ERRORE TM_MQFB (X'0000010A')
Il campo *Format* in MQMD specifica MQFMT_TRIGGER, ma il messaggio non inizia con una struttura MQTM valida. Ad esempio, l'eye-catcher mnemonico *StrucId* potrebbe non essere valido, il *Version* potrebbe non essere riconosciuto o la lunghezza del messaggio trigger potrebbe non essere sufficiente per contenere la struttura MQTM.

Su z/OS, la transazione CKTI CICS è un esempio di applicazione che elabora i messaggi di trigger e può generare questo codice di feedback.

MQFB_XMIT_Q_MSG_ERROR (X'0000010F')
Un agente del canale dei messaggi ha trovato che un messaggio sulla coda di trasmissione non è nel formato corretto. L'agente del canale dei messaggi inserisce il messaggio nella coda di messaggi non recapitabili utilizzando questo codice di feedback.

Una causa comune è che un messaggio è stato inserito direttamente nella coda di trasmissione, quindi il messaggio non ha l'intestazione XQH prevista. I messaggi devono essere inseriti in una coda di trasmissione attraverso una coda remota, a meno che l'applicazione non crei l'intestazione MQXQH.

Il valore iniziale di questo campo è MQRC_NONE.

DestQName (MQCHAR48) per MQDLH
DestQName è il nome della coda messaggi che era la destinazione originale per il messaggio.

La lunghezza di questo campo è fornita da MQ_Q_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

DestQMgrNome (MQCHAR48) per MQDLH
DestQMgrIl nome del gestore code che era la destinazione originale del messaggio.

La lunghezza di questo campo viene fornita da MQ_Q_MGR_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

Codifica (MQLONG) per MQDLH
La codifica è la codifica numerica dei dati che seguono la struttura MQDLH (di solito i dati del messaggio originale); non si applica ai dati numerici nella struttura MQDLH stessa.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati.

Il valore iniziale di questo campo è `0`.

CodedCharSetId (MQLONG) per MQDLH
CodedCharSetId è l'identificativo della serie di caratteri dei dati che passano attraverso la struttura MQDLH (di norma i dati del messaggio originale); non si applica ai dati di carattere nella struttura MQDLH stessa.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. È possibile utilizzare il seguente valore speciale:

MQCCSI_INHERIT
I dati carattere nei dati che seguono questa struttura si trovano nella stessa serie di caratteri di questa struttura.

Il gestore code modifica questo valore nella struttura inviata nel messaggio nell'effettivo identificativo della serie di caratteri della struttura. Se non si verifica alcun errore, il valore MQCCSI_INHERIT non viene restituito dalla chiamata MQGET.

Non è possibile utilizzare MQCCSI_INHERIT se il valore del campo *PutApplType* in MQMD è MQAT_BROKER.

Questo valore è supportato nei seguenti ambienti:

- AIX
- IBM i
- Linux

- Windows Windows

e per i client IBM MQ connessi a questi sistemi.

Il valore iniziale di questo campo è MQCCSI_UNDEFINED.

Formato (MQCHAR8) per MQDLH
Il formato è il nome formato dei dati che seguono la struttura MQDLH (di solito i dati del messaggio originale).

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono uguali a quelle per la codifica del campo *Format* in MQMD.

La lunghezza di questo campo è fornita da MQ_FORMAT_LENGTH. Il valore iniziale di questo campo è MQFMT_NONE.

PutApplTipo (MQLONG) per MQDLH
PutApplIl tipo di applicazione che inserisce il messaggio nella coda di messaggi non recapitabili.

Questo campo ha lo stesso significato del campo *PutApplType* nel descrittore del messaggio MQMD (consultare “MQMD - Descrittore messaggi” a pagina 431 per i dettagli).

Se il gestore code reindirizza il messaggio alla coda di messaggi non instradabili, *PutApplType* ha il valore MQAT_QMGR.

Il valore iniziale di questo campo è 0.

Nome PutAppl(MQCHAR28) per MQDLH
PutApplIl nome è il nome dell'applicazione che inserisce il messaggio nella coda di messaggi non recapitabili.

Il formato del nome dipende dal campo *PutApplType* . Il formato può variare da release a release. Consultare la descrizione del campo *PutApplName* in “MQMD - Descrittore messaggi” a pagina 431.

Se il gestore code reindirizza il messaggio alla coda di messaggi non recapitabili, *PutApplName* contiene i primi 28 caratteri del nome del gestore code, riempiti con spazi vuoti, se necessario.

La lunghezza di questo campo è fornita da MQ_PUT_APPL_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 28 caratteri vuoti in altri linguaggi di programmazione.

PutDate (MQCHAR8) per MQDLH
PutDate è la data in cui il messaggio è stato inserito nella coda di messaggi non recapitabili.

Il formato utilizzato per la data in cui questo campo viene generato dal gestore code è:

- AAAAMMGG

dove i caratteri rappresentano:

AAAA
anno (quattro cifre)

MI
mese dell'anno (da 01 a 12)

GG
giorno del mese (da 01 a 31)

GMT (Greenwich Mean Time) viene utilizzato per i campi *PutDate* e *PutTime* , in base all'orologio di sistema impostato in modo accurato su GMT.

La lunghezza di questo campo è fornita da MQ_PUT_DATE_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e otto caratteri vuoti in altri linguaggi di programmazione.

PutTime (MQCHAR8) per MQDLH

PutTime è l'ora in cui il messaggio è stato inserito nella coda dei messaggi non recapitabili (messaggi non recapitabili).

Il formato utilizzato per l'ora in cui questo campo viene generato dal gestore code è:

- HHMMSSTH

dove i caratteri rappresentano:

OO
ore (da 00 a 23)

MI
minuti (da 00 a 59)

SS
secondi (da 00 a 59; vedere nota)

T
decimi di secondo (da 0 a 9)

H
centesimi di secondo (da 0 a 9)

Nota: Se l'orologio di sistema è sincronizzato con uno standard di tempo molto accurato, è possibile in rare occasioni che 60 o 61 vengano restituiti per i secondi in *PutTime*. Ciò si verifica quando i secondi bisestili vengono inseriti nello standard temporale globale.

GMT (Greenwich Mean Time) viene utilizzato per i campi *PutDate* e *PutTime* , in base all'orologio di sistema impostato in modo accurato su GMT.

La lunghezza di questo campo è fornita da MQ_PUT_TIME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e otto caratteri vuoti in altri linguaggi di programmazione.

MQDMHO - Opzioni di eliminazione dell'handle del messaggio

La struttura **MQDMHO** consente alle applicazioni di specificare le opzioni che controllano il modo in cui vengono eliminati gli handle dei messaggi. La struttura è un parametro di immissione sulla chiamata **MQDLTMH** .

Serie di caratteri e codifica

I dati in **MQDMHO** devono essere nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (**MQENC_NATIVE**).

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 486. Campi in MQDMHO

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID MQDMHO_STRUC_ID	'DMHO'
Versione (numero versione struttura)	MQDMHO_VERSION_1	1
Opzioni (opzioni)	MQDMHO_NONE	0

Tabella 486. Campi in MQDMHO (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**

Note:

1. Nel linguaggio di programmazione C, la variabile macroMQDMHO_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura:

```
MQDMHO MyDMHO = {MQDMHO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQDMHO

```
typedef struct tagMQDMHO;
struct tagMQDMHO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of MQDLTMH */
};
```

Dichiarazione COBOL per MQDMHO

```
**   MQDMHO structure
  10 MQDMHO.
**    Structure identifier
   15 MQDMHO-STRUCID       PIC X(4).
**    Structure version number
   15 MQDMHO-VERSION       PIC S9(9) BINARY.
**    Options that control the action of MQDLTMH
   15 MQDMHO-OPTIONS       PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQDMHO

```
dcl
 1 MQDMHO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action of MQDLTMH */
```

Dichiarazione High Level Assembler per MQDMHO

```
MQDMHO                DSECT
MQDMHO_STRUCID        DS   CL4   Structure identifier
MQDMHO_VERSION        DS   F     Structure version number
MQDMHO_OPTIONS        DS   F     Options that control the action of
*                                MQDLTMH
MQDMHO_LENGTH         EQU  *-MQDMHO
MQDMHO_AREA           DS   CL(MQDMHO_LENGTH)
```

StrucId (MQCHAR4) per MQDMHO

Questo è l'identificativo della struttura delle opzioni di eliminazione dell'handle del messaggio. È sempre un campo di immissione. Il valore è MQDMHO_STRUC_ID.

Il valore deve essere:

ID MQDMHO_STRUC_ID

Identificativo per la struttura delle opzioni di gestione del messaggio di eliminazione.

Per il linguaggio di programmazione C, viene definita anche la costante **MQDMHO_STRUC_ID_ARRAY** . Ha lo stesso valore di **MQDMHO_STRUC_ID**, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQDMHO

Questo è il numero di versione della struttura; il valore deve essere:

MQDMHO_VERSION_1
Version-1 eliminare la struttura di opzioni dell'handle del messaggio.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQDMHO_CURRENT_
Versione corrente della struttura delle opzioni di gestione dei messaggi di eliminazione.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **MQDMHO_VERSION_1**.

Opzioni (MQLONG) per MQDMHO

Il valore deve essere:

MQDMHO_NONE

Nessuna opzione specificata.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **MQDMHO_NONE**.

MQDMPO - Opzioni di eliminazione proprietà messaggio

La struttura MQDMPO consente alle applicazioni di specificare le opzioni che controllano il modo in cui vengono eliminate le proprietà dei messaggi. La struttura è un parametro di input sulla chiamata MQDLTMP.

Serie di caratteri e codifica

I dati in MQDMPO devono essere nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (MQENC_NATIVE).

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 487. Campi in MQDPMO

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQDMPO	'DMPO'
Versione (numero versione struttura)	MQDMPO_VERSION_1	1
Opzioni (opzioni che controllano l'azione di MQDMPO)	Opzioni che controllano l'azione di MQDLTMP	MQDMPO_NONE

Tabella 487. Campi in MQDPMO (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**

Note:

1. Nel linguaggio di programmazione C, la variabile macroMQDMPO_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQDMPO MyDMPO = {MQDMPO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQDMPO

```
typedef struct tagMQDMPO MQDMPO;
struct tagMQDMPO {
  MQCHAR4  StrucId;        /* Structure identifier */
  MQLONG   Version;        /* Structure version number */
  MQLONG   Options;        /* Options that control the action of
                              MQDLTMP */
};
```

Dichiarazione COBOL per MQDMPO

```
**   MQDMPO structure
  10 MQDMPO.
**    Structure identifier
   15 MQDMPO-STRUCID               PIC X(4).
**    Structure version number
   15 MQDMPO-VERSION               PIC S9(9) BINARY.
**    Options that control the action of MQDLTMP
   15 MQDMPO-OPTIONS               PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQDMPO

```
Dcl
 1 MQDMPO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action
                                     of MQDLTMP */
```

Dichiarazione High Level Assembler per MQDMPO

```
MQDMPO                  DSECT
MQDMPO_STRUCID          DS   CL4  Structure identifier
MQDMPO_VERSION          DS   F    Structure version number
MQDMPO_OPTIONS          DS   F    Options that control the
*                                 action of MQDLTMP
MQDMPO_LENGTH           EQU  *-MQDMPO
MQDMPO_AREA             DS   CL(MQDMPO_LENGTH)
```

StrucId (MQCHAR4) per MQDMPO

È l'identificativo della struttura della struttura delle opzioni della proprietà di eliminazione del messaggio. È sempre un campo di immissione. Il valore è MQDMPO_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQDMPO
Identificativo per la struttura delle opzioni di eliminazione della proprietà del messaggio.

Per il linguaggio di programmazione C, viene definita anche la costante MQDMPO_STRUC_ID_ARRAY. Ha lo stesso valore di MQDMPO_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQDMPO
Elimina struttura delle opzioni delle proprietà del messaggio - Campo Versione

Questo è il numero di versione della struttura. Il valore deve essere:

MQDMPO_VERSION_1
Numero di versione per la struttura di opzioni della proprietà di eliminazione del messaggio.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQDMPO_CURRENT_
Versione corrente della struttura di opzioni della proprietà di eliminazione del messaggio.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQDMPO_VERSION_1.

Opzioni (MQLONG) per MQDMPO
Struttura delle opzioni della proprietà Elimina messaggio - Campo Opzioni

Opzioni di ubicazione: le opzioni riportate di seguito si riferiscono alla posizione relativa della proprietà rispetto al cursore della proprietà.

MQDMPO_DEL_PRIMO
Elimina la prima proprietà che corrisponde al nome specificato.

MQDMPO_DEL_PROP_UNDER_CURSOR
Elimina la proprietà indicata dal cursore della proprietà; questa è la proprietà che è stata interrogata l'ultima volta utilizzando l'opzione MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

Il cursore della proprietà viene reimpostato quando l'handle del messaggio viene riutilizzato. Viene anche reimpostato quando l'handle del messaggio viene specificato nel campo *MsgHandle* della struttura MQGMO su una chiamata MQGET o la struttura MQPMO su una chiamata MQPUT.

Se questa opzione viene utilizzata quando il cursore della proprietà non è stato ancora stabilito, la chiamata ha esito negativo con codice di completamento MQCC_FAILED e motivo MQRC_PROPERTY_NOT_AVAILABLE. Se la proprietà indicata dal cursore della proprietà è già stata eliminata, la chiamata ha esito negativo anche con codice di completamento MQCC_FAILED e motivo MQRC_PROPERTY_NOT_AVAILABLE.

Se nessuna delle opzioni è richiesta, è possibile utilizzare la seguente opzione:

MQDMPO_NONE
Nessuna opzione specificata.

Questo campo è sempre un campo di input. Il valore iniziale di questo campo è MQDMPO_DEL_FIRST.

MQEPH - Intestazione PCF integrata

La struttura MQEPH descrive i dati aggiuntivi che sono presenti in un messaggio quando tale messaggio è un messaggio PCF (programmable command format). Il campo *PCFHeader* definisce i parametri PCF che seguono questa struttura e ciò consente di seguire i dati del messaggio PCF con altre intestazioni.

Nome formato

MQFMT_EMBEDDED_PCF

Serie di caratteri e codifica

I dati in MQEPH devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da MQENC_NATIVE.

Impostare la serie di caratteri e la codifica di MQEPH nei campi *CodedCharSetId* e *Encoding* in MQMD (se la struttura MQEPH si trova all'inizio dei dati del messaggio) o la struttura dell'intestazione che precede la struttura MQEPH (tutti gli altri casi).

Utilizzo

Non è possibile utilizzare strutture MQEPH per inviare comandi al server dei comandi o a qualsiasi altro server di accettazione PCF del gestore code.

Allo stesso modo, il server dei comandi o qualsiasi altro server di accettazione PCF del gestore code non genera risposte o eventi contenenti strutture MQEPH.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 488. Campi in MQEPH per MQEPH

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQEPH	'EPH¬'
Versione (numero versione struttura)	MQEPH_VERSION_1	1
StrucLength (lunghezza della struttura MQEPH più le strutture MQCFH e dei parametri che la seguono)	MQEPH_STRUC_LENGTH_FIXED	68
Codifica (codifica numerica dei dati che seguono l'ultima struttura del parametro PCF)	Nessuna	0
CodedCharSetId (identificativo della serie di caratteri dei dati che seguono l'ultima struttura di parametri PCF)	MQCCSI_UNDEFINED	0
Formato (nome formato dei dati che seguono l'ultima struttura del parametro PCF)	MQFMT_NONE	Spazi
Indicatori (indicatori)	MQEPH_NONE	0
PCFHeader (intestazione PCF (programmable command format))	Nomi e valori come definiti in Tabella 489 a pagina 376	0

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macroMQEPH_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQEPH MyEPH = {MQEPH_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQEPH

```
typedef struct tagMQEPH MQEPH;
struct tagMQDH {
  MQCHAR4  StrucId;          /* Structure identifier */
  MQLONG   Version;          /* Structure version number */
  MQLONG   StrucLength;      /* Total length of MQEPH including the MQCFH
                                and parameter structures that follow it */
  MQLONG   Encoding;         /* Numeric encoding of data that follows last
                                PCF parameter structure */
  MQLONG   CodedCharSetId;   /* Character set identifier of data that
                                follows last PCF parameter structure */
  MQCHAR8  Format;           /* Format name of data that follows last PCF
                                parameter structure */
  MQLONG   Flags;            /* Flags */
  MQCFH    PCFHeader;        /* Programmable command format header */
 };
```

Dichiarazione COBOL per MQEPH

```
**   MQEPH structure
  10 MQEPH.
**    Structure identifier
   15 MQEPH-STRUCID         PIC X(4).
**    Structure version number
   15 MQEPH-VERSION         PIC S9(9) BINARY.
**    Total length of MQEPH structure including the MQCFH
**    and parameter structures that follow it
   15 MQEPH-STRUCLENGTH     PIC S9(9) BINARY.
**    Numeric encoding of data that follows last
**    PCF structure
   15 MQEPH-ENCODING        PIC S9(9) BINARY.
**    Character set identifier of data that
**    follows last PCF parameter structure
   15 MQEPH-CODEDCHARSETID  PIC S9(9) BINARY.
**    Format name of data that follows last PCF
**    parameter structure
   15 MQEPH-FORMAT          PIC X(8).
**    Flags
   15 MQEPH-FLAGS           PIC S9(9) BINARY.
**    Programmable command format header
   15 MQEPH-PCFHEADER.
**      Structure type
     20 MQEPH-PCFHEADER-TYPE           PIC S9(9) BINARY.
**      Structure length
     20 MQEPH-PCFHEADER-STRUCLENGTH    PIC S9(9) BINARY.
**      Structure version number
     20 MQEPH-PCFHEADER-VERSION        PIC S9(9) BINARY.
**      Command identifier
     20 MQEPH-PCFHEADER-COMMAND        PIC S9(9) BINARY.
**      Message sequence number
     20 MQEPH-PCFHEADER-MSGSEQNUMBER   PIC S9(9) BINARY.
**      Control options
     20 MQEPH-PCFHEADER-CONTROL        PIC S9(9) BINARY.
**      Completion code
     20 MQEPH-PCFHEADER-COMPCODE       PIC S9(9) BINARY.
**      Reason code qualifying completion code
     20 MQEPH-PCFHEADER-REASON         PIC S9(9) BINARY.
**      Count of parameter structures
     20 MQEPH-PCFHEADER-PARAMETERCOUNT PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQEPH

```
dcl
 1 MQEPH based,
  3 StrucId         char(4),       /* Structure identifier */
  3 Version         fixed bin(31), /* Structure version number */
  3 StrucLength     fixed bin(31), /* Total Length of MQEPH including the
                                      MQCFH and parameter structures that
                                      follow it
  3 Encoding        fixed bin(31), /* Numeric encoding of data that follows
                                      last PCF parameter structure
  3 CodedCharSetId  fixed bin(31), /* Character set identifier of data that
```

```
                                   follows last PCF parameter structure
  3 Format          char(8),       /* Format name of data that follows last
                                      PCF parameter structure */
  3 Flags           fixed bin(31), /* Flags */
  3 PCFHeader,                     /* Programmable command format header
   5 Type           fixed bin(31), /* Structure type */
   5 StrucLength    fixed bin(31), /* Structure length */
   5 Version        fixed bin(31), /* Structure version number */
   5 Command        fixed bin(31), /* Command identifier */
   5 MsgseqNumber   fixed bin(31), /* Message sequence number */
   5 Control        fixed bin(31), /* Control options */
   5 CompCode       fixed bin(31), /* Completion code */
   5 Reason         fixed bin(31), /* Reason code qualifying completion code */
   5 ParameterCount fixed bin(31); /* Count of parameter structures */
```

Dichiarazione High Level Assembler per MQEPH

```
MQEPH                           DSECT
MQEPH_STRUCID                   DS   CL4   Structure identifier
MQEPH_VERSION                   DS   F     Structure version number
MQEPH_STRUCLENGTH               DS   F     Total length of MQEPH including the
*                                          MQCFH and parameter structures that
                                           follow it
MQEPH_ENCODING                  DS   F     Numeric encoding of data that follows
*                                          last PCF parameter structure
MQEPH_CODEDCHARSETID            DS   F     Character set identifier of data that
*                                          follows last PCF parameter structure
MQEPH_FORMAT                    DS   CL8   Format name of data that follows last
*                                          PCF parameter structure
MQEPH_FLAGS                     DS   F     Flags
MQEPH_PCFHEADER                 DS   0F    Force fullword alignment
MQEPH_PCFHEADER_TYPE            DS   F     Structure type
MQEPH_PCFHEADER_STRUCLENGTH     DS   F     Structure length
MQEPH_PCFHEADER_VERSION         DS   F     Structure version number
MQEPH_PCFHEADER_COMMAND         DS   F     Command identifier
MQEPH_PCFHEADER_MSGSEQNUMBER    DS   F     Structure length
MQEPH_PCFHEADER_CONTROL         DS   F     Control options
MQEPH_PCFHEADER_COMPCODE        DS   F     Completion code
MQEPH_PCFHEADER_REASON          DS   F     Reason code qualifying completion code
MQEPH_PCFHEADER_PARAMETER COUNT DS   F     Count of parameter structures
MQEPH_PCFHEADER_LENGTH          EQU  *-MQEPH_PCFHEADER
                                ORG  MQEPH_PCFHEADER
MQEPH_PCFHEADER_AREA            DS   CL(MQEPH_PCFHEADER_LENGTH)
*
MQEPH_LENGTH                    EQU  *-MQEPH
                                ORG  MQEPH
MQEPH_AREA                      DS   CL(MQEPH_LENGTH)
```

Dichiarazione Visual Basic per MQEPH

```
Type MQEPH
  StrucId         As String*4 'Structure identifier'
  Version         As Long     'Structure version number'
  StrucLength     As Long     'Total length of MQEPH structure including the MQCFH'
                              'and parameter structures that follow it'
  Encoding        As Long     'Numeric encoding of data that follows last'
                              'PCF parameter structure'
  CodedCharSetId  As Long     'Character set identifier of data that'
                              'follows last PCF parameter structure'
  Format          As String*8 'Format name of data that follows last PCF'
                              'parameter structure'
  Flags           As Long     'Flags'
  PCFHeader       As MQCFH    'Programmable command format header'
 End Type

Global MQEPH_DEFAULT As MQEPH
```

StrucId (MQCHAR4) per MQEPH

Questo è l'identificativo della struttura dell'intestazione PCF incorporata. È sempre un campo di immissione. Il valore è MQEPH_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQEPH
L'identificativo per la struttura dell'intestazione PCF incorporata.

Per il linguaggio di programmazione C, viene definita anche la costante MQEPH_STRUC_ID_ARRAY. Ha lo stesso valore di MQEPH_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQEPH

Il valore deve essere:

MQEPH_VERSION_1
Numero versione per la struttura di intestazione PCF incorporata.

La seguente costante specifica il numero di versione della versione corrente:

MQCFH_VERSION_3
La versione corrente della struttura dell'intestazione PCF incorporata.

Il valore iniziale di questo campo è MQEPH_VERSION_1.

StrucLength (MQLONG) per MQEPH

Questa è la quantità di dati che precedono la successiva struttura dell'intestazione. Comprendono:

- La lunghezza dell'intestazione MQEPH
- La lunghezza di tutti i parametri PCF che seguono l'intestazione
- Qualsiasi riempimento vuoto che segue tali parametri

StrucLength deve essere un multiplo di 4.

La parte a lunghezza fissa della struttura è definita da MQEPH_STRUC_LENGTH_FIXED.

Il valore iniziale di questo campo è 68.

Codifica (MQLONG) per MQEPH

Questa è la codifica numerica dei dati che seguono la struttura MQEPH e i parametri PCF associati; non si applica ai dati carattere nella struttura MQEPH stessa.

Il valore iniziale di questo campo è 0.

CodedCharSetId (MQLONG) per MQEPH

Questo è l'identificativo della serie di caratteri dei dati che seguono la struttura MQEPH e i parametri PCF associati; non si applica ai dati carattere nella stessa struttura MQEPH.

Il valore iniziale di questo campo è MQCCSI_UNDEFINED.

Formato (MQCHAR8) per MQEPH

Questo è il nome formato dei dati che seguono la struttura MQEPH e i parametri PCF associati.

Il valore iniziale di questo campo è MQFMT_NONE.

Indicatori (MQLONG) per MQEPH

Sono disponibili i seguenti lavori:

MQEPH_NONE
Non è stato specificato alcun indicatore. MQEPH_NONE è definito per la documentazione del programma. Non è previsto che questa costante venga utilizzata con altre, ma poiché il valore è zero, tale utilizzo non può essere rilevato.

MQEPH_CCSID_EMBEDDED
La serie di caratteri dei parametri contenenti i dati dei caratteri viene specificata singolarmente all'interno del campo CodedCharSetId in ciascuna struttura. La serie di caratteri dei campi StrucId e

Format è definito dal campo CodedCharSetId nella struttura dell'intestazione che precede la struttura MQEPH oppure dal campo CodedCharSetId in MQMD se MQEPH si trova all'inizio del messaggio.

Il valore iniziale di questo campo è MQEPH_NONE.

PCFHeader (MQCFH) per MQEPH

Si tratta dell'intestazione PCF (Programmable Command Format), che definisce i parametri PCF che seguono la struttura MQEPH. Ciò consente di seguire i dati del messaggio PCF con altre intestazioni.

L'intestazione PCF viene definita inizialmente con i valori seguenti:

Tabella 489. Campi in MQCFH

Nome campo	Nome della costante	Valore della costante
Type	MQCFT_NONE	0
StrucLength	LUNGHEZZA_STRUTTURA_MQCFH_STRUCT	36
Version	MQCFH_VERSION_3	3
StrucLength	Nessuna	0
Command	MQCMD_NONE	0
MsgSeqNumber	Nessuna	1
Control	MQCF_LAST	1
CompCode	MQCC_OK	0
Reason	MQRC_NONE	0
ParameterCount	Nessuna	0

L'applicazione deve modificare il Type da MQCFT_NONE in un tipo di struttura valido per l'utilizzo dell'intestazione PCF incorporata.

MQGMO - Opzioni Get - message

La struttura MQGMO consente all'applicazione di controllare la modalità di rimozione dei messaggi dalle code. La struttura è un parametro di input / output nella chiamata MQGET.

Versione

La versione corrente di MQGMO è MQGMO_VERSION_4. Alcuni campi sono disponibili solo in determinate versioni di MQGMO. Se è necessario eseguire il port delle applicazioni tra diversi ambienti, è necessario verificare che la versione di MQGMO sia congruente in tutti gli ambienti. I campi che esistono solo in particolari versioni della struttura sono identificati come tali in “MQGMO - Opzioni Get - message” a pagina 376 e nelle descrizioni dei campi.

I file di intestazione, COPY e INCLUDE forniti per i linguaggi di programmazione supportati contengono la versione più recente di MQGMO supportata dall'ambiente, ma con il valore iniziale del campo *Version* impostato su MQGMO_VERSION_1. Per utilizzare i campi che non sono presenti nella struttura version-1 , impostare il campo *Version* sul numero di versione della versione richiesta.

Serie di caratteri e codifica

I dati in MQGMO devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da MQENC_NATIVE. Tuttavia, se l'applicazione è in esecuzione come client MQ MQI, la struttura deve essere nella serie di caratteri e nella codifica del client.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 490. Campi in MQGMO per MQGMO

Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
StrucId (identificativo della struttura)	ID_STRUC_MQGMO	'GMO¬'
Versione (numero versione struttura)	MQGMO_VERSION_1	1
MQGMO - Campo Opzioni (opzioni che controllano l'azione di MQGET)	MQGMO_NO_WAIT	0
WaitInterval (intervallo di attesa)	Nessuna	0
Signal1 (segnale)	Nessuna	Puntatore null su z/OS ; 0 altrimenti
Signal2 (identificativo del segnale)	Nessuna	0
ResolvedQName (nome risolto della coda di destinazione)	Nessuna	Stringa null o spazi vuoti
Nota: I restanti campi vengono ignorati se *Version* è minore di MQGMO_VERSION_2.		
MatchOptions (opzioni che controllano i criteri di selezione utilizzati per MQGET)	MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID	3
GroupStatus (indicatore che indica se il messaggio richiamato è in un gruppo)	MQGS_NOT_IN_GROUP	'¬'
SegmentStatus (indicatore che indica se il messaggio richiamato è un segmento di un messaggio logico)	MQSS_NOT_A_SEGMENT	'¬'
Segmentazione (indicatore che indica se è consentita un'ulteriore segmentazione per il messaggio richiamato)	MQSEG_INIBITO	'¬'
Reserved1 (riservato)	Nessuna	'¬'
Nota: I restanti campi vengono ignorati se *Version* è minore di MQGMO_VERSION_3.		
MsgToken (token messaggio)	MQMTOK_NONE	Valori null
ReturnedLength (lunghezza in byte dei dati del messaggio restituiti)	MQRL_UNDEFINED	-1
Nota: I restanti campi vengono ignorati se *Version* è minore di MQGMO_VERSION_4.		
Reserved2 (riservato)	Nessuna	'¬'
MsgHandle (handle per un messaggio che deve essere popolato con le proprietà del messaggio richiamato dalla coda)	MQHM_NONE	0

Tabella 490. Campi in MQGMO per MQGMO (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.
3. Nel linguaggio di programmazione C, la variabile macroMQGMO_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura:

```
MQGMO MyGMO = {MQGMO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQGMO

```
typedef struct tagMQGMO MQGMO;
struct tagMQGMO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of */
                             MQGET */
  MQLONG    WaitInterval;    /* Wait interval */
  MQLONG    Signal1;         /* Signal */
  MQLONG    Signal2;         /* Signal identifier */
  MQCHAR48  ResolvedQName;   /* Resolved name of destination queue */
  /* Ver:1 */
  MQLONG    MatchOptions;    /* Options controlling selection */
                             criteria used for MQGET */
  MQCHAR    GroupStatus;     /* Flag indicating whether message */
                             retrieved is in a group */
  MQCHAR    SegmentStatus;   /* Flag indicating whether message */
                             retrieved is a segment of a logical */
                             message */
  MQCHAR    Segmentation;    /* Flag indicating whether further */
                             segmentation is allowed for the */
                             message retrieved */
  MQCHAR    Reserved1;       /* Reserved */
  /* Ver:2 */
  MQBYTE16  MsgToken;        /* Message token */
  MQLONG    ReturnedLength;  /* Length of message data returned */
                             (bytes) */
  /* Ver:3 */
  MQLONG    Reserved2;       /* Reserved */
  MQHMSG    MsgHandle;       /* Message handle */
  /* Ver:4 */
};
```

Nota: Su z/OS, il campo *Signal1* è dichiarato come PMQLONG.

Dichiarazione COBOL per MQGMO

```
**   MQGMO structure
  10 MQGMO.
**    Structure identifier
   15 MQGMO-STRUCID        PIC X(4).
**    Structure version number
   15 MQGMO-VERSION        PIC S9(9) BINARY.
**    Options that control the action of MQGET
   15 MQGMO-OPTIONS        PIC S9(9) BINARY.
**    Wait interval
   15 MQGMO-WAITINTERVAL   PIC S9(9) BINARY.
**    Signal
```

```
   15 MQGMO-SIGNAL1        PIC S9(9) BINARY.
**    Signal identifier
   15 MQGMO-SIGNAL2        PIC S9(9) BINARY.
**    Resolved name of destination queue
   15 MQGMO-RESOLVEDQNAME  PIC X(48).
**    Options controlling selection criteria used for MQGET
   15 MQGMO-MATCHOPTIONS   PIC S9(9) BINARY.
**    Flag indicating whether message retrieved is in a group
   15 MQGMO-GROUPSTATUS    PIC X.
**    Flag indicating whether message retrieved is a segment of a
**    logical message
   15 MQGMO-SEGMENTSTATUS  PIC X.
**    Flag indicating whether further segmentation is allowed for the
**    message retrieved
   15 MQGMO-SEGMENTATION   PIC X.
**    Reserved
   15 MQGMO-RESERVED1      PIC X.
**    Message token
   15 MQGMO-MSGTOKEN       PIC X(16).
**    Length of message data returned (bytes)
   15 MQGMO-RETURNEDLENGTH PIC S9(9) BINARY.
**    Reserved
   15 MQGMO-RESERVED2      PIC S9(9) BINARY.
**    Message handle
   15 MQGMO-MSGHANDLE      PIC S9(18) BINARY.
```

Nota: Su z/OS, il campo *Signal1* è dichiarato come POINTER.

Dichiarazione PL/I per MQGMO

```
dcl
 1 MQGMO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action of
                                     MQGET */
  3 WaitInterval   fixed bin(31), /* Wait interval */
  3 Signal1        fixed bin(31), /* Signal */
  3 Signal2        fixed bin(31), /* Signal identifier */
  3 ResolvedQName  char(48),      /* Resolved name of destination
                                     queue */
  3 MatchOptions   fixed bin(31), /* Options controlling selection
                                     criteria used for MQGET */
  3 GroupStatus    char(1),       /* Flag indicating whether message
                                     retrieved is in a group */
  3 SegmentStatus  char(1),       /* Flag indicating whether message
                                     retrieved is a segment of a logical
                                     message */
  3 Segmentation   char(1),       /* Flag indicating whether further
                                     segmentation is allowed for the
                                     message retrieved */
  3 Reserved1      char(1),       /* Reserved */
  3 MsgToken       char(16),      /* Message token */
  3 ReturnedLength fixed bin(31); /* Length of message data returned
                                     (bytes) */
  3 Reserved2      fixed bin(31); /* Reserved */
  3 MsgHandle      fixed bin(63); /* Message handle */
```

Nota: Su z/OS, il campo *Signal1* è dichiarato come `pointer`.

Dichiarazione High Level Assembler per MQGMO

```
MQGMO                DSECT
MQGMO_STRUCID        DS   CL4   Structure identifier
MQGMO_VERSION        DS   F     Structure version number
MQGMO_OPTIONS        DS   F     Options that control the action of
*                               MQGET
MQGMO_WAITINTERVAL   DS   F     Wait interval
MQGMO_SIGNAL1        DS   F     Signal
MQGMO_SIGNAL2        DS   F     Signal identifier
MQGMO_RESOLVEDQNAME  DS   CL48  Resolved name of destination queue
MQGMO_MATCHOPTIONS   DS   F     Options controlling selection criteria
*                               used for MQGET
MQGMO_GROUPSTATUS    DS   CL1   Flag indicating whether message
*                               retrieved is in a group
MQGMO_SEGMENTSTATUS  DS   CL1   Flag indicating whether message
*                               retrieved is a segment of a logical
```

```
*                                 message
MQGMO_SEGMENTATION    DS   CL1    Flag indicating whether further
*                                 segmentation is allowed for the message
*                                 retrieved
MQGMO_RESERVED1       DS   CL1    Reserved
MQGMO_MSGTOKEN        DS   XL16   Message token
MQGMO_RETURNEDLENGTH  DS   F      Length of message data returned (bytes)
MQGMO_RESERVED2       DS   F      Reserved
MQGMO_MSGHANDLE       DS   D      Message handle
MQGMO_LENGTH          EQU  *-MQGMO
                      ORG  MQGMO
MQGMO_AREA            DS   CL(MQGMO_LENGTH)
```

Dichiarazione High Level Assembler per MQGMO

```
Type MQGMO
  StrucId        As String*4  'Structure identifier'
  Version        As Long      'Structure version number'
  Options        As Long      'Options that control the action of MQGET'
  WaitInterval   As Long      'Wait interval'
  Signal1        As Long      'Signal'
  Signal2        As Long      'Signal identifier'
  ResolvedQName  As String*48 'Resolved name of destination queue'
  MatchOptions   As Long      'Options controlling selection criteria'
                              'used for MQGET'
  GroupStatus    As String*1  'Flag indicating whether message'
                              'retrieved is in a group'
  SegmentStatus  As String*1  'Flag indicating whether message'
                              'retrieved is a segment of a logical'
                              'message'
  Segmentation   As String*1  'Flag indicating whether further'
                              'segmentation is allowed for the message'
                              'retrieved'
  Reserved1      As String*1  'Reserved'
  MsgToken       As MQBYTE16  'Message token'
  ReturnedLength As Long      'Length of message data returned (bytes)'
End Type
```

Opzioni di canale PROPCTL per MQGMO

Utilizzare l'attributo del canale **PROPCTL** per controllare quali proprietà del messaggio sono incluse in un messaggio inviato da un gestore code IBM MQ 9.4 a una versione precedente di IBM MQ.

Tabella 491. Impostazioni attributo proprietà messaggio canale

PROPCTL	Descrizione
TUTTO	Utilizzare questa opzione se le applicazioni connesse al gestore code partner da una versione precedente sono in grado di elaborare tutte le proprietà inserite in un messaggio da un'applicazione IBM MQ 9.4 . Tutte le proprietà vengono inviate al gestore code partner, oltre a tutte le coppie nome - valore collocate in MQRFH2. È necessario considerare due problemi di progettazione dell'applicazione: 1. Un'applicazione connessa a un gestore code partner deve essere in grado di elaborare i messaggi contenenti intestazioni MQRFH2 generate su un gestore code IBM MQ 9.4 . 2. L'applicazione connessa al gestore code partner deve elaborare correttamente le nuove proprietà del messaggio contrassegnate con MQPD_SUPPORT_REQUIRED . Con l'opzione del canale ALL impostata, le applicazioni JMS possono interagire tra IBM MQ 9.4 e una versione precedente utilizzando il canale. Le nuove applicazioni IBM MQ 9.4 che utilizzano le proprietà dei messaggi possono interagire con le applicazioni di una versione precedente, a seconda di come l'applicazione della versione precedente gestisce le intestazioni MQRFH2 .

Tabella 491. Impostazioni attributo proprietà messaggio canale (Continua)	
PROPCTL	**Descrizione**
COMPAT	Utilizzare questa opzione per inviare le proprietà dei messaggi alle applicazioni connesse a un gestore code partner di una versione precedente in alcuni casi, ma non in tutti. Le proprietà del messaggio vengono inviate solo se vengono soddisfatte due condizioni: 1. Nessuna proprietà deve essere contrassegnata come richiedente l'elaborazione della proprietà del messaggio. 2. Almeno una delle proprietà del messaggio deve trovarsi in una cartella "riservata" ; consultare Nota. Con l'opzione del canale COMPAT impostata, le applicazioni JMS possono interagire tra IBM MQ 9.4 e una versione precedente utilizzando il canale. Il canale non è disponibile per tutte le applicazioni che utilizzano le proprietà dei messaggi, ma solo per quelle che utilizzano le cartelle riservate. Le regole relative all'invio del messaggio o della proprietà sono: 1. Se il messaggio dispone di proprietà, ma nessuna delle proprietà è associata a una cartella "riservata" , non viene inviata alcuna proprietà del messaggio. 2. Se una proprietà del messaggio è stata creata in una cartella di proprietà "riservata" , vengono inviate tutte le proprietà del messaggio associate al messaggio. Tuttavia: a. Se una delle proprietà del messaggio è contrassegnata come richiesta di supporto, MQPD_SUPPORT_REQUIRED o MQPD_SUPPORT_REQUIRED_IF_LOCAL, l'intero messaggio viene rifiutato. Viene restituito, eliminato o inviato alla coda di messaggi non recapitabili in base al valore delle relative opzioni di report. b. Se nessuna proprietà del messaggio è contrassegnata come richiesta, una singola proprietà potrebbe non essere inviata. Se uno dei campi del descrittore della proprietà del messaggio è impostato su valori non predefiniti, la singola proprietà non viene inviata. Il messaggio è ancora inviato. Un esempio di un valore di campo descrittore proprietà non predefinito è MQPD_USER_CONTEXT. **Nota:** I nomi delle cartelle "riservate" iniziano con mcd., jms., usr.o mqext.. Queste cartelle vengono create per applicazioni che utilizzano l'interfaccia JMS . In IBM MQ 9.4 tutte le coppie nome - valore inserite in queste cartelle vengono considerate come proprietà del messaggio. Le proprietà del messaggio vengono inviate in un'intestazione MQRFH2 , in aggiunta a tutte le coppie nome - valore inserite in un'intestazione MQRFH2 . Tutte le coppie nome - valore inserite in un'intestazione MQRFH2 vengono inviate finché il messaggio non viene rifiutato.
Nessuno	Utilizzare questa opzione per impedire che le proprietà del messaggio vengano inviate alle applicazioni connesse a un gestore code partner di una versione precedente. Un MQRFH2 che contiene coppie nome - valore e proprietà del messaggio viene ancora inviato, ma solo con le coppie nome - valore. Con l'opzione del canale NONE impostata, un messaggio JMS viene inviato come JMSTextMessage o JMSBytesMessage senza alcuna proprietà del messaggio JMS . Se è possibile per un'applicazione di una versione precedente ignorare tutte le proprietà impostate in un'applicazione IBM MQ 9.4 , può interagire con essa.

PROPCTL opzioni di coda per MQGMO

Utilizzare l'attributo della coda **PROPCTL** per controllare il modo in cui le proprietà del messaggio vengono restituite a un'applicazione che richiama **MQGET** senza impostare alcuna **MQGMO** opzione della proprietà del messaggio.

Tabella 492. Impostazioni attributo proprietà messaggio coda

PROPCTL	Descrizione
TUTTO	Utilizzare l'opzione ALL in modo che diverse applicazioni che leggono un messaggio dalla stessa coda possano elaborare il messaggio in modi diversi. • Un'applicazione, migrata senza modifiche da una precedente versione, può continuare a leggere direttamente MQRFH2 . Le proprietà sono direttamente accessibili nell'intestazione MQRFH2 . È necessario modificare l'applicazione per gestire le nuove proprietà e gli attributi delle nuove proprietà. È possibile che l'applicazione sia influenzata dalle modifiche del layout e del numero di intestazioni MQRFH2 . Alcuni attributi della cartella potrebbero essere rimossi o che IBM MQ riporta un errore nel layout dell'intestazione MQRFH2 che ha ignorato in una versione precedente. • Un'applicazione nuova o modificata può utilizzare la proprietà del messaggio MQI per interrogare le proprietà del messaggio e leggere le coppie nome - valore direttamente nell'intestazione MQRFH2 . Tutte le proprietà nel messaggio vengono restituite all'applicazione. • Se l'applicazione richiama MQCRTMH per creare un handle del messaggio, è necessario interrogare le proprietà del messaggio utilizzando MQINQMP. Le coppie nome - valore che non sono proprietà del messaggio rimangono in MQRFH2, che viene eliminato da tutte le proprietà del messaggio. • Se l'applicazione non crea un handle del messaggio, tutte le proprietà del messaggio e le coppie nome - valore rimangono in MQRFH2. ALL ha questo effetto solo se l'applicazione ricevente non ha impostato un'opzione MQGMO_PROPERTIES o l'ha impostata su MQGMO_PROPERTIES_AS_Q_DEF.

Tabella 492. Impostazioni attributo proprietà messaggio coda (Continua)

PROPCTL	Descrizione
COMPAT (valore predefinito)	COMPAT è l'opzione predefinita. Se GMO_PROPERTIES_* non è impostato, come in un'applicazione non modificata da una versione precedente, si presuppone COMPAT . Utilizzando l'opzione COMPAT , un'applicazione di una versione precedente che non ha creato esplicitamente un MQRFH2, funziona senza modifiche su IBM MQ 9.4. Utilizzare questa opzione se è stata scritta un'applicazione MQI di una versione precedente per leggere i messaggi JMS . • Le proprietà JMS , memorizzate in un'intestazione MQRFH2 , vengono restituite all'applicazione in un'intestazione MQRFH2 in cartelle con nomi che iniziano con mcd., jms., usr.o mqext. • Se il messaggio dispone di cartelle JMS e se un'applicazione IBM MQ 9.4 aggiunge nuove cartelle di proprietà al messaggio, tali proprietà vengono restituite anche in MQRFH2. Di conseguenza, è necessario modificare l'applicazione per gestire le nuove proprietà e gli attributi delle nuove proprietà. È possibile che un'applicazione non modificata possa essere influenzata da modifiche nel layout e nel numero di intestazioni MQRFH2 . È possibile che alcuni attributi della cartella vengano rimossi o che IBM MQ rilevi errori nel layout dell'intestazione MQRFH2 che ha ignorato in una versione precedente. **Nota:** In questo scenario, il comportamento dell'applicazione è lo stesso se è connessa a una versione precedente o a un gestore code IBM MQ 9.4 . Se l'attributo **PROPCTL** del canale è impostato su COMPAT o ALL , le nuove proprietà del messaggio vengono inviate nel messaggio al gestore code partner della versione precedente. • Se il messaggio non è un messaggio JMS , ma contiene altre proprietà, tali proprietà non vengono restituite all'applicazione in un'intestazione MQRFH2 .[1] • L'opzione consente inoltre alle applicazioni di versioni precedenti che creano esplicitamente un MQRFH2 di funzionare correttamente, in molti casi. Ad esempio, un programma MQI che crea un MQRFH2 contenente JMS proprietà del messaggio continua a funzionare correttamente. Se un messaggio viene creato senza le proprietà del messaggio JMS , ma con altre cartelle MQRFH2 , le cartelle vengono restituite all'applicazione. Solo se le cartelle sono cartelle di proprietà dei messaggi, tali cartelle specifiche vengono rimosse da MQRFH2. Le cartelle delle proprietà del messaggio sono identificate dall'attributo della nuova cartella content='properties' oppure sono cartelle con nomi elencati in Nome cartella proprietà definita o Nome cartella proprietà non raggruppata. • Se l'applicazione richiama MQCRTMH per creare un handle del messaggio, è necessario interrogare le proprietà del messaggio utilizzando MQINQMP. Le proprietà del messaggio vengono rimosse dalle intestazioni MQRFH2 . Le coppie nome - valore che non sono proprietà del messaggio rimangono in MQRFH2. • Se l'applicazione richiama MQCRTMH per creare un handle del messaggio, può eseguire la query di tutte le proprietà del messaggio, indipendentemente dal fatto che il messaggio abbia o meno cartelle JMS . • Se l'applicazione non crea un handle del messaggio, tutte le proprietà del messaggio e le coppie nome - valore rimangono in MQRFH2. Se un messaggio contiene nuove cartelle di proprietà utente, è possibile dedurre che il messaggio è stato creato da un'applicazione IBM MQ 9.4 nuova o modificata. Se l'applicazione ricevente deve elaborare queste nuove proprietà direttamente in un MQRFH2, è necessario modificare l'applicazione per utilizzare l'opzione ALL . Con l'opzione predefinita COMPAT impostata, un'applicazione non modificata continua ad elaborare il resto del file MQRFH2, senza le proprietà IBM MQ 9.4 . Lo scopo dell'interfaccia PROPCTL è supportare le vecchie applicazioni che leggono le cartelle MQRFH2 e le applicazioni nuove e modificate utilizzando l'interfaccia delle proprietà del messaggio. Puntare alle nuove applicazioni per utilizzare l'interfaccia delle proprietà del messaggio per tutte le proprietà del messaggio utente ed evitare di leggere o scrivere direttamente le intestazioni MQRFH2 . COMPAT [illegible]

Tabella 492. Impostazioni attributo proprietà messaggio coda (Continua)	
PROPCTL	**Descrizione**
Forza	L'opzione FORCE inserisce tutte le proprietà dei messaggi nelle intestazioni MQRFH2 . Tutte le proprietà del messaggio e le coppie nome - valore nelle intestazioni MQRFH2 rimangono nel messaggio. Le proprietà del messaggio non vengono rimosse da MQRFH2e rese disponibili tramite un handle del messaggio. L'effetto della scelta dell'opzione FORCE è quello di consentire a un'applicazione appena migrata di leggere le proprietà del messaggio dalle intestazioni MQRFH2 . Si supponga di aver modificato un'applicazione per elaborare le proprietà del messaggio IBM MQ 9.4 , ma di aver anche conservato la capacità di lavorare direttamente con le intestazioni MQRFH2 , come in precedenza. È possibile decidere quando passare dall'applicazione all'utilizzo delle proprietà del messaggio impostando inizialmente l'attributo della coda PROPCTL su FORCE. Impostare l'attributo della coda **PROPCTL** su un altro valore quando si è pronti ad utilizzare le proprietà del messaggio. Se la nuova funzione nell'applicazione non funziona come previsto, impostare nuovamente l'opzione **PROPCTL** su FORCE. FORCE ha effetto solo se l'applicazione ricevente non ha impostato un'opzione MQGMO_PROPERTIES o l'ha impostata su MQGMO_PROPERTIES_AS_Q_DEF.
Nessuno	Utilizzare l'opzione NONE in modo che un'applicazione esistente possa elaborare un messaggio, ignorando tutte le proprietà del messaggio e che un'applicazione nuova o modificata possa eseguire la query delle proprietà del messaggio. • Se l'applicazione richiama MQCRTMH per creare un handle del messaggio, è necessario interrogare le proprietà del messaggio utilizzando MQINQMP. Le coppie nome - valore che non sono proprietà del messaggio rimangono in MQRFH2, che viene eliminato da tutte le proprietà del messaggio. • Se l'applicazione non crea un handle del messaggio, tutte le proprietà del messaggio vengono rimosse da MQRFH2. Le coppie nome - valore nelle intestazioni MQRFH2 rimangono nel messaggio. NONE ha questo effetto solo se l'applicazione ricevente non ha impostato un'opzione MQGMO_PROPERTIES o l'ha impostata su MQGMO_PROPERTIES_AS_Q_DEF.
V6COMPAT	Utilizzare questa opzione per ricevere un MQRFH2 nello stesso formato in cui è stato inviato. Se l'applicazione di invio, o il gestore code, crea ulteriori proprietà del messaggio, queste vengono restituite nell'handle del messaggio. Questa opzione deve essere impostata sia sulle code di invio e di ricezione che sulle code di trasmissione intermedie. Sostituisce qualsiasi altra opzione PROPCTL impostata sulle definizioni della coda nel percorso di risoluzione del nome della coda. Utilizzare l'opzione V6COMPAT solo in circostanze eccezionali. Ad esempio, se si stanno migrando le applicazioni da una versione precedente a IBM MQ 9.4, l'opzione è utile perché preserva il funzionamento della versione precedente. L'opzione potrebbe avere un impatto sulla velocità di trasmissione dei messaggi. È anche più difficile da amministrare; è necessario assicurarsi che l'opzione sia impostata sulle code di trasmissione del mittente, del destinatario e dell'intervento. V6COMPAT ha questo effetto solo se l'applicazione ricevente non ha impostato un'opzione MQGMO_PROPERTIES o l'ha impostata su MQGMO_PROPERTIES_AS_Q_DEF.

[1] L'esistenza di specifiche cartelle di proprietà create da IBM MQ classes for JMS indica un JMS messaggio. Le cartelle delle proprietà sono mcd., jms., usr.o mqext.

Per ulteriori informazioni sulle proprietà del messaggio e sulle coppie nome - valore, consultare “NameValueDati (MQCHARn) per MQRFH2” a pagina 547.

Opzioni delle proprietà dei messaggi per MQGMO

Utilizzare le opzioni della proprietà del messaggio **MQGMO** per controllare in che modo le proprietà del messaggio vengono restituite a un'applicazione.

Tabella 493. Impostazioni delle opzioni delle proprietà dei messaggi MQGMO

MQGMO Opzione	**Descrizione**
MQGMO_PROPERTIES_AS_Q_DEF	Le applicazioni IBM MQ che leggono dalla stessa coda e non impostano GMO_PROPERTIES_*, ricevono le proprietà del messaggio in modo diverso. Le applicazioni IBM MQ che non creano un handle del messaggio, sono controllate dall'attributo **PROPCTL** della coda. Un'applicazione IBM MQ può scegliere di ricevere le proprietà del messaggio in MQRFH2oppure creare un handle del messaggio ed eseguire la query delle proprietà del messaggio. Se l'applicazione crea un handle del messaggio, le proprietà vengono rimosse da MQRFH2. • Un'applicazione IBM MQ nuova o modificata che non imposta GMO_PROPERTIES_* o la imposta su MQGMO_PROPERTIES_AS_Q_DEF può scegliere di interrogare le proprietà del messaggio. È necessario impostare MQCRTMH per creare un handle del messaggio e le proprietà del messaggio di query utilizzando la chiamata MQI MQINQMP . • Se un'applicazione nuova o modificata non crea un handle del messaggio, deve leggere tutte le proprietà del messaggio che riceve direttamente dalle intestazioni MQRFH2 . • Se l'attributo della coda **PROPCTL** è impostato su FORCE, non viene restituita alcuna proprietà nell'handle del messaggio. Tutte le proprietà vengono restituite nelle intestazioni MQRFH2 . • Se l'attributo della coda **PROPCTL** è impostato su NONEo COMPAT, un'applicazione IBM MQ che crea un handle del messaggio, riceve tutte le proprietà del messaggio.
MQGMO_PROPERTIES_IN_HANDLE	Forza un'applicazione a utilizzare le proprietà del messaggio. Utilizzare questa opzione per rilevare se un'applicazione modificata non riesce a creare l'handle del messaggio. L'applicazione potrebbe tentare di leggere le proprietà del messaggio direttamente da un MQRFH2, piuttosto che richiamare MQINQMP.
MQGMO_NO_PROPERTIES	• Tutte le proprietà vengono rimosse.. Le proprietà generate dal gestore code, come le proprietà JMS , vengono rimosse. • Le proprietà vengono rimosse anche se viene creato un handle del messaggio. Le coppie nome - valore in altre cartelle MQRFH2 sono disponibili nei dati del messaggio.

Tabella 493. Impostazioni delle opzioni delle proprietà dei messaggi MQGMO (Continua)	
MQGMO Opzione	**Descrizione**
MQGMO_PROPERTIES_FORCE_MQRFH2	Le proprietà vengono restituite nelle intestazioni MQRFH2 , anche se viene creato un handle del messaggio. • MQINQMP non restituisce alcuna proprietà del messaggio, anche se viene creato un handle del messaggio. MQRC_PROPERTY_NOT_AVAILABLE viene restituito se una proprietà viene interrogata.
MQGMO_PROPERTIES_COMPATIBILITY	Se il messaggio proviene da un client JMS , le proprietà JMS vengono restituite nelle intestazioni MQRFH2 . Le applicazioni IBM MQ nuove o modificate, che creano un handle del messaggio, si comportano in modo diverso. • Tutte le proprietà nelle cartelle delle proprietà del messaggio vengono restituite se il messaggio contiene una cartella mcd., jms., usr.o mqext . • Se il messaggio contiene cartelle delle proprietà, ma non una cartella mcd., jms., usr.o mqext , non viene restituita alcuna proprietà del messaggio in un MQRFH2. • Se un'applicazione IBM MQ nuova o modificata crea un handle del messaggio, eseguire la query delle proprietà del messaggio utilizzando la chiamata MQI MQINQMP . Tutte le proprietà del messaggio vengono rimosse da MQRFH2. • Se un'applicazione IBM MQ nuova o modificata crea un handle del messaggio, è possibile eseguire la query di tutte le proprietà nel messaggio. Anche se il messaggio non contiene una cartella mcd., jms., usr.o mqext , tutte le proprietà del messaggio sono interrogabili.

Riferimenti correlati
PROPCTL
2471 (09A7) (RC2471): MQRC_PROPERTY_NOT_AVAILABLE

StrucId (MQCHAR4) per MQGMO

Questo è l'identificativo della struttura delle opzioni di richiamo del messaggio. È sempre un campo di immissione. Il valore è MQGMO_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQGMO
Identificativo per la struttura delle opzioni di richiamo del messaggio.

Per il linguaggio di programmazione C, viene definita anche la costante MQGMO_STRUC_ID_ARRAY. Ha lo stesso valore di MQGMO_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQGMO

Versione è il numero di versione della struttura.

Il valore deve essere uno dei seguenti.

MQGMO_VERSION_1
Struttura delle opzioni Version-1 get - message.

Questa versione è supportata in tutti gli ambienti.

MQGMO_VERSION_2
Struttura delle opzioni Version-2 get - message.

Questa versione è supportata in tutti gli ambienti.

MQGMO_VERSION_3
Version-3 struttura delle opzioni get - message.

Questa versione è supportata in tutti gli ambienti.

MQGMO_VERSION_4
Struttura delle opzioni Version-4 get - message.

Questa versione è supportata in tutti gli ambienti.

I campi che esistono solo nelle versioni più recenti della struttura vengono identificati come tali nelle descrizioni dei campi. La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQGMO_CURRENT_
Versione corrente della struttura delle opzioni get - message.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQGMO_VERSION_1.

Opzioni (MQLONG) per MQGMO

Le opzioni **MQGMO** controllano l'azione di MQGET. È possibile specificare zero o più opzioni. Se è necessario più di un valore facoltativo:

- Aggiungere i valori (non aggiungere la stessa costante più di una volta) oppure
- Combinare i valori utilizzando l'operazione OR bitwise (se il linguaggio di programmazione supporta le operazioni bit).

Vengono annotate le combinazioni di opzioni non valide; tutte le altre combinazioni sono valide.

Opzioni di attesa

Le seguenti opzioni sono relative all'attesa dell'arrivo dei messaggi sulla coda:

MQGMO_WAIT
L'applicazione attende l'arrivo di un messaggio adatto. Il tempo massimo di attesa dell'applicazione è specificato in *WaitInterval*.

Importante: Non c'è attesa, o ritardo, se un messaggio adatto è disponibile immediatamente.

Se le richieste MQGET sono inibite o MQGET diventano inibite durante l'attesa, l'attesa viene annullata. La chiamata viene completata con MQCC_FAILED e il codice motivo MQRC_GET_INHIBITED, indipendentemente dal fatto che ci siano messaggi adatti sulla coda.

È possibile utilizzare MQGMO_WAIT con le opzioni MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT .

Se diverse applicazioni sono in attesa sulla stessa coda condivisa, le seguenti regole selezionano quale applicazione viene attivata quando arriva un messaggio appropriato:

Tabella 494. Regole per l'attivazione delle chiamate MQGET su una coda condivisa.

Numero di chiamate MQGET in attesa di attivazione		**Risultato**
Con un'opzione BROWSE	**Senza un'opzione BROWSE**[2]	
Nessuna	Uno o più	Viene attivata una chiamata MQGET senza un'opzione BROWSE .
Uno o più	Nessuna	Tutte le chiamate MQGET con opzione BROWSE vengono attivate.

[2] Una chiamata MQGET che specifichi l'opzione MQGMO_LOCK viene considerata come una chiamata non browse.

Tabella 494. Regole per l'attivazione delle chiamate MQGET su una coda condivisa. (Continua)		
Numero di chiamate MQGET in attesa di attivazione		**Risultato**
Con un'opzione BROWSE	**Senza un'opzione BROWSE[2]**	
Uno o più	Uno o più	Viene attivata una chiamata MQGET senza un'opzione BROWSE . Il numero di chiamate MQGET con un'opzione BROWSE attivate è imprevedibile.

Se più di una chiamata MQGET senza l'opzione BROWSE è in attesa sulla stessa coda, ne viene attivata solo una. Il gestore code tenta di dare priorità alle chiamate in attesa nel seguente ordine:

1. Richieste get - wait specifiche che possono essere soddisfatte solo da determinati messaggi, ad esempio, quelli con uno specifico MsgId o CorrelId (o entrambi).
2. Richieste get - wait generali che possono essere soddisfatte da qualsiasi messaggio.

Nota:

- All'interno della prima categoria, non viene assegnata alcuna priorità aggiuntiva alle richieste get - wait più specifiche. Ad esempio, richieste che specifichino sia MsgId che CorrelId .
- All'interno di entrambe le categorie, non è possibile prevedere quale applicazione è selezionata. In particolare, l'applicazione in attesa più lunga non è necessariamente quella selezionata.
- La lunghezza del percorso e le considerazioni sulla pianificazione della priorità del sistema operativo possono indicare che un'applicazione in attesa con priorità del sistema operativo inferiore a quella prevista richiama il messaggio.
- Può anche accadere che un'applicazione che non è in attesa recuperi il messaggio piuttosto che uno.

z/OS In z/OS, si applicano i punti seguenti:

- Se si desidera che l'applicazione proceda con un altro lavoro in attesa dell'arrivo del messaggio, utilizzare invece l'opzione di segnale (MQGMO_SET_SIGNAL). Tuttavia, l'opzione di segnale è specifica dell'ambiente; le applicazioni da trasferire tra ambienti differenti non devono utilizzarla.
- Se c'è più di una chiamata MQGET in attesa dello stesso messaggio, con una combinazione di opzioni di attesa e segnale, ogni chiamata in attesa viene considerata allo stesso modo. È un errore specificare MQGMO_SET_SIGNAL con MQGMO_WAIT. È anche un errore specificare questa opzione con un handle di coda per cui un segnale è in sospeso.
- Se si specifica MQGMO_WAIT o MQGMO_SET_SIGNAL per una coda che ha un IndexType di MQIT_MSG_TOKEN, non è consentito alcun criterio di selezione. Questo vuol dire che:
  - Se si utilizza una version-1 MQGMO, impostare i campi MsgId e CorrelId in MQMD specificato nella chiamata MQGET a MQMI_NONE e MQCI_NONE.
  - Se si utilizza una version-2 o successiva MQGMO, impostare il campo MatchOptions su MQMO_NONE.
- Per una chiamata MQGET su una coda condivisa e la chiamata è una richiesta di ricerca o una ricezione distruttiva di un messaggio di gruppo e non è necessario che MsgId né CorrelId corrispondano, il segnale ECB viene inviato MQEC_MSG_ARRIVED dopo 200 millisecondi.

  Ciò si verifica, anche se un messaggio adatto potrebbe non essere arrivato sulla coda, fino a quando l'intervallo di attesa non è scaduto, quando la coda viene inviata con MQEC_WAIT_INTERVAL_EXPIRED. Quando viene inviato MQEC_MSG_ARRIVED, è necessario emettere nuovamente una seconda chiamata MQGET per richiamare il messaggio, se disponibile.

[2] Una chiamata MQGET che specifichi l'opzione MQGMO_LOCK viene considerata come una chiamata non browse.

[2] Una chiamata MQGET che specifichi l'opzione MQGMO_LOCK viene considerata come una chiamata non browse.

Questa tecnica viene utilizzata per garantire che l'utente venga informato in modo tempestivo dell'arrivo di un messaggio, ma può apparire come un sovraccarico di elaborazione imprevisto quando viene confrontato con una sequenza di chiamata simile su una coda non condivisa.

MQGMO_WAIT viene ignorato se specificato con MQGMO_BROWSE_MSG_UNDER_CURSOR o MQGMO_MSG_UNDER_CURSOR ; non viene generato alcun errore.

MQGMO_NO_WAIT
L'applicazione non attende se non è disponibile alcun messaggio adatto. MQGMO_NO_WAIT è l'opposto di MQGMO_WAIT. MQGMO_NO_WAIT è definito per aiutare la documentazione del programma. È il valore predefinito se non viene specificato nessuno dei due.

MQGMO_SET_SIGNAL
Utilizzare questa opzione con i campi Signal1 e Signal2 . Consente alle applicazioni di procedere con un altro lavoro in attesa dell'arrivo di un messaggio. Consente inoltre (se sono disponibili funzioni del sistema operativo adatte) alle applicazioni di attendere i messaggi in arrivo su più di una coda.

Nota: L'opzione MQGMO_SET_SIGNAL è specifica dell'ambiente; non utilizzarla per le applicazioni che si desidera trasferire.

In due circostanze, la chiamata viene completata come se questa opzione non fosse stata specificata:

1. Se un messaggio attualmente disponibile soddisfa i criteri specificati nel descrittore del messaggio.
2. Se viene rilevato un errore di parametro o un altro errore sincrono.

Se nessun messaggio che soddisfa i criteri specificati nel descrittore del messaggio è attualmente disponibile, il controllo ritorna all'applicazione senza attendere l'arrivo di un messaggio. I parametri **CompCode** e **Reason** sono impostati su MQCC_WARNING e MQRC_SIGNAL_REQUEST_ACCEPTED. Non vengono impostati altri campi di output nel descrittore del messaggio e i parametri di output della chiamata MQGET . Quando un messaggio appropriato arriva più tardi, il segnale viene inviato inviando la BCE.

Il chiamante deve quindi emettere nuovamente la chiamata MQGET per recuperare il messaggio. L'applicazione può attendere questo segnale, utilizzando le funzioni fornite dal sistema operativo.

Se il sistema operativo fornisce un meccanismo di attesa multiplo, è possibile utilizzarlo per attendere l'arrivo di un messaggio su una delle diverse code.

Se viene specificato un WaitInterval diverso da zero, il segnale viene consegnato dopo la scadenza dell'intervallo di attesa. Il Gestore code può anche annullare l'attesa, nel qual caso il segnale viene recapitato.

Più di una chiamata MQGET può impostare un segnale per lo stesso messaggio. L'ordine in cui le applicazioni vengono attivate è lo stesso descritto per MQGMO_WAIT.

Se più di una chiamata MQGET è in attesa dello stesso messaggio, ogni chiamata in attesa viene considerata allo stesso modo. Le chiamate possono includere una combinazione di opzioni di attesa e segnale.

In determinate circostanze, la chiamata MQGET può richiamare un messaggio e può essere consegnato un segnale risultante dall'arrivo dello stesso messaggio. Quando un segnale viene consegnato, un'applicazione deve essere preparata per non rendere disponibile alcun messaggio.

Un gestore code non può avere più di una richiesta di segnale in sospeso.

Questa opzione non è valida con nessuna delle seguenti opzioni:

- MQGMO_UNLOCK
- MQGMO_WAIT

Per una chiamata MQGET su una coda condivisa e la chiamata è una richiesta di ricerca o un richiamo distruttivo di un messaggio di gruppo, e non è necessario associare né MsgId né CorrelId , il segnale dell'utente ECB viene inviato MQEC_MSG_ARRIVED dopo 200 millisecondi.

Ciò si verifica, anche se un messaggio adatto potrebbe non essere arrivato sulla coda, fino a quando l'intervallo di attesa non è scaduto, quando la coda viene inviata con MQEC_WAIT_INTERVAL_EXPIRED. Quando MQEC_MSG_ARRIVED viene pubblicato, è necessario emettere nuovamente una seconda chiamata MQGET per richiamare il messaggio, se disponibile.

Questa tecnica viene utilizzata per garantire che l'utente venga informato in modo tempestivo dell'arrivo di un messaggio, ma può apparire come un sovraccarico di elaborazione imprevisto quando viene confrontato con una sequenza di chiamata simile su una coda non condivisa.

Questo non è un metodo efficiente di richiamo dei messaggi quando i messaggi vengono aggiunti raramente. Per evitare questo sovraccarico per il caso di esplorazione, specificare MsgId (se non indicizzato o indicizzato da MsgId) o *CorrelId* (se indicizzato da CorrelId) corrispondente sulla chiamata MQGET .

z/OS Questa opzione è supportata solo su z/OS .

MQGMO_FAIL_IF_QUIESCING
Forzare l'esito negativo della chiamata MQGET se il gestore code è in stato di inattività.

z/OS In z/OS, questa opzione forza anche l'esito negativo della chiamata MQGET se la connessione (per un'applicazione CICS o IMS) si trova nello stato di inattività.

Se questa opzione è specificata con MQGMO_WAIT o MQGMO_SET_SIGNALe l'attesa o il segnale è in sospeso nel momento in cui il gestore code entra nello stato di sospensione:

- L'attesa viene annullata e la chiamata restituisce il codice di completamento MQCC_FAILED con codice motivo MQRC_Q_MGR_QUIESCING o MQRC_CONNECTION_QUIESCING.
- Il segnale viene annullato con un codice di completamento del segnale specifico dell'ambiente.

  z/OS Su z/OS, il segnale viene completato con il codice di completamento evento MQEC_Q_MGR_QUIESCING o MQEC_CONNECTION_QUIESCING.

Se MQGMO_FAIL_IF_QUIESCING non viene specificato e il gestore code o la connessione entra nello stato di sospensione, l'attesa o il segnale non vengono annullati.

Opzioni del punto di sincronizzazione

Le seguenti opzioni si riferiscono alla partecipazione della chiamata MQGET all'interno di un'unità di lavoro:

SYNCPOINT MQGMO
La richiesta è di operare all'interno dei normali protocolli di unità di lavoro. Il messaggio è contrassegnato come non disponibile per altre applicazioni, ma viene eliminato dalla coda solo quando viene eseguito il commit dell'unità di lavoro. Il messaggio viene reso nuovamente disponibile se viene eseguito il backout dell'unità di lavoro.

È possibile lasciare MQGMO_SYNCPOINT e MQGMO_NO_SYNCPOINT non impostati. In tal caso, l'inclusione della richiesta get nei protocolli UOW è determinata dall'ambiente che esegue il gestore code. Non è determinato dall'ambiente che esegue l'applicazione.

- z/OS Su z/OS, la richiesta get si trova all'interno di un'unità di lavoro.
- In tutti gli ambienti ad eccezione di z/OS, la richiesta get non si trova all'interno di un'unità di lavoro.

A causa di queste differenze, un'applicazione che si desidera trasferire non deve consentire l'impostazione predefinita di questa opzione; specificare esplicitamente MQGMO_SYNCPOINT o MQGMO_NO_SYNCPOINT .

Questa opzione non è valida con nessuna delle seguenti opzioni:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

- MQGMO_LOCK
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

MQGMO_SYNCPOINT_IF_PERSISTENT
La richiesta è di operare all'interno dei normali protocolli dell'unità di lavoro, ma solo se il messaggio richiamato è persistente. Un messaggio persistente ha il valore MQPER_PERSISTENT nel campo Persistence in MQMD.

- Se il messaggio è persistente, il gestore code elabora la chiamata come se l'applicazione avesse specificato MQGMO_SYNCPOINT.
- Se il messaggio non è persistente, il gestore code elabora la chiamata come se l'applicazione avesse specificato MQGMO_NO_SYNCPOINT.

Questa opzione non è valida con nessuna delle seguenti opzioni:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_COMPLETE_MSG
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT
- MQGMO_UNLOCK

Questa opzione è supportata nei seguenti ambienti:

- AIX
- IBM i
- Linux
- z/OS

e per IBM MQ MQI clients collegati a questi sistemi.

MQGMO_NO_SYNCPOINT
La richiesta è di operare al di fuori dei normali protocolli di unità di lavoro. Se si riceve un messaggio senza un'opzione di ricerca, viene eliminato immediatamente dalla coda. Il messaggio non può essere reso nuovamente disponibile eseguendo il backout dell'unità di lavoro.

Questa opzione viene assunta se si specifica MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT.

È possibile lasciare MQGMO_SYNCPOINT e MQGMO_NO_SYNCPOINT non impostati. In tal caso, l'inclusione della richiesta get nei protocolli UOW è determinata dall'ambiente che esegue il gestore code. Non è determinato dall'ambiente che esegue l'applicazione.

- z/OS Su z/OS, la richiesta get si trova all'interno di un'unità di lavoro.
- In tutti gli ambienti ad eccezione di z/OS, la richiesta get non si trova all'interno di un'unità di lavoro.

A causa di queste differenze, un'applicazione che si desidera trasferire non deve consentire l'impostazione predefinita di questa opzione; specificare esplicitamente MQGMO_SYNCPOINT o MQGMO_NO_SYNCPOINT .

Questa opzione non è valida con nessuna delle seguenti opzioni:

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT

- MQGMO_SYNCPOINT_IF_PERSISTENT

z/OS **MQGMO_MARK_SKIP_BACKOUT**

Ripristinare un'unità di lavoro senza ripristinare sulla coda il messaggio contrassegnato con questa opzione.

Questa opzione è supportata solo su z/OS.

Se questa opzione è specificata, è necessario specificare anche MQGMO_SYNCPOINT . MQGMO_MARK_SKIP_BACKOUT non è valido con nessuna delle seguenti opzioni:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_LOCK
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Nota: Su IMS e CICS, potrebbe essere necessario emettere una chiamata IBM MQ extran dopo aver eseguito il backout di un'unità di lavoro contenente un messaggio contrassegnato con MQGMO_MARK_SKIP_BACKOUT. È necessario emettere una chiamata IBM MQ prima di eseguire il commit della nuova unità di lavoro contenente il messaggio contrassegnato. La chiamata può essere qualsiasi chiamata IBM MQ che ti piace.

1. Su IMS, se non hai applicato IMS APAR PN60855 e stai eseguendo un'applicazione IMS MPP o BMP.
2. Su CICS, se si sta eseguendo un'applicazione.

In entrambi i casi, emettere una chiamata IBM MQ prima di eseguire il commit della nuova unità di lavoro contenente il messaggio di cui è stato eseguito il backout.

Nota: All'interno di un'unità di lavoro, può esserci solo una richiesta get contrassegnata come backout ignorato, così come nessuna o diverse richieste get non contrassegnate.

Se un'applicazione esegue il backout di un'unità di lavoro, un messaggio richiamato utilizzando MQGMO_MARK_SKIP_BACKOUT non viene ripristinato allo stato precedente. Viene eseguito il backout degli altri aggiornamenti delle risorse. Il messaggio viene considerato come se fosse stato richiamato in una nuova unità di lavoro avviata dalla richiesta di backout. Il messaggio viene richiamato senza l'opzione MQGMO_MARK_SKIP_BACKOUT .

MQGMO_MARK_SKIP_BACKOUT è utile se, dopo la modifica di alcune risorse, risulta evidente che l'unità di lavoro non può essere completata correttamente. Se si omette questa opzione, il backout dell'unità di lavoro reinstalla il messaggio sulla coda. La stessa sequenza di eventi si verifica di nuovo, quando il messaggio viene successivamente richiamato.

Tuttavia, se si specifica MQGMO_MARK_SKIP_BACKOUT sulla chiamata MQGET originale, il backout dell'unità di lavoro esegue il backout degli aggiornamenti alle altre risorse. Il messaggio viene considerato come se fosse stato richiamato sotto una nuova unità di lavoro. L'applicazione può eseguire la gestione errori appropriata. Può inviare un messaggio di report al mittente del messaggio originale o posizionare il messaggio originale nella coda di messaggi non recapitabili. Può quindi eseguire il commit della nuova unità di lavoro. Il commit della nuova unità di lavoro rimuove definitivamente il messaggio dalla coda originale.

MQGMO_MARK_SKIP_BACKOUT contrassegna un singolo messaggio fisico. Se il messaggio appartiene a un gruppo di messaggi, gli altri messaggi nel gruppo non vengono contrassegnati. Allo stesso modo, se il messaggio contrassegnato è un segmento di un messaggio logico, gli altri segmenti nel messaggio logico non vengono contrassegnati.

Qualsiasi messaggio in un gruppo può essere contrassegnato, ma se i messaggi vengono richiamati utilizzando MQGMO_LOGICAL_ORDER, è vantaggioso contrassegnare il primo messaggio nel gruppo.

Se viene eseguito il backout dell'unità di lavoro, il primo messaggio (contrassegnato) viene spostato nella nuova unità di lavoro. Il secondo e i successivi messaggi nel gruppo vengono reintegrati nella coda. I messaggi lasciati sulla coda non possono essere richiamati da un'altra applicazione utilizzando MQGMO_LOGICAL_ORDER. Il primo messaggio nel gruppo non è più nella coda. Tuttavia, l'applicazione che ha eseguito il backup dell'unità di lavoro può recuperare il secondo e i successivi messaggi nella nuova unità di lavoro utilizzando l'opzione MQGMO_LOGICAL_ORDER . Il primo messaggio è già stato richiamato.

Occasionalmente potrebbe essere necessario eseguire il backout della nuova unità di lavoro. Ad esempio, perché la coda di messaggi non recapitabili è piena e il messaggio non deve essere eliminato. Il ripristino della nuova unità di lavoro ripristina il messaggio sulla coda originale, impedendo la perdita del messaggio. Tuttavia, in questa situazione l'elaborazione non può continuare. Dopo aver eseguito il backout della nuova unità di lavoro, l'applicazione deve informare l'operatore o l'amministratore che si è verificato un errore irreversibile e quindi terminare.

MQGMO_MARK_SKIP_BACKOUT funziona solo se l'unità di lavoro che contiene la richiesta get viene interrotta dall'applicazione che ne esegue il backout. Se viene eseguito il backout dell'unità di lavoro contenente la richiesta get perché la transazione o il sistema ha esito negativo, MQGMO_MARK_SKIP_BACKOUT viene ignorato. Qualsiasi messaggio richiamato utilizzando questa opzione viene reintegrato nella coda nello stesso modo dei messaggi richiamati senza questa opzione.

Opzioni Sfoglia

Le seguenti opzioni sono relative alla visualizzazione dei messaggi sulla coda:

MQGMO_BROWSE_FIRST
Quando una coda viene aperta con l'opzione MQOO_BROWSE, viene stabilito un cursore di ricerca, posizionato logicamente prima del primo messaggio sulla coda. È quindi possibile utilizzare le chiamate MQGET specificando l'opzione MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXTo MQGMO_BROWSE_MSG_UNDER_CURSOR per richiamare i messaggi dalla coda in modo non distruttivo. Il cursore di esplorazione contrassegna la posizione, all'interno dei messaggi sulla coda, da cui la successiva chiamata MQGET con MQGMO_BROWSE_NEXT ricerca un messaggio adatto.

MQGMO_BROWSE_FIRST non è valido con nessuna delle seguenti opzioni:

- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

È anche un errore se la coda non è stata aperta per la ricerca.

Una chiamata MQGET con MQGMO_BROWSE_FIRST ignora la precedente posizione del cursore di ricerca. Viene richiamato il primo messaggio sulla coda che soddisfa le condizioni specificate nel descrittore del messaggio. Il messaggio rimane nella coda e il cursore di ricerca è posizionato su questo messaggio.

Dopo questa chiamata, il cursore di ricerca viene posizionato sul messaggio che è stato restituito. Il messaggio potrebbe essere rimosso dalla coda prima che venga emessa la successiva chiamata MQGET con MQGMO_BROWSE_NEXT . In questo caso, il cursore di ricerca rimane nella posizione nella coda occupata dal messaggio, anche se tale posizione è ora vuota.

Utilizzare l'opzione MQGMO_MSG_UNDER_CURSOR con una chiamata MQGET non - browse per rimuovere il messaggio dalla coda.

Il cursore di ricerca non viene spostato da una chiamata MQGET non di ricerca, anche se si utilizza lo stesso handle *Hobj* . Inoltre, non viene spostato da una chiamata MQGET di ricerca che restituisce un codice di completamento di MQCC_FAILEDo un codice motivo di MQRC_TRUNCATED_MSG_FAILED.

Specificare l'opzione MQGMO_LOCK con questa opzione, per bloccare il messaggio visualizzato.

È possibile specificare MQGMO_BROWSE_FIRST con qualsiasi combinazione valida delle opzioni MQGMO_* e MQMO_* che controllano l'elaborazione dei messaggi in gruppi e segmenti di messaggi logici.

Se si specifica MQGMO_LOGICAL_ORDER, i messaggi vengono esaminati in ordine logico. Se si omette tale opzione, i messaggi vengono visualizzati in ordine fisico. Se si specifica MQGMO_BROWSE_FIRST, è possibile passare tra ordine logico e ordine fisico. Le chiamate MQGET successive che utilizzano MQGMO_BROWSE_NEXT sfogliano la coda nello stesso ordine della chiamata più recente che ha specificato MQGMO_BROWSE_FIRST per l'handle di coda.

Il gestore code conserva due serie di informazioni di gruppi e segmenti per le chiamate MQGET . Le informazioni sul gruppo e sul segmento per le chiamate di esplorazione vengono conservate separatamente dalle informazioni per le chiamate che rimuovono i messaggi dalla coda. Se si specifica MQGMO_BROWSE_FIRST, il gestore code ignora le informazioni sul gruppo e sul segmento per la ricerca. Esegue la scansione della coda come se non ci fossero gruppi correnti e messaggi logici correnti. Se la chiamata MQGET ha esito positivo, codice di completamento MQCC_OK o MQCC_WARNING, le informazioni sul gruppo e sul segmento per la ricerca sono impostate su quelle del messaggio restituito. Se la chiamata ha esito negativo, le informazioni sul gruppo e sul segmento rimangono le stesse che erano prima della chiamata.

MQGMO_BROWSE_SUCCESSIVO
Avanzare il cursore di ricerca al messaggio successivo sulla coda che soddisfa i criteri di selezione specificati nella chiamata MQGET . Il messaggio viene restituito all'applicazione, ma rimane nella coda.

MQGMO_BROWSE_NEXT non è valido con nessuna delle seguenti opzioni:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

È anche un errore se la coda non è stata aperta per la ricerca.

MQGMO_BROWSE_NEXT si comporta come MQGMO_BROWSE_FIRST, se è la prima chiamata per sfogliare una coda, dopo che la coda è stata aperta per la ricerca.

Il messaggio sotto il cursore potrebbe essere rimosso dalla coda prima che venga emessa la successiva chiamata MQGET con MQGMO_BROWSE_NEXT . Il cursore di ricerca rimane logicamente nella posizione nella coda occupata dal messaggio, anche se tale posizione è ora vuota.

I messaggi vengono memorizzati nella coda in due modi:

- FIFO entro la priorità (MQMDS_PRIORITY) oppure
- FIFO indipendentemente dalla priorità (MQMDS_FIFO)

L'attributo della coda **MsgDeliverySequence** indica quale metodo si applica (consultare “Attributi per le code” a pagina 858 per i dettagli).

Una coda potrebbe avere un MsgDeliverySequence di MQMDS_PRIORITY. Un messaggio arriva sulla coda con una priorità più alta di quella attualmente indicata dal cursore di ricerca. In tal caso, il messaggio con priorità più alta non viene trovato durante lo sweep corrente della coda utilizzando MQGMO_BROWSE_NEXT. Può essere trovato solo dopo che il cursore di ricerca è stato reimpostato con MQGMO_BROWSE_FIRSTo riaprendo la coda.

L'opzione MQGMO_MSG_UNDER_CURSOR può essere utilizzata con una chiamata MQGET non - browse, se necessario, per rimuovere il messaggio dalla coda.

Il cursore di esplorazione non viene spostato da chiamate MQGET non di esplorazione che utilizzano lo stesso handle Hobj .

Specificare l'opzione MQGMO_LOCK con questa opzione per vincolare il messaggio visualizzato.

È possibile specificare MQGMO_BROWSE_NEXT con qualsiasi combinazione valida delle opzioni MQGMO_* e MQMO_* che controllano l'elaborazione dei messaggi in gruppi e segmenti di messaggi logici.

Se si specifica MQGMO_LOGICAL_ORDER, i messaggi vengono esaminati in ordine logico. Se si omette tale opzione, i messaggi vengono visualizzati in ordine fisico. Se si specifica MQGMO_BROWSE_FIRST, è possibile passare tra ordine logico e ordine fisico. Le chiamate MQGET successive che utilizzano MQGMO_BROWSE_NEXT sfogliano la coda nello stesso ordine della chiamata più recente che ha specificato MQGMO_BROWSE_FIRST per l'handle di coda. La chiamata ha esito negativo con codice motivo MQRC_INCONSISTENT_BROWSE se questa condizione non viene soddisfatta.

Nota: Prestare particolare attenzione quando si utilizza una chiamata MQGET per sfogliare oltre la fine di un gruppo di messaggi se MQGMO_LOGICAL_ORDER non è specificato. Ad esempio, si supponga che l'ultimo messaggio nel gruppo preceda il primo messaggio nel gruppo nella coda. Utilizzando MQGMO_BROWSE_NEXT per esplorare oltre la fine del gruppo, specificando MQMO_MATCH_MSG_SEQ_NUMBER con MsgSeqNumber impostato su 1 viene restituito il primo messaggio nel gruppo già esplorato. Questo risultato può verificarsi immediatamente o un numero di chiamate MQGET successive se sono presenti gruppi che intervengono. La stessa considerazione si applica per un messaggio logico non presente in un gruppo.

Le informazioni sul gruppo e sul segmento per le chiamate di esplorazione vengono conservate separatamente dalle informazioni per le chiamate che rimuovono i messaggi dalla coda.

MQGMO_BROWSE_MSG_UNDER_CURSOR
Richiamare il messaggio a cui punta il cursore di esplorazione in modo non distruttivo, indipendentemente dalle opzioni MQMO_* specificate nel campo MatchOptions in MQGMO.

MQGMO_BROWSE_MSG_UNDER_CURSOR non è valido con nessuna delle seguenti opzioni:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

È anche un errore se la coda non è stata aperta per la ricerca.

Il messaggio puntato dal cursore di ricerca è quello che è stato richiamato per ultimo utilizzando l'opzione MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT . La chiamata ha esito negativo se nessuna di queste chiamate è stata emessa per questa coda da quando è stata aperta. La chiamata ha esito negativo anche se il messaggio che si trovava sotto il cursore di ricerca è stato richiamato in modo distruttivo.

La posizione del cursore di ricerca non viene modificata da questa chiamata.

L'opzione MQGMO_MSG_UNDER_CURSOR può essere utilizzata con una chiamata MQGET non - browse, per rimuovere il messaggio dalla coda.

Il cursore di ricerca non viene spostato da una chiamata MQGET non di ricerca, anche se si utilizza lo stesso handle Hobj . Inoltre, non viene spostato da una chiamata MQGET di ricerca che restituisce un codice di completamento di MQCC_FAILEDo un codice motivo di MQRC_TRUNCATED_MSG_FAILED.

Se MQGMO_BROWSE_MSG_UNDER_CURSOR viene specificato con MQGMO_LOCK:

- Se è già presente un messaggio bloccato, deve essere quello sotto il cursore, in modo che venga restituito senza sbloccare e bloccare nuovamente. Il messaggio rimane bloccato.

- Se non c'è alcun messaggio bloccato e c'è un messaggio sotto il cursore di ricerca, viene bloccato e restituito all'applicazione. Se non c'è alcun messaggio sotto il cursore di esplorazione, la chiamata non riesce.

Se MQGMO_BROWSE_MSG_UNDER_CURSOR viene specificato senza MQGMO_LOCK:

- Se esiste già un messaggio bloccato, deve essere quello sotto il cursore. Il messaggio viene restituito all'applicazione e sbloccato. Poiché il messaggio è ora sbloccato, non vi è alcuna garanzia che possa essere visualizzato di nuovo o richiamato in modo distruttivo dalla stessa applicazione. Potrebbe essere stato richiamato in modo distruttivo da un'altra applicazione che richiama i messaggi dalla coda.
- Se non c'è alcun messaggio bloccato e c'è un messaggio sotto il cursore di ricerca, viene restituito all'applicazione. Se non è presente alcun messaggio sotto il cursore di esplorazione, la chiamata non riesce.

Se MQGMO_COMPLETE_MSG viene specificato con MQGMO_BROWSE_MSG_UNDER_CURSOR, il cursore di esplorazione deve identificare un messaggio il cui campo Offset in MQMD è zero. Se questa condizione non viene soddisfatta, la chiamata ha esito negativo con codice di errore MQRC_INVALID_MSG_UNDER_CURSOR.

Le informazioni sul gruppo e sul segmento per le chiamate di esplorazione vengono conservate separatamente dalle informazioni per le chiamate che rimuovono i messaggi dalla coda.

MQGMO_MSG_UNDER_CURSOR
Richiamare il messaggio a cui punta il cursore di esplorazione, indipendentemente dalle opzioni MQMO_* specificate nel campo MatchOptions in MQGMO. Il messaggio viene rimosso dalla coda.

Il messaggio puntato dal cursore di ricerca è quello che è stato richiamato per ultimo utilizzando l'opzione MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT .

Se MQGMO_COMPLETE_MSG viene specificato con MQGMO_MSG_UNDER_CURSOR, il cursore di esplorazione deve identificare un messaggio il cui campo Offset in MQMD è zero. Se questa condizione non viene soddisfatta, la chiamata ha esito negativo con codice di errore MQRC_INVALID_MSG_UNDER_CURSOR.

Questa opzione non è valida con nessuna delle seguenti opzioni:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_UNLOCK

È anche un errore se la coda non è stata aperta sia per la ricerca che per l'input. Se il cursore di ricerca non punta attualmente a un messaggio richiamabile, viene restituito un errore dalla chiamata MQGET .

MQGMO_MARK_BROWSE_HANDLE
Il messaggio restituito da un MQGETriuscito o identificato dal MsgTokenrestituito, viene contrassegnato. Il contrassegno è specifico per l'handle oggetto utilizzato nella chiamata.

Il messaggio non viene rimosso dalla coda.

MQGMO_MARK_BROWSE_HANDLE è valido solo se viene specificata anche una delle seguenti opzioni:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

MQGMO_MARK_BROWSE_HANDLE non è valido con nessuna delle seguenti opzioni:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK

- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

Il messaggio rimane in questo stato fino a quando non si verifica uno dei seguenti eventi:

- L'handle dell'oggetto interessato è chiuso, normalmente o in altro modo.
- Il messaggio non è contrassegnato per questo handle da una chiamata a MQGET con opzione MQGMO_UNMARK_BROWSE_HANDLE.
- Il messaggio viene restituito da una chiamata a MQGETdistruttivo, che viene completata con MQCC_OK o MQCC_WARNING. Lo stato del messaggio rimane modificato anche se il MQGET viene successivamente sottoposto a rollback.
- Il messaggio scade.

MQGMO_MARK_BROWSE_CO_OP
Il messaggio restituito da un MQGETriuscito o identificato dal *MsgToken*restituito, è contrassegnato per tutti gli handle nella serie cooperante.

Il contrassegno di livello cooperativo è in aggiunta a qualsiasi contrassegno di livello handle che potrebbe essere stato impostato.

Il messaggio non viene rimosso dalla coda.

MQGMO_MARK_BROWSE_CO_OP è valido solo se l'handle dell'oggetto utilizzato è stato restituito da una chiamata a MQOPEN che ha specificato MQOO_CO_OP. È necessario specificare anche una delle seguenti opzioni MQGMO :

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

Questa opzione non è valida con nessuna delle seguenti opzioni:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

Se il messaggio è già contrassegnato e l'opzione MQGMO_UNMARKED_BROWSE_MSG non è specificata, la chiamata ha esito negativo con MQCC_FAILED e codice motivo MQRC_MSG_MARKED_BROWSE_CO_OP.

Il messaggio rimane in questo stato fino a quando non si verifica uno dei seguenti eventi:

- Tutte le maniglie dell'oggetto nella serie cooperante sono chiuse.
- Il messaggio non è contrassegnato per i browser cooperanti da una chiamata a MQGET con l'opzione MQGMO_UNMARK_BROWSE_CO_OP.
- Il messaggio viene automaticamente annullato dal gestore code.
- Il messaggio viene restituito da una chiamata a un MQGETnon - browse. Lo stato del messaggio rimane modificato anche se il MQGET viene successivamente sottoposto a rollback.
- Il messaggio scade.

MQGMO_UNMARKED_BROWSE_MSG
Una chiamata a MQGET che specifica MQGMO_UNMARKED_BROWSE_MSG restituisce un messaggio che viene considerato non contrassegnato per il relativo handle. Non restituisce un messaggio se il messaggio è stato contrassegnato per la sua gestione. Inoltre, non restituisce il messaggio se la coda è stata aperta da una chiamata a MQOPEN, con l'opzione MQOO_CO_OP, e il messaggio è stato contrassegnato da un membro della serie cooperante.

Questa opzione non è valida con nessuna delle seguenti opzioni:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

MQGMO_UNMARK_BROWSE_CO_OP
Dopo una chiamata a MQGET che specifica questa opzione, il messaggio non viene più considerato da alcun handle aperto nella serie di handle cooperanti da contrassegnare per la serie cooperante. Il messaggio viene ancora considerato come contrassegnato a livello di gestione se è stato contrassegnato a livello di gestione prima di questa chiamata.

L'uso di MQGMO_UNMARK_BROWSE_CO_OP è valido solo con un handle restituito da una chiamata riuscita a MQOPEN con opzione MQOO_CO_OP. Il MQGET ha esito positivo anche se il messaggio non è considerato contrassegnato dalla serie di handle cooperante.

MQGMO_UNMARK_BROWSE_CO_OP non è valido su una chiamata MQGET non di ricerca o con una delle seguenti opzioni:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNLOCK
- MQGMO_UNMARKED_BROWSE_MSG

MQGMO_UNMARK_BROWSE_HANDLE
Dopo una chiamata a MQGET che specifica questa opzione, il messaggio individuato non viene più considerato contrassegnato da questo handle.

La chiamata ha esito positivo anche se il messaggio non è contrassegnato per questo handle.

Questa opzione non è valida su una chiamata MQGET non di ricerca o con una delle seguenti opzioni:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNLOCK
- MQGMO_UNMARKED_BROWSE_MSG

Opzioni di blocco

Le seguenti opzioni si riferiscono al blocco dei messaggi sulla coda:

LOCK_MQGMO
Bloccare il messaggio che viene esplorato, in modo che il messaggio diventi invisibile a qualsiasi altro handle aperto per la coda. L'opzione può essere specificata solo se viene specificata anche una delle seguenti opzioni:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

Solo un messaggio può essere bloccato per ogni handle di coda. Il messaggio può essere un messaggio logico o un messaggio fisico:

- Se si specifica MQGMO_COMPLETE_MSG, tutti i segmenti di messaggio che costituiscono il messaggio logico vengono bloccati nell'handle della coda. I messaggi devono essere tutti presenti nella coda e disponibili per il recupero.
- Se si omette MQGMO_COMPLETE_MSG, solo un singolo messaggio fisico viene bloccato all'handle della coda. Se questo messaggio è un segmento di un messaggio logico, il segmento bloccato impedisce ad altre applicazioni di utilizzare MQGMO_COMPLETE_MSG per richiamare o sfogliare il messaggio logico.

Il messaggio bloccato è sempre quello sotto il cursore di ricerca. Il messaggio può essere rimosso dalla coda da una chiamata MQGET successiva che specifica l'opzione MQGMO_MSG_UNDER_CURSOR . Anche altre chiamate MQGET che utilizzano l'handle della coda possono rimuovere il messaggio (ad esempio, una chiamata che specifica l'identificativo del messaggio bloccato).

Se la chiamata restituisce il codice di completamento MQCC_FAILEDo MQCC_WARNING con codice motivo MQRC_TRUNCATED_MSG_FAILED, non viene bloccato alcun messaggio.

Se l'applicazione non rimuove il messaggio dalla coda, il blocco viene rilasciato da una delle seguenti azioni:

- Emettere un'altra chiamata MQGET per questo handle, specificando MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT. Il blocco viene rilasciato se la chiamata viene completata con MQCC_OK o MQCC_WARNING. Il messaggio rimane bloccato se la chiamata viene completata con MQCC_FAILED. Tuttavia, si applicano le seguenti eccezioni:
  - Il messaggio non viene sbloccato se MQCC_WARNING viene restituito con MQRC_TRUNCATED_MSG_FAILED.
  - Il messaggio viene sbloccato se MQCC_FAILED viene restituito con MQRC_NO_MSG_AVAILABLE.

  Se si specifica anche MQGMO_LOCK, il messaggio restituito è bloccato. Se si omette MQGMO_LOCK, non vi è alcun messaggio bloccato dopo la chiamata.

  Se si specifica MQGMO_WAITe non è immediatamente disponibile alcun messaggio, il messaggio originale viene sbloccato prima dell'inizio dell'attesa.
- Esecuzione di un'altra chiamata MQGET per questo handle, con MQGMO_BROWSE_MSG_UNDER_CURSOR, senza MQGMO_LOCK. Il blocco viene rilasciato se la chiamata viene completata con MQCC_OK o MQCC_WARNING. Il messaggio rimane bloccato se la chiamata viene completata con MQCC_FAILED. Tuttavia, si applica la seguente eccezione:
  - Il messaggio non viene sbloccato se MQCC_WARNING viene restituito con MQRC_TRUNCATED_MSG_FAILED.
- Emissione di un'altra chiamata MQGET per questo handle con MQGMO_UNLOCK.
- Esecuzione di una chiamata MQCLOSE utilizzando l'handle. MQCLOSE potrebbe essere implicito, causato dalla chiusura dell'applicazione.

Non è richiesta alcuna opzione MQOPEN speciale per specificare MQGMO_LOCK, diversa da MQOO_BROWSE, che è necessaria per specificare un'opzione di ricerca di accompagnamento.

MQGMO_LOCK non è valido con nessuna delle seguenti opzioni:

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

MQGMO_UNLOCK
Il messaggio da sbloccare deve essere stato precedentemente bloccato da una chiamata MQGET con l'opzione MQGMO_LOCK . Se non è presente alcun messaggio bloccato per questo handle, la chiamata viene completata con MQCC_WARNING e MQRC_NO_MSG_LOCKED.

I parametri **MsgDesc**, **BufferLength**, **Buffer**e **DataLength** non vengono controllati o modificati se si specifica MQGMO_UNLOCK. Nessun messaggio viene restituito in *Buffer*.

Non è richiesta alcuna opzione di apertura speciale per specificare MQGMO_UNLOCK (anche se MQOO_BROWSE è necessario per emettere la richiesta di blocco in primo luogo).

Questa opzione non è valida con le opzioni tranne le seguenti:

- MQGMO_NO_WAIT
- MQGMO_NO_SYNCPOINT

Si presuppone che entrambe queste opzioni siano specificate o meno.

Opzioni dei dati del messaggio

Le seguenti opzioni si riferiscono all'elaborazione dei dati del messaggio quando il messaggio viene letto dalla coda:

MQGMO_ACCEPT_TRUNCATED_MSG
Se il buffer di messaggi è troppo piccolo per contenere il messaggio completo, consentire alla chiamata MQGET di riempire il buffer. MQGET riempie il buffer con il maggior numero di messaggi possibile. Emette un codice di completamento di avvertenza e completa la sua elaborazione. Questo vuol dire che:

- Quando si sfogliano i messaggi, il cursore di esplorazione viene avanzato al messaggio restituito.
- Quando si rimuovono i messaggi, il messaggio restituito viene rimosso dalla coda.
- Se non si verifica alcun altro errore, viene restituito il codice motivo MQRC_TRUNCATED_MSG_ACCEPTED.

Senza questa opzione, il buffer viene ancora riempito con la quantità di messaggi che può contenere. Viene emesso un codice di completamento di avvertenza, ma l'elaborazione non è stata completata. Questo vuol dire che:

- Quando si sfogliano i messaggi, il cursore di esplorazione non è avanzato.
- Quando si rimuovono i messaggi, il messaggio non viene rimosso dalla coda.
- Se non si verifica alcun altro errore, viene restituito il codice di errore MQRC_TRUNCATED_MSG_FAILED .

MQGMO_CONVERT
Questa opzione converte i dati dell'applicazione nel messaggio in modo che siano conformi ai valori CodedCharSetId e Encoding specificati nel parametro **MsgDesc** nella chiamata MQGET . I dati vengono convertiti prima di essere copiati sul parametro **Buffer** .

Il campo Format specificato quando il messaggio è stato inserito viene assunto dal processo di conversione per identificare la natura dei dati nel messaggio. I dati del messaggio vengono convertiti dal gestore code per i formati integrati e da un'uscita scritta dall'utente per altri formati. Consultare “Uscita conversione dati” a pagina 932 per i dettagli dell'uscita di conversione dati.

- Se la conversione ha esito positivo, i campi CodedCharSetId e Encoding specificati nel parametro **MsgDesc** non vengono modificati al ritorno dalla chiamata MQGET .
- Se solo la conversione non riesce, i dati del messaggio vengono restituiti non convertiti. I campi CodedCharSetId e Encoding in MsgDesc sono impostati sui valori per il messaggio non convertito. Il codice di completamento è MQCC_WARNING in questo caso.

In entrambi i casi, questi campi descrivono l'identificativo della serie di caratteri e la codifica dei dati del messaggio restituiti nel parametro **Buffer** .

Consultare il campo *Format* descritto in “MQMD - Descrittore messaggi” a pagina 431 per un elenco dei nomi di formato per cui il gestore code esegue la conversione.

Opzioni di gruppo e segmento

Le seguenti opzioni sono relative all'elaborazione dei messaggi in gruppi e segmenti di messaggi logici. Prima delle descrizioni delle opzioni, ecco alcune definizioni di termini importanti:

Messaggio fisico
Un messaggio fisico è l'unità di informazioni più piccola che può essere inserita o rimossa da una coda. Spesso corrisponde alle informazioni specificate o richiamate in una singola chiamata MQPUT, MQPUT1o MQGET . Ogni messaggio fisico ha il proprio descrittori di messaggi, MQMD. Generalmente, i messaggi fisici vengono distinti da valori differenti per l'identificativo del messaggio, il campo MsgId in MQMD. Il gestore code non applica valori diversi.

Messaggio logico
Un messaggio logico è una singola unità di informazioni sull'applicazione. In assenza di vincoli di sistema, un messaggio logico è uguale a un messaggio fisico. Se i messaggi logici sono grandi, i vincoli di sistema potrebbero rendere consigliabile o necessario suddividere un messaggio logico in due o più messaggi fisici, denominati segmenti.

Un messaggio logico che è stato segmentato è costituito da due o più messaggi fisici che hanno lo stesso identificativo di gruppo non null, campo GroupId in MQMD. Hanno lo stesso numero di sequenza del messaggio, campo MsgSeqNumber in MQMD. I segmenti sono distinti da valori differenti per l'offset del segmento, campo Offset in MQMD. L'offset del segmento è l'offset dei dati nel messaggio fisico dall'inizio dei dati nel messaggio logico. Poiché ogni segmento è un messaggio fisico, i segmenti in un messaggio logico di solito hanno identificativi di messaggio differenti.

Un messaggio logico che non è stato segmentato, ma per cui la segmentazione è stata consentita dall'applicazione di invio, ha anche un identificativo di gruppo non null. In questo caso, esiste un solo messaggio fisico con tale identificativo di gruppo se il messaggio logico non appartiene a un gruppo di messaggi. I messaggi logici, per i quali la segmentazione è stata inibita dall'applicazione di invio, hanno un identificativo di gruppo null, MQGI_NONE, a meno che il messaggio logico non appartenga a un gruppo di messaggi.

Gruppo di messaggi
Un gruppo di messaggi è una serie di uno o più messaggi logici che hanno lo stesso identificativo di gruppo non null. I messaggi logici nel gruppo sono distinti da valori differenti per il numero di sequenza del messaggio. Il numero di sequenza è un numero intero compreso tra 1 e n, dove n è il numero di messaggi logici nel gruppo. Se uno o più messaggi logici sono segmentati, nel gruppo sono presenti più di n messaggi fisici.

ORDER LOGICAL_MQGMO_
MQGMO_LOGICAL_ORDER controlla l'ordine in cui i messaggi vengono restituiti dalle chiamate MQGET successive per l'handle della coda. L'opzione deve essere specificata su ogni chiamata.

Se MQGMO_LOGICAL_ORDER viene specificato per le chiamate MQGET successive per lo stesso handle della coda, i messaggi nei gruppi vengono restituiti nell'ordine dei relativi numeri di sequenza dei messaggi. I segmenti dei messaggi logici vengono restituiti nell'ordine fornito dagli offset dei segmenti. Questo ordine potrebbe essere diverso dall'ordine in cui tali messaggi e segmenti si verificano nella coda.

Nota: La specifica di MQGMO_LOGICAL_ORDER non ha conseguenze negative sui messaggi che non appartengono ai gruppi e che non sono segmenti. In effetti, tali messaggi vengono trattati come se ciascuno appartenesse a un gruppo di messaggi costituito da un solo messaggio. È sicuro specificare MQGMO_LOGICAL_ORDER quando si richiamano i messaggi dalle code che contengono una combinazione di messaggi in gruppi, segmenti di messaggi e messaggi non segmentati non in gruppi.

Per restituire i messaggi nell'ordine richiesto, il gestore code conserva le informazioni sul gruppo e sul segmento tra le chiamate MQGET successive. Le informazioni sul gruppo e sul segmento identificano il gruppo di messaggi corrente e il messaggio logico corrente per l'handle della coda. Identifica inoltre

la posizione corrente all'interno del gruppo e del messaggio logico e se i messaggi vengono richiamati all'interno di un'unità di lavoro. Poiché il gestore code conserva queste informazioni, l'applicazione non deve impostare le informazioni sul gruppo e sul segmento prima di ogni chiamata MQGET . In particolare, significa che l'applicazione non deve impostare i campi GroupId, MsgSeqNumbere Offset in MQMD. Tuttavia, l'applicazione deve impostare correttamente l'opzione MQGMO_SYNCPOINT o MQGMO_NO_SYNCPOINT su ogni chiamata.

Quando la coda viene aperta, non esiste alcun gruppo di messaggi corrente e nessun messaggio logico corrente. Un gruppo di messaggi diventa il gruppo di messaggi corrente quando un messaggio con l'indicatore MQMF_MSG_IN_GROUP viene restituito dalla chiamata MQGET . Con MQGMO_LOGICAL_ORDER specificato su chiamate successive, quel gruppo rimane il gruppo corrente fino a quando non viene restituito un messaggio che ha:

- MQMF_LAST_MSG_IN_GROUP senza MQMF_SEGMENT (vale a dire, l'ultimo messaggio logico nel gruppo non è segmentato) oppure
- MQMF_LAST_MSG_IN_GROUP con MQMF_LAST_SEGMENT (ossia, il messaggio restituito è l'ultimo segmento dell'ultimo messaggio logico del gruppo).

Quando viene restituito un messaggio di questo tipo, il gruppo di messaggi viene terminato e al completamento della chiamata MQGET non esiste più un gruppo corrente. In modo simile, un messaggio logico diventa il messaggio logico corrente quando un messaggio con l'indicatore MQMF_SEGMENT viene restituito dalla chiamata MQGET . Il messaggio logico viene terminato quando viene restituito il messaggio con l'indicatore MQMF_LAST_SEGMENT .

Se non viene specificato alcun criterio di selezione, le chiamate MQGET successive restituiscono, nell'ordine corretto, i messaggi per il primo gruppo di messaggi sulla coda. Restituiscono quindi i messaggi per il secondo gruppo di messaggi e così via, fino a quando non sono disponibili ulteriori messaggi. È possibile selezionare i gruppi di messaggi particolari restituiti specificando una o più delle seguenti opzioni nel campo MatchOptions :

- MQMO_MATCH_MSG_ID
- MQMO_MATCH_CORREL_ID
- MQMO_MATCH_GROUP_ID

Tuttavia, queste opzioni sono valide solo quando non esiste un gruppo di messaggi corrente o un messaggio logico. Per ulteriori dettagli, consultare il campo MatchOptions descritto in “MQGMO - Opzioni Get - message” a pagina 376 .

Tabella 495 a pagina 403 mostra i valori dei campi MsgId, CorrelId, GroupId, MsgSeqNumbere Offset che il gestore code cerca quando tenta di individuare un messaggio da restituire alla chiamata MQGET . Le regole si applicano sia alla rimozione dei messaggi dalla coda che alla visualizzazione dei messaggi sulla coda. Nella tabella, indica Sì o No:

LOG ORD
Indica se l'opzione MQGMO_LOGICAL_ORDER è specificata nella chiamata.

Cur grp
Indica se esiste un gruppo di messaggi corrente prima della chiamata.

Cur log msg
Indica se esiste un messaggio logico corrente prima della chiamata.

Altre colonne
Mostra i valori che il gestore code cerca. Precedente indica il valore restituito per il campo nel messaggio precedente per la gestione della coda.

Tabella 495. Opzioni MQGET relative ai messaggi in gruppi e segmenti di messaggi logici

Opzioni specificate	**Stato messaggio di log e gruppo prima della chiamata**		**Valori che il gestore code cerca**				
LOGORD	**Curgrp**	**Curlogmsg**	*MsgId*	*CorrelId*	*GroupId*	*MsgSeqNumber*	*Offset*
Sì	No	No	Controllato da *MatchOptions*	Controllato da *MatchOptions*	Controllato da *MatchOptions*	1	0
Sì	No	Sì	Qualsiasi identificativo di messaggio	Qualsiasi identificativo di correlazione	Identificativo gruppo precedente	1	Offset precedente + lunghezza segmento precedente
Sì	Sì	No	Qualsiasi identificativo di messaggio	Qualsiasi identificativo di correlazione	Identificativo gruppo precedente	Numero di sequenza precedente + 1	0
Sì	Sì	Sì	Qualsiasi identificativo di messaggio	Qualsiasi identificativo di correlazione	Identificativo gruppo precedente	Numero sequenza precedente	Offset precedente + lunghezza segmento precedente
No	Entrambi	Entrambi	Controllato da *MatchOptions*	Controllato da *MatchOptions*	Controllato da *MatchOptions*	Controllato da *MatchOptions*	Controllato da *MatchOptions*

Se sulla coda sono presenti più gruppi di messaggi idonei per la restituzione, i gruppi vengono restituiti nell'ordine determinato dalla posizione sulla coda del primo segmento del primo messaggio logico in ciascun gruppo. Vale a dire, i messaggi fisici che hanno un numero di sequenza di messaggi pari a 1 e gli offset pari a 0, determinano l'ordine in cui vengono restituiti i gruppi idonei.

L'opzione MQGMO_LOGICAL_ORDER influisce sulle unità di lavoro nel modo seguente:

- Se il primo messaggio logico o segmento in un gruppo viene richiamato all'interno di un'unità di lavoro, tutti gli altri messaggi logici e segmenti nel gruppo devono essere richiamati all'interno di un'unità di lavoro, se viene utilizzato lo stesso handle di coda. Tuttavia, non è necessario recuperarli all'interno della stessa unità di lavoro. Ciò consente a un gruppo di messaggi costituito da molti messaggi fisici di essere suddiviso in due o più unità di lavoro consecutive per la gestione della coda.
- Se il primo messaggio logico o segmento in un gruppo non viene richiamato all'interno di un'unità di lavoro e viene utilizzato lo stesso handle della coda, nessuno degli altri messaggi logici e segmenti nel gruppo può essere richiamato all'interno di un'unità di lavoro.

Se queste condizioni non vengono soddisfatte, la chiamata MQGET ha esito negativo con codice di errore MQRC_INCONSISTENT_UOW.

Quando si specifica MQGMO_LOGICAL_ORDER , l' MQGMO fornito nella chiamata MQGET non deve essere minore di MQGMO_VERSION_2e l' MQMD non deve essere minore di MQMD_VERSION_2. Se questa condizione non viene soddisfatta, la chiamata ha esito negativo con il codice di errore MQRC_WRONG_GMO_VERSION o MQRC_WRONG_MD_VERSION, come appropriato.

Se MQGMO_LOGICAL_ORDER non è specificato per le successive chiamate MQGET per l'handle della coda, i messaggi vengono restituiti indipendentemente dal fatto che appartengano a gruppi di

messaggi o che siano segmenti di messaggi logici. Ciò significa che i messaggi o i segmenti di un particolare gruppo o messaggio logico potrebbero essere restituiti fuori ordine o mescolati con messaggi o segmenti di altri gruppi o messaggi logici o con messaggi che non sono in gruppi e non sono segmenti. In questa situazione, i particolari messaggi restituiti da chiamate MQGET successive vengono controllati dalle opzioni MQMO_* specificate su tali chiamate (consultare il campo *MatchOptions* descritto in "MQGMO - Opzioni Get - message" a pagina 376 per dettagli su queste opzioni).

Questa è la tecnica che può essere utilizzata per riavviare un gruppo di messaggi o un messaggio logico nel mezzo, dopo che si è verificato un errore di sistema. Quando il sistema viene riavviato, l'applicazione può impostare i campi GroupId, MsgSeqNumber, Offsete MatchOptions sui valori appropriati, quindi emettere la chiamata MQGET con MQGMO_SYNCPOINT o MQGMO_NO_SYNCPOINT impostati, ma senza specificare MQGMO_LOGICAL_ORDER. Se questa chiamata ha esito positivo, il gestore code conserva le informazioni sul gruppo e sul segmento e le successive chiamate MQGET che utilizzano tale handle di coda possono specificare MQGMO_LOGICAL_ORDER come normale.

Le informazioni sul segmento e sul gruppo che il gestore code conserva per la chiamata MQGET sono separate dalle informazioni sul gruppo e sul segmento che conserva per la chiamata MQPUT. Inoltre, il gestore code conserva informazioni separate per:

- Chiamate MQGET che rimuovono i messaggi dalla coda.
- MQGET chiamate che sfogliano i messaggi sulla coda.

Per qualsiasi handle di coda fornito, l'applicazione può combinare chiamate MQGET che specificano MQGMO_LOGICAL_ORDER con chiamate MQGET che non lo fanno. Tuttavia, notare i seguenti punti:

- Se si omette MQGMO_LOGICAL_ORDER, ogni chiamata MQGET eseguita con esito positivo fa sì che il gestore code imposti le informazioni sul gruppo e sul segmento salvate sui valori corrispondenti al messaggio restituito; ciò sostituisce le informazioni sul gruppo e sul segmento esistenti conservate dal gestore code per l'handle della coda. Vengono modificate solo le informazioni appropriate all'azione della chiamata (sfoglia o rimuovi).
- Se si omette MQGMO_LOGICAL_ORDER, la chiamata non avrà esito negativo se è presente un gruppo di messaggi corrente o un messaggio logico; la chiamata potrebbe avere esito positivo con un codice di completamento MQCC_WARNING . Tabella 496 a pagina 404 mostra i vari casi che possono verificarsi. In questi casi, se il codice di completamento non è MQCC_OK, il codice di errore è uno dei seguenti (come appropriato):
  - MQRC_INCOMPLETE_GROUP
  - MQRC_INCOMPLETE_MSG
  - MQRC_INCONSISTENT_UOW

  Nota: Il gestore code non controlla le informazioni sul gruppo e sul segmento durante l'esplorazione di una coda o durante la chiusura di una coda aperta per l'esplorazione ma non l'immissione; in questi casi il codice di completamento è sempre MQCC_OK (supponendo che non vi siano altri errori).

Tabella 496. Risultato quando la chiamata MQGET o MQCLOSE non è congruente con le informazioni sul gruppo e sul segmento

La chiamata corrente è	La chiamata precedente era MQGET con MQGMO_LOGICAL_ORDER	La chiamata precedente era MQGET senza MQGMO_LOGICAL_ORDER
MQGET con MQGMO_LOGICAL_ORDER	MQCC_FAILED	MQCC_FAILED
MQGET senza MQGMO_LOGICAL_ORDER	MQCC_WARNING	MQCC_OK
MQCLOSE con un gruppo non terminato o un messaggio logico	MQCC_WARNING	MQCC_OK

Le applicazioni che desiderano richiamare i messaggi e i segmenti in ordine logico sono consigliabili per specificare MQGMO_LOGICAL_ORDER, poiché questa è l'opzione più semplice da utilizzare. Questa opzione allevia l'applicazione della necessità di gestire le informazioni sul gruppo e sul segmento, poiché il gestore code gestisce tali informazioni. Tuttavia, le applicazioni specializzate potrebbero richiedere un controllo maggiore rispetto a quello fornito dall'opzione MQGMO_LOGICAL_ORDER e ciò può essere ottenuto non specificando tale opzione. L'applicazione deve quindi verificare che i campi MsgId, CorrelId, GroupId, MsgSeqNumbere Offset in MQMDe le opzioni MQMO_* in MatchOptions in MQGMOsiano impostati correttamente, prima di ogni chiamata MQGET .

Ad esempio, un'applicazione che desidera inoltrare i messaggi fisici ricevuti, indipendentemente dal fatto che tali messaggi si trovino in gruppi o segmenti di messaggi logici, non deve specificare MQGMO_LOGICAL_ORDER. In una rete complessa con più percorsi tra i gestori code di invio e di ricezione, i messaggi fisici potrebbero arrivare fuori ordine. Specificando né MQGMO_LOGICAL_ORDER, né il MQPMO_LOGICAL_ORDER corrispondente nella chiamata MQPUT, l'applicazione di inoltro può recuperare e inoltrare ogni messaggio fisico non appena arriva, senza dover attendere il successivo in ordine logico.

È possibile specificare MQGMO_LOGICAL_ORDER con una qualsiasi delle altre opzioni MQGMO_* e con varie opzioni MQMO_* nelle circostanze appropriate (vedere la sezione precedente).

- z/OS Su z/OS, questa opzione è supportata per code private e condivise, ma la coda deve avere un tipo di indice MQIT_GROUP_ID. Per le code condivise, l'oggetto CFSTRUCT a cui è associata la coda deve essere a CFLEVEL (3) o superiore.
- Questa opzione è supportata per tutte le code locali per le seguenti piattaforme:
  - AIX AIX
  - Linux Linux
  - IBM i IBM i
  - Windows Windows

  e per IBM MQ MQI clients connessi a questi sistemi,.

MQGMO_COMPLETE_MSG
La chiamata MQGET può restituire solo un messaggio logico completo. Se il messaggio logico è segmentato, il gestore code riassembla i segmenti e restituisce il messaggio logico completo all'applicazione; il fatto che il messaggio logico sia stato segmentato non è evidente per l'applicazione che lo richiama.

Nota: Questa è l'unica opzione che consente al gestore code di riassemblare i segmenti di messaggi. Se non viene specificato, i segmenti vengono restituiti singolarmente all'applicazione se sono presenti nella coda (e soddisfano gli altri criteri di selezione specificati nella chiamata MQGET). Le applicazioni che non desiderano ricevere singoli segmenti devono specificare sempre MQGMO_COMPLETE_MSG.

Per utilizzare questa opzione, l'applicazione deve fornire un buffer abbastanza grande da contenere il messaggio completo oppure specificare l'opzione MQGMO_ACCEPT_TRUNCATED_MSG .

Se la coda contiene messaggi segmentati con alcuni dei segmenti mancanti (forse perché sono stati ritardati nella rete e non sono ancora arrivati), specificando MQGMO_COMPLETE_MSG si impedisce il richiamo dei segmenti appartenenti a messaggi logici incompleti. Tuttavia, tali segmenti di messaggi contribuiscono ancora al valore dell'attributo della coda **CurrentQDepth** ; ciò significa che potrebbero non essere presenti messaggi logici richiamabili, anche se *CurrentQDepth* è maggiore di zero.

Per i messaggi permanenti, il gestore code può riassemblare i segmenti solo all'interno di un'unità di lavoro:

- Se la chiamata MQGET sta operando all'interno di un'unità di lavoro definita dall'utente, viene utilizzata tale unità di lavoro. Se la chiamata non riesce durante il processo di riassemblaggio, il gestore code reinstalla sulla coda tutti i segmenti che sono stati rimossi durante il riassemblaggio. Tuttavia, l'errore non impedisce il corretto commit dell'unità di lavoro.

- Se la chiamata opera al di fuori di un'unità di lavoro definita dall'utente e non esiste alcuna unità di lavoro definita dall'utente, il gestore code crea un'unità di lavoro per la durata della chiamata. Se la chiamata ha esito positivo, il gestore code esegue automaticamente il commit dell'unità di lavoro (non è necessario che l'applicazione esegua questa operazione). Se la chiamata ha esito negativo, il gestore code esegue il backout dell'unità di lavoro.
- Se la chiamata opera all'esterno di un'unità di lavoro definita dall'utente, ma esiste un'unità di lavoro definita dall'utente, il gestore code non può riassemblare. Se il messaggio non richiede il riassemblaggio, la chiamata può ancora avere esito positivo. Ma se il messaggio richiede il riassemblaggio, la chiamata ha esito negativo con codice di errore MQRC_UOW_NOT_AVAILABLE.

Per i messaggi non persistenti, il gestore code non necessita di un'unità di lavoro disponibile per eseguire il riassemblaggio.

Ogni messaggio fisico che è un segmento ha il proprio descrittore di messaggi. Per i segmenti che costituiscono un singolo messaggio logico, la maggior parte dei campi nel descrittore del messaggio sono gli stessi per tutti i segmenti nel messaggio logico; di solito sono solo i campi MsgId, Offsete MsgFlags che differiscono tra i segmenti nel messaggio logico. Tuttavia, se un segmento viene posizionato su una coda di messaggi non recapitabili in un gestore code intermedio, il gestore DLQ richiama il messaggio specificando l'opzione MQGMO_CONVERT e ciò può causare la modifica della serie di caratteri o della codifica del segmento. Se il gestore DLQ invia correttamente il segmento nel suo percorso, il segmento potrebbe avere una serie di caratteri o una codifica diversa dagli altri segmenti nel messaggio logico quando il segmento arriva al gestore code di destinazione.

Un messaggio logico costituito da segmenti in cui i campi di CodedCharSetId e Encoding differiscono non può essere riassemblato dal gestore code in un singolo messaggio logico. Invece, il gestore code riassembla e restituisce i primi segmenti consecutivi all'inizio del messaggio logico che hanno gli stessi identificativi e codifiche della serie di caratteri e la chiamata MQGET viene completata con il codice di completamento MQCC_WARNING e il codice motivo MQRC_INCONSISTENT_CCSIDS o MQRC_INCONSISTENT_ENCODINGS, come appropriato. Ciò si verifica indipendentemente dal fatto che MQGMO_CONVERT sia specificato o meno. Per richiamare i rimanenti segmenti, l'applicazione deve emettere nuovamente la chiamata MQGET senza l'opzione MQGMO_COMPLETE_MSG , richiamando i segmenti uno alla volta. MQGMO_LOGICAL_ORDER può essere utilizzato per richiamare i rimanenti segmenti in ordine.

Un'applicazione che inserisce segmenti può anche impostare altri campi nel descrittore del messaggio su valori che differiscono tra i segmenti. Tuttavia, non vi è alcun vantaggio se l'applicazione ricevente utilizza MQGMO_COMPLETE_MSG per richiamare il messaggio logico. Quando il gestore code riassembla un messaggio logico, restituisce nel descrittore del messaggio i valori del descrittore del messaggio per il primo segmento; l'unica eccezione è il campo MsgFlags , che il gestore code imposta per indicare che il messaggio riassemblato è l'unico segmento.

Se MQGMO_COMPLETE_MSG è specificato per un messaggio di report, il gestore code esegue un'elaborazione speciale. Il gestore code controlla la coda per vedere se tutti i messaggi di report di quel tipo relativi ai diversi segmenti nel messaggio logico sono presenti nella coda. Se lo sono, possono essere richiamati come un singolo messaggio specificando MQGMO_COMPLETE_MSG. Affinché ciò sia possibile, i messaggi di report devono essere generati da un gestore code o da un MCA che supporta la segmentazione oppure l'applicazione di origine deve richiedere almeno 100 byte di dati del messaggio (ovvero, è necessario specificare le opzioni MQRO_*_WITH_DATA o MQRO_*_WITH_FULL_DATA appropriate). Se per un segmento è presente meno dell'intera quantità di dati dell'applicazione, i byte mancanti vengono sostituiti da valori null nel messaggio di report restituito.

Se MQGMO_COMPLETE_MSG viene specificato con MQGMO_MSG_UNDER_CURSOR o MQGMO_BROWSE_MSG_UNDER_CURSOR, il cursore di esplorazione deve essere posizionato su un messaggio il cui campo *Offset* in MQMD ha un valore pari a 0. Se questa condizione non viene soddisfatta, la chiamata ha esito negativo con codice di errore MQRC_INVALID_MSG_UNDER_CURSOR.

MQGMO_COMPLETE_MSG implica MQGMO_ALL_SEGMENTS_AVAILABLE, che quindi non deve essere specificato.

MQGMO_COMPLETE_MSG può essere specificato con qualsiasi altra opzione MQGMO_* ad eccezione di MQGMO_SYNCPOINT_IF_PERSISTENTe con qualsiasi altra opzione MQMO_* ad eccezione di MQMO_MATCH_OFFSET.

- z/OS Su z/OS, questa opzione è supportata per code private e condivise, ma la coda deve avere un tipo di indice MQIT_GROUP_ID. Per le code condivise, l'oggetto CFSTRUCT a cui è associata la coda deve essere a CFLEVEL (3) o superiore.
- Sulle seguenti piattaforme:
  - AIX AIX
  - IBM i IBM i
  - Linux Linux
  - Windows Windows

  e per IBM MQ MQI clients connessi a tali sistemi, questa opzione è supportata per tutte le code locali.

MQGMO_ALL_MSGS_AVAILABLE

I messaggi in un gruppo diventano disponibili per il richiamo solo quando sono disponibili tutti i messaggi nel gruppo. Se la coda contiene gruppi di messaggi con alcuni dei messaggi mancanti (forse perché sono stati ritardati nella rete e non sono ancora arrivati), specificando MQGMO_ALL_MSGS_AVAILABLE si impedisce il richiamo dei messaggi appartenenti a gruppi incompleti. Tuttavia, tali messaggi contribuiscono ancora al valore dell'attributo della coda **CurrentQDepth** ; ciò significa che potrebbero non essere presenti gruppi di messaggi richiamabili, anche se CurrentQDepth è maggiore di zero. Se non sono presenti altri messaggi richiamabili, il codice di errore MQRC_NO_MSG_AVAILABLE viene restituito dopo la scadenza dell'intervallo di attesa specificato (se presente).

L'elaborazione di MQGMO_ALL_MSGS_AVAILABLE dipende dal fatto che MQGMO_LOGICAL_ORDER sia specificato o meno:

- Se vengono specificate entrambe le opzioni, MQGMO_ALL_MSGS_AVAILABLE ha effetto solo quando non vi è alcun gruppo corrente o messaggio logico. Se è presente un gruppo corrente o un messaggio logico, MQGMO_ALL_MSGS_AVAILABLE viene ignorato. Ciò significa che MQGMO_ALL_MSGS_AVAILABLE può rimanere attivo durante l'elaborazione dei messaggi in ordine logico.
- Se MQGMO_ALL_MSGS_AVAILABLE viene specificato senza MQGMO_LOGICAL_ORDER, MQGMO_ALL_MSGS_AVAILABLE ha sempre un effetto. Ciò significa che l'opzione deve essere disattivata dopo che il primo messaggio nel gruppo è stato rimosso dalla coda, per poter rimuovere i restanti messaggi nel gruppo.

Il corretto completamento di una chiamata MQGET specificando MQGMO_ALL_MSGS_AVAILABLE significa che, al momento dell'emissione della chiamata MQGET , tutti i messaggi nel gruppo si trovavano nella coda. Tuttavia, tenere presente che altre applicazioni possono ancora rimuovere i messaggi dal gruppo (il gruppo non è bloccato per l'applicazione che richiama il primo messaggio nel gruppo).

Se si omette questa opzione, i messaggi appartenenti ai gruppi possono essere richiamati anche quando il gruppo è incompleto.

MQGMO_ALL_MSGS_AVAILABLE implica MQGMO_ALL_SEGMENTS_AVAILABLE, che quindi non deve essere specificato.

MQGMO_ALL_MSGS_AVAILABLE può essere specificato con una qualsiasi delle altre opzioni MQGMO_* e con una delle opzioni MQMO_* .

- z/OS Su z/OS, questa opzione è supportata per code private e condivise, ma la coda deve avere un tipo di indice MQIT_GROUP_ID. Per le code condivise, l'oggetto CFSTRUCT a cui è associata la coda deve essere a CFLEVEL (3) o superiore.

- Sulle seguenti piattaforme:
  -  AIX
  -  IBM i
  -  Linux
  -  Windows

  e per IBM MQ MQI clients connessi a tali sistemi, questa opzione è supportata per tutte le code locali.

MQGMO_ALL_SEGMENTS_AVAILABLE

I segmenti in un messaggio logico diventano disponibili per il richiamo solo quando tutti i segmenti nel messaggio logico sono disponibili. Se la coda contiene messaggi segmentati con alcuni dei segmenti mancanti (forse perché sono stati ritardati nella rete e non sono ancora arrivati), specificando MQGMO_ALL_SEGMENTS_AVAILABLE si impedisce il richiamo dei segmenti appartenenti a messaggi logici incompleti. Tuttavia, tali segmenti contribuiscono ancora al valore dell'attributo della coda **CurrentQDepth** ; ciò significa che potrebbero non essere presenti messaggi logici richiamabili, anche se CurrentQDepth è maggiore di zero. Se non sono presenti altri messaggi richiamabili, il codice di errore MQRC_NO_MSG_AVAILABLE viene restituito dopo la scadenza dell'intervallo di attesa specificato (se presente).

L'elaborazione di MQGMO_ALL_SEGMENTS_AVAILABLE dipende dal fatto che MQGMO_LOGICAL_ORDER sia specificato o meno:

- Se vengono specificate entrambe le opzioni, MQGMO_ALL_SEGMENTS_AVAILABLE ha effetto solo quando non è presente alcun messaggio logico corrente. Se è presente un messaggio logico corrente, MQGMO_ALL_SEGMENTS_AVAILABLE viene ignorato. Ciò significa che MQGMO_ALL_SEGMENTS_AVAILABLE può rimanere attivo durante l'elaborazione dei messaggi in ordine logico.
- Se MQGMO_ALL_SEGMENTS_AVAILABLE viene specificato senza MQGMO_LOGICAL_ORDER, MQGMO_ALL_SEGMENTS_AVAILABLE ha sempre un effetto. Ciò significa che l'opzione deve essere disattivata dopo che il primo segmento nel messaggio logico è stato rimosso dalla coda, in modo da poter rimuovere i restanti segmenti nel messaggio logico.

Se questa opzione non viene specificata, i segmenti del messaggio possono essere richiamati anche quando il messaggio logico è incompleto.

Mentre sia MQGMO_COMPLETE_MSG che MQGMO_ALL_SEGMENTS_AVAILABLE richiedono che tutti i segmenti siano disponibili prima che possano essere richiamati, il primo restituisce il messaggio completo, mentre il secondo consente ai segmenti di essere richiamati uno per uno.

Se MQGMO_ALL_SEGMENTS_AVAILABLE è specificato per un messaggio di report, il gestore code controlla la coda per verificare se esiste almeno un messaggio di report per ciascuno dei segmenti che costituiscono il messaggio logico completo. In tal caso, viene soddisfatta la condizione MQGMO_ALL_SEGMENTS_AVAILABLE . Tuttavia, il gestore code non controlla il *tipo* dei messaggi di report presenti, pertanto potrebbe esserci una combinazione di tipi di report nei messaggi di report relativi ai segmenti del messaggio logico. Di conseguenza, l'esito positivo di MQGMO_ALL_SEGMENTS_AVAILABLE non implica che MQGMO_COMPLETE_MSG avrà esito positivo. Se esiste una combinazione di tipi di report presenti per i segmenti di un particolare messaggio logico, tali messaggi di report devono essere richiamati uno alla volta.

È possibile specificare MQGMO_ALL_SEGMENTS_AVAILABLE con qualsiasi altra opzione MQGMO_* e con una qualsiasi delle opzioni MQMO_* .

- Su z/OS, questa opzione è supportata per code private e condivise, ma la coda deve avere un tipo di indice MQIT_GROUP_ID. Per le code condivise, l'oggetto CFSTRUCT a cui è associata la coda deve essere a CFLEVEL (3) o superiore.
- Sulle seguenti piattaforme:

–  AIX

–  IBM i

–  Linux

–  Windows

e per IBM MQ MQI clients connessi a tali sistemi, questa opzione è supportata per tutte le code locali.

Opzioni proprietà

Le seguenti opzioni sono correlate alle proprietà del messaggio:

MQGMO_PROPERTIES_AS_Q_DEF

Le proprietà del messaggio, eccetto quelle contenute nel descrittore del messaggio (o estensione), devono essere rappresentate come definito dall'attributo della coda **PropertyControl** . Se viene fornito un MsgHandle , questa opzione viene ignorata e le proprietà del messaggio sono disponibili tramite MsgHandle, a meno che il valore dell'attributo della coda **PropertyControl** non sia MQPROP_FORCE_MQRFH2.

Questa è l'azione predefinita se non vengono specificate opzioni della proprietà.

MQGMO_PROPERTIES_IN_HANDLE

Le proprietà del messaggio devono essere rese disponibili tramite MsgHandle. Se non viene fornita alcuna gestione messaggio, la chiamata non riesce con codice d'errore MQRC_HMSG_ERROR.

Nota: Se il messaggio viene letto in un secondo momento da un'applicazione che non crea un handle del messaggio, il gestore code inserisce le proprietà del messaggio in una struttura MQRFH2 . È possibile che la presenza di un'intestazione MQRFH2 non prevista interrompe il comportamento di una applicazione esistente.

MQGMO_NO_PROPERTIES

Non verrà richiamata alcuna proprietà del messaggio, ad eccezione di quelle contenute nel descrittore del messaggio (o estensione). Se viene fornito un MsgHandle , verrà ignorato.

MQGMO_PROPERTIES_FORCE_MQRFH2

Le proprietà del messaggio, eccetto quelle contenute nel descrittore del messaggio (o estensione) devono essere rappresentate utilizzando intestazioni MQRFH2. Ciò fornisce la compatibilit ... con la versione precedente per le applicazioni che prevedono di recuperare le propriet ... ma non possono essere modificate per utilizzare gli handle dei messaggi. Se viene fornito un MsgHandle , viene ignorato.

MQGMO_PROPERTIES_COMPATIBILITY

Se il messaggio contiene una proprietà con prefisso **"mcd."**, **"jms."**, **"usr."**o **"mqext."**, tutte le proprietà del messaggio vengono consegnate all'applicazione in un'intestazione MQRFH2 . Altrimenti tutte le proprietà del messaggio, eccetto quelle contenute nel descrittore messaggi (o nell'estensione), vengono eliminate e non sono più accessibili sull'applicazione.

Opzione predefinita

Se nessuna delle opzioni descritte è richiesta, è possibile utilizzare la seguente opzione:

MQGMO_NONE

Utilizzare questo valore per indicare che non sono state specificate altre opzioni; tutte le opzioni assumono i propri valori predefiniti. MQGMO_NONE supporta la documentazione del programma; non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Il valore iniziale del campo Options è MQGMO_NO_WAIT più MQGMO_PROPERTIES_AS_Q_DEF.

WaitInterval (MQLONG) per MQGMO

Si tratta del tempo approssimativo, espresso in millisecondi, in cui la chiamata MQGET attende l'arrivo di un messaggio adatto (ossia, un messaggio che soddisfa i criteri di selezione specificati nel parametro **MsgDesc** della chiamata MQGET.

Importante: Non c'è attesa, o ritardo, se un messaggio adatto è disponibile immediatamente.

Consultare il campo *MsgId* descritto in "MQMD - Descrittore messaggi" a pagina 431 per ulteriori dettagli). Se non è arrivato alcun messaggio appropriato dopo questo tempo, la chiamata viene completata con MQCC_FAILED e codice motivo MQRC_NO_MSG_AVAILABLE.

Su z/OS, il periodo di tempo in cui la chiamata MQGET è effettivamente in attesa è influenzato dal caricamento del sistema e dalle considerazioni sulla pianificazione del lavoro e può variare tra il valore specificato per *WaitInterval* e circa 100 millisecondi in più di *WaitInterval*.

WaitInterval viene utilizzato insieme all'opzione MQGMO_WAIT o MQGMO_SET_SIGNAL. Viene ignorato se non viene specificato nessuno di questi valori. Se viene specificato uno di questi valori, *WaitInterval* deve essere maggiore o uguale a zero oppure il seguente valore speciale:

MQWI_ILLIMITATO
Intervallo di attesa illimitato.

Il valore iniziale di questo campo è 0.

Signal1 (MQLONG) per MQGMO

Questo è un campo di input utilizzato solo insieme all'opzione MQGMO_SET_SIGNAL; identifica un segnale che deve essere consegnato quando un messaggio è disponibile.

Nota: Il tipo di dati e l'utilizzo di questo campo sono determinati dall'ambiente; per questo motivo, le applicazioni che si desidera trasferire tra ambienti differenti non devono utilizzare segnali.

- In z/OS, questo campo deve contenere l'indirizzo di un ECB (Event Control Block). La BCE deve essere autorizzata dall'applicazione prima dell'emissione della chiamata MQGET. La memoria contenente la BCE non deve essere liberata fino alla chiusura della coda. La BCE viene inviata dal gestore code con uno dei codici di completamento del segnale descritti. Questi codici di completamento sono impostati in bit da 2 a 31 della BCE, l'area definita nella macro di mappatura z/OS IHAECB come per un codice di completamento utente.
- In tutti gli altri ambienti, questo è un campo riservato; il suo valore non è significativo.

I codici di completamento del segnale sono:

MQEC_MSG_ARRIVED
È arrivato un messaggio adatto sulla coda. Questo messaggio non è stato riservato al chiamante; è necessario emettere una seconda richiesta MQGET, ma un'altra applicazione potrebbe richiamare il messaggio prima che venga effettuata la seconda richiesta.

MQEC_WAIT_INTERVAL_EXPIRED
Il *WaitInterval* specificato è scaduto senza l'arrivo di un messaggio adatto.

MQEC_WAIT_ANNULLATO
L'attesa è stata annullata per un motivo indeterminato (come la chiusura del gestore code o la disabilitazione della coda). Emettere nuovamente la richiesta se si desidera un'ulteriore diagnosi.

MQEC_Q_MGR_QUIESCING
L'attesa è stata annullata perché il gestore code è entrato nello stato di sospensione (MQGMO_FAIL_IF_QUIESCING è stato specificato nella chiamata MQGET).

MQEC_CONNECTION_QUIESCING
L'attesa è stata annullata perché la connessione è entrata nello stato di sospensione (MQGMO_FAIL_IF_QUIESCING è stato specificato nella chiamata MQGET).

Il valore iniziale di questo campo è determinato dall'ambiente:

- Su z/OS, il valore iniziale è il puntatore null.

- In tutti gli ambienti, il valore iniziale è 0.

Signal2 (MQLONG) per MQGMO

Questo è un campo di input utilizzato solo insieme all'opzione MQGMO_SET_SIGNAL. È un campo riservato; il suo valore non è significativo.

Il valore iniziale di questo campo è 0.

ResolvedQName (MQCHAR48) per MQGMO

È un campo di output che il gestore code imposta sul nome locale della coda da cui è stato richiamato il messaggio, come definito nel gestore code locale. Questo è diverso dal nome utilizzato per aprire la coda se:

- È stata aperta una coda alias (in tal caso, viene restituito il nome della coda locale a cui è stato risolto l'alias) oppure
- È stata aperta una coda modello (in tal caso, viene restituito il nome della coda locale dinamica).

La lunghezza di questo campo è fornita da MQ_Q_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

MatchOptions (MQLONG) per MQGMO

Queste opzioni consentono all'applicazione di scegliere quali campi nel parametro **MsgDesc** utilizzare per selezionare il messaggio restituito dalla chiamata MQGET. L'applicazione imposta le opzioni richieste in questo campo e imposta i campi corrispondenti nel parametro **MsgDesc** sui valori richiesti per tali campi. Solo i messaggi che hanno tali valori in MQMD per il messaggio sono candidati per il richiamo utilizzando tale parametro **MsgDesc** nella chiamata MQGET. I campi per i quali non viene specificata l'opzione di corrispondenza vengono ignorati quando si seleziona il messaggio da restituire. Se non si specifica alcun criterio di selezione nella chiamata MQGET (ovvero, *qualsiasi* messaggio è accettabile), impostare *MatchOptions* su MQMO_NONE.

- Su z/OS, i criteri di selezione che possono essere utilizzati potrebbero essere limitati dal tipo di indice utilizzato per la coda. Consultare l'attributo della coda **IndexType** per ulteriori dettagli.

Se si specifica MQGMO_LOGICAL_ORDER, solo alcuni messaggi sono idonei per la restituzione dalla successiva chiamata MQGET:

- Se non è presente alcun gruppo corrente o messaggio logico, solo i messaggi che hanno *MsgSeqNumber* uguale a 1 e *Offset* uguale a 0 sono idonei per la restituzione. In questa situazione, è possibile utilizzare una o più delle seguenti opzioni di corrispondenza per selezionare quale dei messaggi idonei viene restituito:
  - ID MQMO_MATCH_MSG_ID
  - ID CORREL_MQMO_MATCH_
  - ID_GROUP_MATCH_MQMO
- Se è presente un gruppo corrente o un messaggio logico, solo il messaggio successivo nel gruppo o il segmento successivo nel messaggio logico è idoneo per la restituzione e non può essere modificato specificando le opzioni MQMO_ *.

In entrambi i casi precedenti, è possibile specificare le opzioni di corrispondenza che non si applicano, ma il valore del campo pertinente nel parametro **MsgDesc** deve corrispondere al valore del campo corrispondente nel messaggio da restituire; la chiamata ha esito negativo con il codice motivo MQRC_MATCH_OPTIONS_ERROR se questa condizione non viene soddisfatta.

MatchOptions viene ignorato se si specifica MQGMO_MSG_UNDER_CURSOR o MQGMO_BROWSE_MSG_UNDER_CURSOR.

L'ottenimento dei messaggi in base alla proprietà del messaggio non viene eseguito utilizzando le opzioni di corrispondenza; per ulteriori informazioni, consultare “SelectionString (MQCHARV) per MQOD” a pagina 505.

È possibile specificare una o più delle seguenti opzioni di corrispondenza:

ID MQMO_MATCH_MSG_ID
Il messaggio da recuperare deve avere un identificativo di messaggio che corrisponda al valore del campo *MsgId* nel parametro **MsgDesc** della chiamata MQGET. Questa corrispondenza è in aggiunta a tutte le altre corrispondenze che potrebbero essere applicate (ad esempio, l'identificativo di correlazione).

Se si omette questa opzione, il campo *MsgId* nel parametro **MsgDesc** viene ignorato e qualsiasi identificativo del messaggio corrisponderà.

Nota: L'identificativo del messaggio MQMI_NONE è un valore speciale che corrisponde a qualsiasi identificativo del messaggio in MQMD per il messaggio. Pertanto, specificare MQMO_MATCH_MSG_ID con MQMI_NONE equivale a non specificare MQMO_MATCH_MSG_ID.

ID CORREL_MQMO_MATCH_
Il messaggio da recuperare deve avere un identificatore di correlazione che corrisponda al valore del campo *CorrelId* nel parametro **MsgDesc** della chiamata MQGET. Questa corrispondenza si aggiunge a qualsiasi altra corrispondenza che potrebbe essere applicata (ad esempio, l'identificativo del messaggio).

Se si omette questa opzione, il campo *CorrelId* nel parametro **MsgDesc** viene ignorato e qualsiasi identificativo di correlazione corrisponderà.

Nota: L'identificativo di correlazione MQCI_NONE è un valore speciale che corrisponde a *qualsiasi* identificativo di correlazione in MQMD per il messaggio. Pertanto, specificare MQMO_MATCH_CORREL_ID con MQCI_NONE equivale a non specificare MQMO_MATCH_CORREL_ID.

ID_GROUP_MATCH_MQMO
Il messaggio da recuperare deve avere un identificativo del gruppo che corrisponda al valore del campo *GroupId* nel parametro **MsgDesc** della chiamata MQGET. Questa corrispondenza è in aggiunta a tutte le altre corrispondenze che potrebbero essere applicate (ad esempio, l'identificativo di correlazione).

Se si omette questa opzione, il campo *GroupId* nel parametro **MsgDesc** viene ignorato e qualsiasi identificativo di gruppo corrisponderà.

Nota: L'identificativo gruppo MQGI_NONE è un valore speciale che corrisponde a *qualsiasi* identificativo gruppo in MQMD per il messaggio. Pertanto, specificare MQMO_MATCH_GROUP_ID con MQGI_NONE equivale a non specificare MQMO_MATCH_GROUP_ID.

NUMERO SEQ MQMO_MATCH_MSG_
Il messaggio da recuperare deve avere un numero di sequenza del messaggio che corrisponda al valore del campo *MsgSeqNumber* nel parametro **MsgDesc** della chiamata MQGET. Questa corrispondenza è in aggiunta a tutte le altre corrispondenze che potrebbero essere applicate (ad esempio, l'identificativo del gruppo).

Se si omette questa opzione, il campo *MsgSeqNumber* nel parametro **MsgDesc** viene ignorato e qualsiasi numero di sequenza del messaggio corrisponderà.

MQMO_MATCH_OFFSET
Il messaggio da recuperare deve avere un offset che corrisponda al valore del campo *Offset* nel parametro **MsgDesc** della chiamata MQGET. Questa corrispondenza è in aggiunta a tutte le altre corrispondenze che potrebbero essere applicate (ad esempio, il numero di sequenza del messaggio).

Se si omette questa opzione non viene specificata, il campo *Offset* nel parametro **MsgDesc** viene ignorato e qualsiasi offset corrisponderà.

- Questa opzione non è supportata su z/OS.

MQMO_MATCH_MSG_TOKEN
Il messaggio da recuperare deve avere un token di messaggio che corrisponda al valore del campo *MsgToken* nella struttura MQGMO specificata nella chiamata MQGET.

È possibile specificare questa opzione per tutte le code locali. Se viene specificato per una coda che ha un *IndexType* MQIT_MSG_TOKEN (una coda gestita da WLM), non è possibile specificare altre opzioni di corrispondenza con MQMO_MATCH_MSG_TOKEN.

Non è possibile specificare MQMO_MATCH_MSG_TOKEN con MQGMO_WAIT o MQGMO_SET_SIGNAL. Se l'applicazione desidera attendere l'arrivo di un messaggio su una coda con *IndexType* MQIT_MSG_TOKEN, specificare MQMO_NONE.

Se si omette questa opzione, il campo *MsgToken* in MQGMO viene ignorato e qualsiasi token del messaggio corrisponderà.

Se non si specifica alcuna delle opzioni descritte, è possibile utilizzare la seguente opzione:

MQMO_NONE
Non utilizzare corrispondenze nella selezione del messaggio da restituire; tutti i messaggi nella coda sono idonei per il richiamo (ma soggetti al controllo delle opzioni MQGMO_ALL_MSGS_AVAILABLE, MQGMO_ALL_SEGMENTS_AVAILABLE e MQGMO_COMPLETE_MSG).

MQMO_NONE supporta la documentazione del programma. Non è previsto che questa opzione venga utilizzata con altre opzioni MQMO_ *, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Questo è un campo di immissione. Il valore iniziale di questo campo è MQMO_MATCH_MSG_ID con MQMO_MATCH_CORREL_ID. Questo campo viene ignorato se *Version* è minore di MQGMO_VERSION_2.

Nota: Il valore iniziale del campo *MatchOptions* è definito per la compatibilità con i precedenti gestori code MQSeries . Tuttavia, quando si legge una serie di messaggi da una coda senza utilizzare criteri di selezione, questo valore iniziale richiede che l'applicazione reimposti i campi *MsgId* e *CorrelId* su MQMI_NONE e MQCI_NONE prima di ogni chiamata MQGET. Evitare la necessità di reimpostare *MsgId* e *CorrelId* impostando *Version* su MQGMO_VERSION_2e *MatchOptions* su MQMO_NONE.

Concetti correlati
Selettori di messaggi in JMS

GroupStatus (MQCHAR) per MQGMO

Questo indicatore indica se il messaggio richiamato si trova in un gruppo.

Ha uno dei seguenti valori:

MQGS_NOT_IN_GROUP
Il messaggio non è in un gruppo.

MQGS_MSG_IN_GROUP
Il messaggio si trova in un gruppo, ma non è l'ultimo nel gruppo.

MQGS_LAST_MSG_IN_GROUP
Il messaggio è l'ultimo nel gruppo.

Questo è anche il valore restituito se il gruppo è composto da un solo messaggio.

Questo è un campo di output. Il valore iniziale di questo campo è MQGS_NOT_IN_GROUP. Questo campo viene ignorato se *Version* è minore di MQGMO_VERSION_2.

SegmentStatus (MQCHAR) per MQGMO

Si tratta di un indicatore che indica se il messaggio richiamato è un segmento di un messaggio logico. Ha uno dei seguenti valori:

MQSS_NOT_A_SEGMENT
Il messaggio non è un segmento.

ISCRIZIONE MQSS_SEGMENT
Il messaggio è un segmento, ma non è l'ultimo segmento del messaggio logico.

ISCRIZIONE MQSS_LAST_SEGMENT
Il messaggio è l'ultimo segmento del messaggio logico.

Questo è anche il valore restituito se il messaggio logico è composto da un solo segmento.

In z/OS, il gestore code imposta sempre questo campo su MQSS_NOT_A_SEGMENT.

Questo è un campo di output. Il valore iniziale di questo campo è MQSS_NOT_A_SEGMENT. Questo campo viene ignorato se *Version* è minore di MQGMO_VERSION_2.

Segmentazione (MQCHAR) per MQGMO

Questo è un indicatore che indica se è consentita un'ulteriore segmentazione per il messaggio richiamato. Ha uno dei seguenti valori:

MQSEG_INIBITO
Segmentazione non consentita.

MQSEG_CONSENTITO
Segmentazione consentita.

In z/OS, il gestore code imposta sempre questo campo su MQSEG_INIITED.

Questo è un campo di output. Il valore iniziale di questo campo è MQSEG_INIITED. Questo campo viene ignorato se *Version* è minore di MQGMO_VERSION_2.

Reserved1 (MQCHAR) for MQGMO

Questo è un campo riservato. Il valore iniziale di questo campo è un carattere vuoto. Questo campo viene ignorato se *Version* è minore di MQGMO_VERSION_2.

MsgToken (MQBYTE16) per MQGMO

Campo MsgToken - Struttura MQGMO. Questo campo viene utilizzato da un gestore code per identificare in modo univoco un messaggio.

Si tratta di una stringa di byte generata dal Gestore code per identificare un messaggio in modo univoco su una coda. Il token del messaggio viene generato quando il primo messaggio viene posizionato sul gestore code e rimane con il messaggio fino a quando il messaggio non viene rimosso definitivamente dal gestore code, a meno che il gestore code non venga riavviato.

Quando il messaggio viene rimosso da una coda, il *MsgToken* che ha identificato tale istanza del messaggio non è più valido e non viene mai riutilizzato. Se il gestore code viene riavviato, il *MsgToken* che ha identificato un messaggio sulla coda prima del riavvio potrebbe non essere valido dopo il riavvio. Tuttavia, *MsgToken* non viene mai riutilizzato per identificare un'altra istanza del messaggio. Il *MsgToken* viene generato dal gestore code e non è visibile ad alcuna applicazione esterna.

Quando un messaggio viene restituito da una chiamata a MQGET in cui viene fornito un MQGMO versione 3 o superiore, il *MsgToken* che identifica il messaggio sulla coda viene restituito in MQGMO dal gestore code. C'è un'unica eccezione: quando il messaggio viene rimosso dalla coda all'esterno del punto di sincronizzazione, il gestore code potrebbe non restituire un *MsgToken* perché non è utile identificare il messaggio restituito in una successiva chiamata MQGET. Le applicazioni devono utilizzare *MsgToken* solo per fare riferimento al messaggio nelle successive chiamate MQGET.

Se viene fornito un *MsgToken* e viene specificato *MatchOption* MQMO_MATCH_MSG_TOKEN e non viene specificato né MQGMO_MSG_UNDER_CURSOR né MQGMO_BROWSE_MSG_UNDER_CURSOR, è possibile restituire solo il messaggio identificato da tale *MsgToken* . L'opzione è valida su tutte le code locali indipendentemente da INDXTYPE e su z/OS è necessario utilizzare INDXTYPE (MSGTOKEN) solo sulle code WLM (Workload Manager).

Tutti gli altri *MatchOptions* specificati vengono controllati e se non corrispondono, viene restituito MQRC_NO_MSG_AVAILABLE. Se MQGMO_BROWSE_NEXT è codificato con MQMO_MATCH_MSG_TOKEN, il messaggio identificato da *MsgToken* viene restituito solo se si trova oltre il cursore di ricerca per l'handle di chiamata.

Se si specifica MQGMO_MSG_UNDER_CURSOR o MQGMO_BROWSE_MSG_UNDER_CURSOR, MQMO_MATCH_MSG_TOKEN viene ignorato.

MQMO_MATCH_MSG_TOKEN non è valido con le seguenti opzioni di richiamo messaggio:

- MQGMO_WAIT
- MQGMO_SET_SIGNAL

Per una chiamata MQGET che specifica MQMO_MATCH_MSG_TOKEN, è necessario fornire un MQGMO versione 3 o successiva alla chiamata, altrimenti viene restituito MQRC_WRONG_GMO_VERSION.

Se *MsgToken* non è valido in questo momento, viene restituito MQCC_FAILED con MQRC_NO_MSG_AVAILABLE, a meno che non si verifichi un altro errore.

ReturnedLength (MQLONG) per MQGMO

Questo è un campo di output che il gestore code imposta sulla lunghezza in byte dei dati del messaggio restituiti dalla chiamata MQGET nel parametro **Buffer** . Se il gestore code non supporta questa funzionalità, *ReturnedLength* è impostato sul valore MQRL_UNDEFINED.

Quando i messaggi vengono convertiti tra codifiche o serie di caratteri, i dati del messaggio a volte possono modificare la dimensione. Al ritorno dalla chiamata MQGET:

- Se *ReturnedLength* non è MQRL_UNDEFINED, il numero di byte dei dati del messaggio restituiti viene fornito da *ReturnedLength*.
- Se *ReturnedLength* ha il valore MQRL_UNDEFINED, il numero di byte dei dati del messaggio restituiti è di solito fornito dal valore più piccolo di *BufferLength* e *DataLength*, ma può essere *minore di* se la chiamata MQGET viene completata con il codice di errore MQRC_TRUNCATED_MSG_ACCEPTED. In questo caso, i byte non significativi nel parametro **Buffer** vengono impostati su valori null.

Viene definito il seguente valore speciale:

MQRL_UNDEFINED
Lunghezza dei dati restituiti non definita.

Su z/OS, il valore restituito per il campo *ReturnedLength* è sempre MQRL_UNDEFINED.

Il valore iniziale di questo campo è MQRL_UNDEFINED. Questo campo viene ignorato se *Version* è inferiore a MQGMO_VERSION_3.

Reserved2 (MQLONG) per MQGMO

Questo è un campo riservato. Il valore iniziale di questo campo è un carattere vuoto. Questo campo viene ignorato se *Version* è minore di **MQGMO_VERSION_4**.

MsgHandle (MQHMSG) per MQGMO

Se l'opzione MQGMO_PROPERTIES_AS_Q_DEF viene specificata e l'attributo della coda **PropertyControl** non è impostato su MQPROP_FORCE_MQRFH2 , questo è l'handle per un messaggio che verrà popolato con le proprietà del messaggio richiamato dalla coda. L'handle viene creato da una chiamata MQCRTMH. Tutte le proprietà già associate all'handle verranno eliminate prima di richiamare un messaggio.

È anche possibile specificare il seguente valore:
MQHM_NONE
Nessun handle del messaggio fornito.

Non è richiesto alcun descrittore di messaggi nella chiamata MQGET se viene fornito un handle del messaggio valido e utilizzato nell'output per contenere le proprietà del messaggio, il descrittore del messaggio associato all'handle del messaggio viene utilizzato per i campi di input.

Se viene specificato un descrittore del messaggio nella chiamata MQGET, ha sempre la precedenza sul descrittore del messaggio associato a un handle del messaggio.

Se viene specificato MQGMO_PROPERTIES_FORCE_MQRFH2 oppure viene specificato MQGMO_PROPERTIES_AS_Q_DEF e l'attributo della coda **PropertyControl** è

MQPROP_FORCE_MQRFH2 , la chiamata ha esito negativo con il codice motivo MQRC_MD_ERROR quando non viene specificato alcun parametro del descrittore del messaggio.

Al ritorno dalla chiamata MQGET, le proprietà e il descrittore del messaggio associati a questo handle del messaggio vengono aggiornati in modo da riflettere lo stato del messaggio richiamato (oltre al descrittore del messaggio se ne è stato fornito uno nella chiamata MQGET). Le proprietà del messaggio possono essere interrogate utilizzando la chiamata MQINQMP.

Fatta eccezione per le estensioni del descrittore del messaggio, quando presente, una proprietà che può essere interrogata con la chiamata MQINQMP non è contenuta nei dati del messaggio; se il messaggio sulla coda conteneva proprietà nei dati del messaggio, queste vengono rimosse dai dati del messaggio prima che i dati vengano restituiti all'applicazione.

Se non viene fornito alcun handle del messaggio o la versione è inferiore a MQGMO_VERSION_4 , è necessario fornire un descrittore di messaggio valido nella chiamata MQGET. Tutte le proprietà del messaggio (ad eccezione di quelle contenute nel descrittore del messaggio) vengono restituite nei dati del messaggio in base al valore delle opzioni della proprietà nella struttura MQGMO e nell'attributo della coda **PropertyControl** .

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQHM_NONE. Questo campo viene ignorato se **Version** è minore di MQGMO_VERSION_4.

MQIIH - Intestazione informazioni IMS

La struttura MQIIH descrive le informazioni di intestazione per un messaggio inviato a IMS attraverso il bridge IMS .Per qualsiasi piattaforma supportata da IBM MQ è possibile creare e trasmettere un messaggio che include la struttura MQIIH, ma solo un gestore code IBM MQ for z/OS può utilizzare il bridge IMS . Pertanto, affinché il messaggio arrivi a IMS da un gestore code nonz/OS , la rete del gestore code deve includere almeno un gestore code z/OS attraverso il quale il messaggio può essere instradato.

Disponibilità

Tutti i sistemi IBM MQ e client IBM MQ .

Nome formato

IMS MQFMT

Serie di caratteri e codifica

Condizioni speciali si applicano alla serie di caratteri e alla codifica utilizzati per la struttura MQIIH e i dati del messaggio dell'applicazione:

- Le applicazioni che si collegano al gestore code che possiede la coda bridge IMS devono fornire una struttura MQIIH che sia nella serie di caratteri e nella codifica del gestore code. Ciò è dovuto al fatto che la conversione dei dati della struttura MQIIH non viene eseguita in questo caso.
- Le applicazioni che si collegano ad altri gestori code possono fornire una struttura MQIIH che si trova in una delle serie di caratteri e codifiche supportate; l'agent del canale dei messaggi di ricezione connesso al gestore code che possiede la coda bridge IMS converte MQIIH.
- I dati del messaggio dell'applicazione che seguono la struttura MQIIH devono essere nella stessa serie di caratteri e codifica della struttura MQIIH. Non utilizzare i campi *CodedCharSetId* e *Encoding* nella struttura MQIIH per indicare la serie di caratteri e la codifica dei dati del messaggio dell'applicazione.

  È necessario fornire un'uscita di conversione dati per convertire i dati del messaggio dell'applicazione se i dati non sono uno dei formati integrati supportati dal gestore code.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 497. Campi in MQIIH per MQIIH

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQIIH	'IIH¬'
Versione (numero versione struttura)	MQIIH_VERSION_1	1
StrucLength (lunghezza della struttura MQIIH)	MQIIH_LENGTH_1	84
Codifica (riservato - consultare "Serie di caratteri e codifica" a pagina 416)	Nessuna	0
CodedCharSetId (riservato - vedere "Serie di caratteri e codifica" a pagina 416)	Nessuna	0
Formato (nome formatoMQ dei dati che seguono MQIIH)	MQFMT_NONE	Spazi
Indicatori (indicatori)	MQIIH_NONE	0
LTermOverride (sovrascrittura terminale logico)	Nessuna	Spazi
MFSMapName (nome mappa dei servizi in formato messaggio)	Nessuna	Spazi
ReplyToFormat (nome formatoMQ del messaggio di risposta)	MQFMT_NONE	Spazi
Programma di autenticazione (passwordRACF o passticket)	MQIAUT_NONE	Spazi
TranInstanceTranInstance (identificatore istanza transazione)	MQITII_NONE	Valori null
TranState (stato transazione)	MQITS_NON_IN_CONVERSAZIONE	'¬'
CommitMode (modalità commit)	MQICM_COMMIT_THEN_SEND	'0'
SecurityScope (ambito di sicurezza)	CHECK MQISS	'C'
Riservato (Riservato)	Nessuna	'¬'

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macroMQIIH_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura:

```
MQIIH MyIIH = {MQIIH_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQIIH

```
typedef struct tagMQIIH MQIIH;
struct tagMQIIH {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
```

```
  MQLONG    StrucLength;     /* Length of MQIIH structure */
  MQLONG    Encoding;        /* Reserved */
  MQLONG    CodedCharSetId;  /* Reserved */
  MQCHAR8   Format;          /* MQ format name of data that follows
                                MQIIH */
  MQLONG    Flags;           /* Flags */
  MQCHAR8   LTermOverride;   /* Logical terminal override */
  MQCHAR8   MFSMapName;      /* Message format services map name */
  MQCHAR8   ReplyToFormat;   /* MQ format name of reply message */
  MQCHAR8   Authenticator;   /* RACF password or passticket */
  MQBYTE16  TranInstanceId;  /* Transaction instance identifier */
  MQCHAR    TranState;       /* Transaction state */
  MQCHAR    CommitMode;      /* Commit mode */
  MQCHAR    SecurityScope;   /* Security scope */
  MQCHAR    Reserved;        /* Reserved */
};
```

Dichiarazione COBOL per MQIIH

```
**   MQIIH structure
  10 MQIIH.
**    Structure identifier
   15 MQIIH-STRUCID        PIC X(4).
**    Structure version number
   15 MQIIH-VERSION        PIC S9(9) BINARY.
**    Length of MQIIH structure
   15 MQIIH-STRUCLENGTH    PIC S9(9) BINARY.
**    Reserved
   15 MQIIH-ENCODING       PIC S9(9) BINARY.
**    Reserved
   15 MQIIH-CODEDCHARSETID PIC S9(9) BINARY.
**    MQ format name of data that follows MQIIH
   15 MQIIH-FORMAT         PIC X(8).
**    Flags
   15 MQIIH-FLAGS          PIC S9(9) BINARY.
**    Logical terminal override
   15 MQIIH-LTERMOVERRIDE  PIC X(8).
**    Message format services map name
   15 MQIIH-MFSMAPNAME     PIC X(8).
**    MQ format name of reply message
   15 MQIIH-REPLYTOFORMAT  PIC X(8).
**    RACF password or passticket
   15 MQIIH-AUTHENTICATOR  PIC X(8).
**    Transaction instance identifier
   15 MQIIH-TRANINSTANCEID PIC X(16).
**    Transaction state
   15 MQIIH-TRANSTATE      PIC X.
**    Commit mode
   15 MQIIH-COMMITMODE     PIC X.
**    Security scope
   15 MQIIH-SECURITYSCOPE  PIC X.
**    Reserved
   15 MQIIH-RESERVED       PIC X.
```

Dichiarazione PL/I per MQIIH

```
dcl
 1 MQIIH based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 StrucLength    fixed bin(31), /* Length of MQIIH structure */
  3 Encoding       fixed bin(31), /* Reserved */
  3 CodedCharSetId fixed bin(31), /* Reserved */
  3 Format         char(8),       /* MQ format name of data that follows
                                     MQIIH */
  3 Flags          fixed bin(31), /* Flags */
  3 LTermOverride  char(8),       /* Logical terminal override */
  3 MFSMapName     char(8),       /* Message format services map name */
  3 ReplyToFormat  char(8),       /* MQ format name of reply message */
  3 Authenticator  char(8),       /* RACF password or passticket */
  3 TranInstanceId char(16),      /* Transaction instance identifier */
  3 TranState      char(1),       /* Transaction state */
  3 CommitMode     char(1),       /* Commit mode */
  3 SecurityScope  char(1),       /* Security scope */
  3 Reserved       char(1);       /* Reserved */
```

Dichiarazione High Level Assembler per MQIIH

```
MQIIH                 DSECT
MQIIH_STRUCID         DS   CL4   Structure identifier
MQIIH_VERSION         DS   F     Structure version number
MQIIH_STRUCLENGTH     DS   F     Length of MQIIH structure
MQIIH_ENCODING        DS   F     Reserved
MQIIH_CODEDCHARSETID  DS   F     Reserved
MQIIH_FORMAT          DS   CL8   MQ format name of data that follows
*                                MQIIH
MQIIH_FLAGS           DS   F     Flags
MQIIH_LTERMOVERRIDE   DS   CL8   Logical terminal override
MQIIH_MFSMAPNAME      DS   CL8   Message format services map name
MQIIH_REPLYTOFORMAT   DS   CL8   MQ format name of reply message
MQIIH_AUTHENTICATOR   DS   CL8   RACF password or passticket
MQIIH_TRANINSTANCEID  DS   XL16  Transaction instance identifier
MQIIH_TRANSTATE       DS   CL1   Transaction state
MQIIH_COMMITMODE      DS   CL1   Commit mode
MQIIH_SECURITYSCOPE   DS   CL1   Security scope
MQIIH_RESERVED        DS   CL1   Reserved
*
MQIIH_LENGTH          EQU  *-MQIIH
                      ORG  MQIIH
MQIIH_AREA            DS   CL(MQIIH_LENGTH)
```

Dichiarazione Visual Basic per MQIIH

```
Type MQIIH
  StrucId        As String*4 'Structure identifier'
  Version        As Long     'Structure version number'
  StrucLength    As Long     'Length of MQIIH structure'
  Encoding       As Long     'Reserved'
  CodedCharSetId As Long     'Reserved'
  Format         As String*8 'MQ format name of data that follows MQIIH'
  Flags          As Long     'Flags'
  LTermOverride  As String*8 'Logical terminal override'
  MFSMapName     As String*8 'Message format services map name'
  ReplyToFormat  As String*8 'MQ format name of reply message'
  Authenticator  As String*8 'RACF password or passticket'
  TranInstanceId As MQBYTE16 'Transaction instance identifier'
  TranState      As String*1 'Transaction state'
  CommitMode     As String*1 'Commit mode'
  SecurityScope  As String*1 'Security scope'
  Reserved       As String*1 'Reserved'
End Type
```

StrucId (MQCHAR4) per MQIIH

Questo è l'identificativo della struttura dell'intestazione delle informazioni IMS . È sempre un campo di immissione. Il valore è MQIIH_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQIIH
Identificativo per la struttura dell'intestazione delle informazioni IMS .

Per il linguaggio di programmazione C, viene definita anche la costante MQIIH_STRUC_ID_ARRAY. Ha lo stesso valore di MQIIH_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQIIH

Questo è il numero di versione della struttura. Il valore deve essere:

MQIIH_VERSION_1
Numero di versione per la struttura dell'intestazione delle informazioni IMS .

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQIIH_CURRENT_
Versione corrente della struttura dell'intestazione delle informazioni IMS .

Il valore iniziale di questo campo è MQIIH_VERSION_1.

StrucLength (MQLONG) per MQIIH

È la lunghezza della struttura MQIIH. Il valore deve essere:

MQIIH_LENGTH_1
Lunghezza della struttura dell'intestazione delle informazioni IMS .

Il valore iniziale di questo campo è MQIIH_LENGTH_1.

Codifica (MQLONG) per MQIIH

Questo è un campo riservato; il suo valore non è significativo. Il valore iniziale di questo campo è 0.

La codifica per le strutture supportate che seguono una struttura MQIIH è uguale a quella della struttura MQIIH stessa e viene presa da qualsiasi intestazione MQ precedente.

CodedCharSetId (MQLONG) per MQIIH

Questo è un campo riservato; il suo valore non è significativo. Il valore iniziale di questo campo è 0.

L'ID serie di caratteri per le strutture supportate che seguono una struttura MQIIH è uguale a quello della struttura MQIIH stessa e viene preso da qualsiasi intestazione MQ precedente.

Formato (MQCHAR8) per MQIIH

Specifica il formato MQ dei dati che seguono la struttura MQIIH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati.

La lunghezza di questo campo è fornita da MQ_FORMAT_LENGTH. Il valore iniziale di questo campo è MQFMT_NONE.

Indicatori (MQLONG) per MQIIH

Il valore degli indicatori deve essere:

MQIIH_NONE
Nessun indicatore.

MQIIH_PASS_ESPIRAZIONE
Il messaggio di risposta contiene:

- Le stesse opzioni del report di scadenza del messaggio di richiesta
- Il tempo di scadenza rimanente dal messaggio di richiesta senza alcuna modifica effettuata per il tempo di elaborazione del ponte

Se questo valore non è impostato, l'ora di scadenza è impostata su *unlimited*.

MQIIH_REPLY_FORMAT_NONE
Imposta MQIIH.Format della risposta a MQFMT_NONE.

MQIIH_IGNORE_PURG
Imposta l'indicatore TMAMIPRG nel prefisso OTMA, che richiede a OTMA di ignorare le chiamate PURG sul PCB TP per transazioni CM0 .

MQIIH_CM0_REQUEST_RESPONSE
Per le transazioni in modalità di esecuzione commit 0 (CM0), questo indicatore imposta l'indicatore TMAMHRSP nel prefisso OTMA. L'impostazione di questo indicatore richiede che OTMA/IMS generi un messaggio DFS2082 RESPONSE MODE TRANSACTION TERMINATE WITHOUT REPLY quando il programma applicativo IMS originale non risponde all'IOPCB né passa ad un'altra transazione.

Il valore iniziale di questo campo è MQIIH_NONE.

LTermOverride (MQCHAR8) per MQIIH

La sostituzione del terminale logico, inserita nel campo PCB IO. È facoltativo; se non viene specificato, viene utilizzato il nome TPIPE. Viene ignorato se il primo byte è vuoto o null.

La lunghezza di questo campo è fornita da MQ_LTERM_OVERRIDE_LENGTH. Il valore iniziale di questo campo è di 8 caratteri vuoti.

MFSMapName (MQCHAR8) per MQIIH

Il nome della mappa dei servizi di formato del messaggio, inserito nel campo PCB IO. È facoltativo. In input rappresenta il MID, in output rappresenta il MOD. Viene ignorato se il primo byte è vuoto o null.

La lunghezza di questo campo è fornita da MQ_MFS_MAP_NAME_LENGTH. Il valore iniziale di questo campo è di 8 caratteri vuoti.

ReplyTo(MQCHAR8) per MQIIH

È il nome del formato MQ del messaggio di risposta inviato in risposta al messaggio corrente. La lunghezza di questo campo è fornita da MQ_FORMAT_LENGTH. Il valore iniziale di questo campo è MQFMT_NONE.

Per convertire i dati nel messaggio di risposta utilizzando MQGMO_CONVERT, specificare MQIIH.replyToFormat= MQFMT_STRING o MQIIH.replyToFormat= MQFMT_IMS_VAR_STRING. Per una spiegazione dell'utilizzo di questi campi, vedere “Formato (MQCHAR8) per MQMD” a pagina 458.

Se il valore predefinito (MQIIH.replyToFormat= MQFMT_NONE) viene utilizzato nel messaggio di richiesta e il messaggio di risposta viene richiamato utilizzando MQGMO_CONVERT, non viene eseguita alcuna conversione dei dati.

Programma di autenticazione (MQCHAR8) per MQIIH

Questa è la password RACF o PassTicket. È facoltativo; se specificato, viene utilizzato con l'ID utente nel contesto di sicurezza MQMD per creare un UTOKEN inviato a IMS per fornire un contesto di sicurezza. Se non viene specificato, l'ID utente viene utilizzato senza verifica. Ciò dipende dall'impostazione degli switch RACF , che potrebbe richiedere la presenza di un programma di autenticazione.

Viene ignorato se il primo byte è vuoto o null. È possibile utilizzare il seguente valore speciale:

MQIAUT_NONE
Nessuna autenticazione.

Per il linguaggio di programmazione C, viene definita anche la costante MQIAUT_NONE_ARRAY, che ha lo stesso valore di MQIAUT_NONE, ma è un array di caratteri invece di una stringa.

La lunghezza di questo campo è fornita da MQ_AUTHENTICATOR_LENGTH. Il valore iniziale di questo campo è MQIAUT_NONE.

ID TranInstance(MQBYTE16) per MQIIH

Questo è l'identificativo dell'istanza della transazione. Questo campo viene utilizzato dai messaggi di output da IMS, quindi viene ignorato al primo input. Se si imposta *TranState* su MQITS_IN_CONVERSATION, questo valore deve essere fornito nell'input successivo e in tutti gli input successivi, per consentire a IMS di correlare i messaggi alla conversazione corretta. È possibile utilizzare il seguente valore speciale:

MQITII_NONE
Nessun identificativo dell'istanza della transazione.

Per il linguaggio di programmazione C, viene definita anche la costante MQITII_NONE_ARRAY, che ha lo stesso valore di MQITII_NONE, ma è un array di caratteri invece di una stringa.

La lunghezza di questo campo è fornita da MQ_TRAN_INSTANCE_ID_LENGTH. Il valore iniziale di questo campo è MQITII_NONE.

TranState (MQCHAR) per MQIIH

Indica lo stato della conversazione IMS . Questo viene ignorato al primo input perché non esiste alcuna conversazione. Negli input successivi indica se una conversazione è attiva o meno. All'output viene impostato da IMS. Il valore deve essere uno dei seguenti.

MQITS_IN_CONVERSAZIONE
Nella conversazione.

MQITS_NON_IN_CONVERSAZIONE
Non in conversazione.

MQITS_ARCHITECTED
Restituire i dati di stato della transazione in formato architettato.

Questo valore viene utilizzato solo con il comando IMS `/DISPLAY TRAN` . Restituisce i dati di stato della transazione in formato IMS invece che in formato carattere. z/OS Per ulteriori informazioni, consultare Writing IMS transaction programs through IBM MQ.

Il valore iniziale di questo campo è MQITS_NOT_IN_CONVERSATION.

CommitMode (MQCHAR) per MQIIH

Questa è la modalità di commit IMS . Consultare il manuale *OTMA Reference* per ulteriori informazioni sulle modalità di commit IMS . Il valore deve essere uno dei seguenti.

MQICM_COMMIT_THEN_SEND
Commit e invio.

Questa modalità implica una doppia accodamento dell'output, ma tempi di occupazione della regione più brevi. Le transazioni interattive e Fast - path non possono essere eseguite con questa modalità.

MQICM_SEND_THEN_COMMIT
Inviare quindi eseguire il commit.

Qualsiasi transazione IMS avviata come risultato di una modalità di commit di MQICM_SEND_THEN_COMMIT viene eseguita in modalità RESPONSE indipendentemente dal modo in cui la transazione viene definita nella definizione del sistema IMS (parametro MSGTYPE nella macro TRANSACT). Ciò si applica anche alle transazioni avviate mediante un cambio di transazione.

Il valore iniziale di questo campo è MQICM_COMMIT_THEN_SEND.

SecurityScope (MQCHAR) per MQIIH

Ciò indica che è richiesta l'elaborazione della sicurezza IMS . Vengono definiti i seguenti valori:

CHECK MQISS
Verificare l'ambito di sicurezza: un ACEE viene creato nella regione di controllo, ma non nella regione dipendente.

MQISS_FULL
Ambito di sicurezza completo: un ACEE memorizzato nella cache viene costruito nella regione di controllo e un ACEE non memorizzato nella cache viene creato nella regione dipendente. Se si utilizza MQISS_FULL, assicurarsi che l'ID utente per cui viene creato l'ACEE abbia accesso alle risorse utilizzate nella regione dipendente.

Se per questo campo non viene specificato né MQISS_CHECK né MQISS_FULL, viene utilizzato MQISS_CHECK.

Il valore iniziale di questo campo è MQISS_CHECK.

Riservato (MQCHAR) per MQIIH

Questo è un campo riservato; deve essere vuoto.

MQIMPO - Opzioni della proprietà Interroga messaggio

La struttura MQIMPO consente alle applicazioni di specificare le opzioni che controllano la modalità di interrogazione delle proprietà dei messaggi. La struttura è un parametro di input nella chiamata MQINQMP.

Disponibilità

Tutti i sistemi IBM MQ e client IBM MQ .

Serie di caratteri e codifica

I dati in MQIMPO devono essere nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (MQENC_NATIVE).

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 498. Campi in MQIPMO

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQIMPO	'IMPO'
Versione (numero versione struttura)	MQIMPO_VERSION_1	1
Opzioni (opzioni che controllano l'azione di MQINQMP)	MQIMPO_INQ_FIRST	
RequestedEncoding (codifica in cui deve essere convertita la proprietà richiesta)	MQEN_NATIVE	
RequestedCCSID (serie di caratteri della proprietà richiesta)	APPL MQCCSI	
ReturnedEncoding (codifica del valore restituito)	MQEN_NATIVE	
ReturnedCCSID	0	
Reserved1 (campo riservato)	carattere vuoto (campo 4 byte)	
ReturnedName (nome della proprietà richiesta)	MQCHARV_PREDEFINITO	
TypeString (rappresentazione stringa del tipo di dati della proprietà)	Stringa null o spazi vuoti	

Note:

1. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.
2. Nel linguaggio di programmazione C, la variabile macroMQIMPO_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQIMPO MyIMPO = {MQIMPO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQIMPO

```
typedef struct tagMQIMPO MQIMPO;
struct tagMQIMPO {
```

```
  MQCHAR4  StrucId;            /* Structure identifier */
  MQLONG   Version;            /* Structure version number */
  MQLONG   Options;            /* Options that control the action of
                                  MQINQMP */
  MQLONG   RequestedEncoding;  /* Requested encoding of Value */
  MQLONG   RequestedCCSID;     /* Requested character set identifier
                                  of Value */
  MQLONG   ReturnedEncoding;   /* Returned encoding of Value */
  MQLONG   ReturnedCCSID;      /* Returned character set identifier
                                  of Value */
  MQCHAR   Reserved1           /* Reserved field */
  MQCHARV  ReturnedName;       /* Returned property name */
  MQCHAR8  TypeString;         /* Property data type as a string */
};
```

Dichiarazione COBOL per MQIMPO

```
**   MQIMPO structure
  10 MQIMPO.
**    Structure identifier
   15 MQIMPO-STRUCID               PIC X(4).
**    Structure version number
   15 MQIMPO-VERSION               PIC S9(9) BINARY.
**    Options that control the action of MQINQMP
   15 MQIMPO-OPTIONS               PIC S9(9) BINARY.
**    Requested encoding of VALUE
   15 MQIMPO-REQUESTEDENCODING     PIC S9(9) BINARY.
**    Requested character set identifier of VALUE
   15 MQIMPO-REQUESTEDCCSID        PIC S9(9) BINARY.
**    Returned encoding of VALUE
   15 MQIMPO-RETURNEDENCODING      PIC S9(9) BINARY.
**    Returned character set identifier of VALUE
   15 MQIMPO-RETURNEDCCSID         PIC S9(9) BINARY.
**    Reserved field
   15 MQIMPO-RESERVED1
**    Returned property name
   15 MQIMPO-RETURNEDNAME.
**     Address of variable length string
    20 MQIMPO-RETURNEDNAME-VSPTR    POINTER.
**     Offset of variable length string
    20 MQIMPO-RETURNEDNAME-VSOFFSET PIC S9(9) BINARY.
**     CCSID of variable length string
    20 MQIMPO-RETURNEDNAME-VSCCSID  PIC S9(9) BINARY.
**    Property data type as string
   15 MQIMPO-TYPESTRING            PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQIMPO

```
dcl
 1 MQIMPO based,
  3 StrucId            char(4),       /* Structure identifier */
  3 Version            fixed bin(31), /* Structure version number */
  3 Options            fixed bin(31), /* Options that control the
                                         action of MQINQMP */
  3 RequestedEncoding  fixed bin(31), /* Requested encoding of
                                         Value */
  3 RequestedCCSID     fixed bin(31), /* Requested character set
                                         identifier of Value */
  3 ReturnedEncoding   fixed bin(31), /* Returned encoding of
                                         Value */
  3 ReturnedCCSID      fixed bin(31), /* Returned character set
                                         identifier of Value */
  3 Reserved1          fixed bin(31), /* Reserved field */
  3 ReturnedName,                     /* Returned property name */
   5 ReturnedName_VSPtr      pointer,       /* Address of returned
                                               name */
   5 5 ReturnedName_VSOffset fixed bin(31), /* Offset of returned
                                               name */
    5 5 ReturnedName_VSCCSID  fixed bin(31), /* CCSID of returned
                                               name */
  3 TypeString         char(8);       /* Property data type as
                                         string */
```

Dichiarazione High Level Assembler per MQIMPO

```
MQIMPO                        DSECT
MQIMPO_STRUCID                DS   CL4  Structure identifier
MQIMPO_VERSION                DS   F    Structure version number
MQIMPO_OPTIONS                DS   F    Options that control the
*                                       action of MQINQMP
MQIMPO_REQUESTEDENCODING      DS   F    Requested encoding of VALUE
MQIMPO_REQUESTEDCCSID         DS   F    Requested character set
*                                       identifier of VALUE
MQIMPO_RETURNEDENCODING       DS   F    Returned encoding of VALUE
MQIMPO_RETURNEDCCSID          DS   F    Returned character set
*                                       identifier of VALUE
MQIMPO_RESERVED1              DS   F    Reserved field
MQIMPO_RETURNEDNAME           DS   0F   Force fullword alignment
MQIMPO_RETURNEDNAME_VSPTR     DS   F    Address of returned name
MQIMPO_RETURNEDNAME_VSOFFSET  DS   F    Offset of returned name
MQIMPO_RETURNEDNAME_VSLENGTH  DS   F    Length of returned name
MQIMPO_RETURNEDNAME_VSCCSID   DS   F    CCSID of returned name
MQIMPO_RETURNEDNAME_LENGTH    EQU  *-MQIMPO_RETURNEDNAME
                              ORG  MQIMPO_RETURNEDNAME
MQIMPO_RETURNEDNAME_AREA      DS   CL(MQIMPO_RETURNEDNAME_LENGTH)
*
MQIMPO_TYPESTRING             DS   CL8  Property data type as string
MQIMPO_LENGTH                 EQU  *-MQIMPO
MQIMPO_AREA                   DS   CL(MQIMPO_LENGTH)
```

StrucId (MQCHAR4) per MQIMPO

Questo è l'identificativo della struttura delle opzioni della proprietà del messaggio di interrogazione. È sempre un campo di immissione. Il valore è MQIMPO_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQIMPO
Identificativo per la struttura di opzioni della proprietà del messaggio di interrogazione.

Per il linguaggio di programmazione C, viene definita anche la costante MQIMPO_STRUC_ID_ARRAY. Ha lo stesso valore di MQIMPO_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQIMPO

Interroga struttura delle opzioni della proprietà del messaggio - Campo Versione

Questo è il numero di versione della struttura. Il valore deve essere:

MQIMPO_VERSION_1
Numero di versione per la struttura di opzioni della proprietà del messaggio di richiesta.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQIMPO_CURRENT_
La versione corrente della struttura di opzioni della proprietà del messaggio di interrogazione.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQIMPO_VERSION_1.

Opzioni (MQLONG) per MQIMPO

Struttura delle opzioni di proprietà del messaggio di interrogazione - Campo Opzioni

Le opzioni riportate di seguito controllano l'operazione di MQINQMP. È possibile specificare una o più di queste opzioni. Per specificare più di un'opzione, aggiungere i valori insieme (non aggiungere la stessa costante più di una volta) o combinare i valori utilizzando l'operazione OR bit per bit (se il linguaggio di programmazione supporta le operazioni bit).

Vengono annotate le combinazioni di opzioni non valide; tutte le altre combinazioni sono valide.

Opzioni dati valore: le seguenti opzioni si riferiscono all'elaborazione dei dati valore quando la proprietà viene richiamata dal messaggio.

VALORE MQIMPO_CONVERT_
Questa opzione richiede che il valore della proprietà sia convertito in modo da essere conforme ai valori di *RequestedCCSID* e *RequestedEncoding* specificati prima della chiamata MQINQMP restituisce il valore della proprietà nell'area *Value* .

- Se la conversione ha esito positivo, i campi *ReturnedCCSID* e *ReturnedEncoding* vengono impostati sullo stesso valore di *RequestedCCSID* e *RequestedEncoding* al ritorno dalla chiamata MQINQMP.
- Se la conversione non riesce, ma la chiamata MQINQMP viene altrimenti completata senza errori, il valore della proprietà viene restituito non convertito.

  Se la proprietà è una stringa, i campi *ReturnedCCSID* e *ReturnedEncoding* sono impostati sulla serie di caratteri e sulla codifica della stringa non convertita.

  Il codice di completamento è MQCC_WARNING in questo caso, con codice motivo MQRC_PROP_VALUE_NOT_CONVERTED. Il cursore della proprietà è avanzato rispetto alla proprietà restituita.

Se il valore della proprietà si espande durante la conversione e supera la dimensione del parametro **Value** , il valore viene restituito non convertito, con il codice di completamento MQCC_FAILED; il codice motivo è impostato su MQRC_PROPERTY_VALUE_TOO_BIG.

Il parametro **DataLength** della chiamata MQINQMP restituisce la lunghezza in cui il valore della proprietà sarebbe stato convertito, per consentire all'applicazione di determinare la dimensione del buffer richiesta per contenere il valore della proprietà convertita. Il cursore della proprietà non viene modificato.

Questa opzione richiede inoltre che:

- Se il nome della proprietà contiene un carattere jolly e
- Il campo *ReturnedName* viene inizializzato con un indirizzo o uno scostamento per il nome restituito,

il nome restituito viene convertito per essere conforme ai valori *RequestedCCSID* e *RequestedEncoding* .

- Se la conversione ha esito positivo, il campo *VSCCSID* di *ReturnedName* e la codifica del nome restituito vengono impostati sul valore di input di *RequestedCCSID* e *RequestedEncoding*.
- Se la conversione non riesce, ma la chiamata MQINQMP viene altrimenti completata senza errori o avvertenze, il nome restituito non viene convertito. Il codice di completamento è MQCC_WARNING in questo caso, con codice motivo MQRC_PROP_NAME_NOT_CONVERT.

  Il cursore della proprietà è avanzato rispetto alla proprietà restituita. MQRC_PROP_VALUE_NOT_CONVERTED viene restituito se il valore e il nome non vengono convertiti.

Se il nome restituito si espande durante la conversione e supera la dimensione del campo *VSBufsize* di *RequestedName*, la stringa restituita viene lasciata non convertita, con codice di completamento MQCC_FAILED e il codice motivo è impostato su MQRC_PROPERTY_NAME_TOO_BIG.

Il campo *VSLength* della struttura MQCHARV restituisce la lunghezza in cui il valore della proprietà sarebbe stato convertito, in modo da consentire all'applicazione di determinare la dimensione del buffer richiesto per contenere il valore della proprietà convertita. Il cursore della proprietà non viene modificato.

TIPO_CONVER_MQIMPO
Questa opzione richiede che il valore della proprietà venga convertito dal tipo di dati corrente, nel tipo di dati specificato nel parametro **Type** della chiamata MQINQMP.

- Se la conversione riesce, il parametro **Type** non viene modificato al ritorno della chiamata MQINQMP.
- Se la conversione ha esito negativo, ma la chiamata MQINQMP viene altrimenti completata senza errori, la chiamata ha esito negativo con il motivo MQRC_PROP_CONV_NOT_SUPPORTED. Il cursore della proprietà non viene modificato.

Se la conversione del tipo di dati causa l'espansione del valore durante la conversione e il valore convertito supera la dimensione del parametro **Value** , il valore viene restituito non convertito, con codice di completamento MQCC_FAILED e il codice motivo è impostato su MQRC_PROPERTY_VALUE_TOO_BIG.

Il parametro **DataLength** della chiamata MQINQMP restituisce la lunghezza in cui il valore della proprietà sarebbe stato convertito, per consentire all'applicazione di determinare la dimensione del buffer richiesta per contenere il valore della proprietà convertita. Il cursore della proprietà non viene modificato.

Se il valore del parametro **Type** della chiamata MQINQMP non è valido, la chiamata ha esito negativo con motivo MQRC_PROPERTY_TYPE_ERROR.

Se la conversione del tipo di dati richiesto non è supportata, la chiamata non riesce con motivo MQRC_PROP_CONV_NOT_SUPPORTED. Sono supportate le seguenti conversioni del tipo di dati:

Tabella 499. Conversioni di tipi di dati supportate

Tipo dati proprietà	Tipi di dati di destinazione supportati
BOOLEAN MQTIPO	MQTYPE_STRING, MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64
MQTYPE_BYTE_STRING	MQTYPE_STRING
MQTYPE_INT8	MQTYPE_STRING, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64
MQTYPE_INT16	MQTYPE_STRING, MQTYPE_INT32, MQTYPE_INT64
MQTYPE_INT32	MQTYPE_STRING, MQTYPE_INT64
MQTYPE_INT64	MQTYPE_STRING
MQTYPE_FLOAT32	MQTYPE_STRING, MQTYPE_FLOAT64
MQTYPE_FLOAT64	MQTYPE_STRING
MQTYPE_STRING	MQTYPE_BOOLEAN, MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64, MQTYPE_FLOAT32, MQTYPE_FLOAT64
MQTYPE_NULL	Nessuna

Le regole generali che disciplinano le conversioni supportate sono le seguenti:

- I valori delle proprietà numeriche possono essere convertiti da un tipo di dati ad un altro, purché non si perda alcun dato durante la conversione.

  Ad esempio, il valore di una proprietà con tipo di dati MQTYPE_INT32 può essere convertito in un valore con tipo di dati MQTYPE_INT64, ma non può essere convertito in un valore con tipo di dati MQTYPE_INT16.
- Un valore di proprietà di qualsiasi tipo di dati può essere convertito in una stringa.
- Un valore della proprietà stringa può essere convertito in qualsiasi altro tipo di dati, a condizione che la stringa sia formattata correttamente per la conversione. Se un'applicazione tenta di convertire un valore della proprietà della stringa che non è formattato correttamente, IBM MQ restituisce il codice motivo MQRC_PROP_NUMBER_FORMAT_ERROR.
- Se un'applicazione tenta una conversione non supportata, IBM MQ restituisce il codice motivo MQRC_PROP_CONV_NOT_SUPPORTED.

Le regole specifiche per convertire un valore di proprietà da un tipo di dati ad un altro sono le seguenti:

- Quando si converte un valore della proprietà MQTYPE_BOOLEAN in una stringa, il valore TRUE viene convertito nella stringa "TRUE" e il valore false viene convertito nella stringa "FALSE".

- Quando si converte un valore della proprietà MQTYPE_BOOLEAN in un tipo di dati numerico, il valore TRUE viene convertito in uno e il valore FALSE viene convertito in zero.
- Quando si converte un valore della proprietà stringa in un valore MQTYPE_BOOLEAN, la stringa "TRUE" o "1" viene convertita in TRUE e la stringa "FALSE" o "0" viene convertita in FALSE.

  Notare che i termini "TRUE" e "FALSE" non sono sensibili al maiuscolo / minuscolo.

  Qualsiasi altra stringa non può essere convertita; IBM MQ restituisce il codice motivo MQRC_PROP_NUMBER_FORMAT_ERROR.
- Quando si converte un valore della proprietà stringa in un valore con tipo di dati MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32 o MQTYPE_INT64, la stringa deve avere il seguente formato:

  ```
  [blanks][sign]digits
  ```

  I significati dei componenti della stringa sono i seguenti:

  blanks
  Caratteri vuoti iniziali facoltativi

  sign
  Un segno più (+) o segno meno (-) facoltativo.

  digits
  Una sequenza contigua di caratteri cifra (0-9). Deve essere presente almeno un carattere cifra.

  Dopo la sequenza di caratteri cifra, la stringa può contenere altri caratteri che non sono caratteri cifra, ma la conversione si interrompe non appena viene raggiunto il primo di questi caratteri. Si presuppone che la stringa rappresenti un numero intero decimale.

  IBM MQ restituisce il codice motivo MQRC_PROP_NUMBER_FORMAT_ERROR se la stringa non è formattata correttamente.
- Quando si converte un valore della proprietà stringa in un valore con tipo di dati MQTYPE_FLOAT32 o MQTYPE_FLOAT64, la stringa deve avere il seguente formato:

  ```
  [blanks][sign]digits[.digits][e_char[e_sign]e_digits]
  ```

  I significati dei componenti della stringa sono i seguenti:

  blanks
  Caratteri vuoti iniziali facoltativi

  sign
  Un segno più (+) o segno meno (-) facoltativo.

  digits
  Una sequenza contigua di caratteri cifra (0-9). Deve essere presente almeno un carattere cifra.

  e_char
  Un carattere esponente, che è "E" o "e".

  e_sign
  Un segno più (+) o un segno meno (-) facoltativo per l'esponente.

  e_digits
  Una sequenza contigua di caratteri cifra (0-9) per l'esponente. Deve essere presente almeno un carattere cifra se la stringa contiene un carattere esponente.

  Dopo la sequenza di caratteri cifra, o i caratteri facoltativi che rappresentano un esponente, la stringa può contenere altri caratteri che non sono caratteri cifra, ma la conversione si interrompe non appena viene raggiunto il primo di questi caratteri. Si presuppone che la stringa rappresenti un numero a virgola mobile decimale con un esponente che sia una potenza di 10.

  IBM MQ restituisce il codice motivo MQRC_PROP_NUMBER_FORMAT_ERROR se la stringa non è formattata correttamente.

- Quando si converte un valore di proprietà numerico in una stringa, il valore viene convertito nella rappresentazione di stringa del valore come numero decimale, non la stringa che contiene il carattere ASCII per tale valore. Ad esempio, il valore intero 65 viene convertito nella stringa "65", non nella stringa "A".
- Quando si converte un valore della proprietà stringa di byte in una stringa, ogni byte viene convertito in due caratteri esadecimali che rappresentano il byte. Ad esempio, l'array di byte {0xF1, 0x12, 0x00, 0xFF} viene convertito nella stringa "F11200FF".

QUERY MQIMPO_LENGTH
Interrogare il tipo e la lunghezza del valore della proprietà. La lunghezza viene restituita dal parametro **DataLength** della chiamata MQINQMP. Il valore della proprietà non viene restituito.

Se viene specificato un buffer **ReturnedName** , il campo *VSLength* della struttura MQCHARV viene riempito con la lunghezza del nome della proprietà. Il nome della proprietà non viene restituito.

Opzioni di iterazione: le seguenti opzioni si riferiscono all'iterazione sulle proprietà, utilizzando un nome con un carattere jolly

MQIMPO_INQ_FIRST
Analizzare la prima proprietà che corrisponde al nome specificato. Dopo questa chiamata, viene stabilito un cursore sulla proprietà restituita.

Questo è il valore predefinito.

L'opzione MQIMPO_INQ_PROP_UNDER_CURSOR può essere successivamente utilizzata con una chiamata MQINQMP, se richiesta, per richiedere nuovamente la stessa proprietà.

Notare che è presente un solo cursore della proprietà; pertanto, se il nome della proprietà, specificato nella chiamata MQINQMP, cambia, il cursore viene reimpostato.

Questa opzione non è valida con una delle seguenti opzioni:

MQIMPO_INQ_NEXT
MQIMPO_INQ_PROP_UNDER_CURSOR

MQIMPO_INQ_NEXT
Interroga la proprietà successiva che corrisponde al nome specificato, continuando la ricerca dal cursore della proprietà. Il cursore è avanzato alla proprietà restituita.

Se questa è la prima chiamata MQINQMP per il nome specificato, viene restituita la prima proprietà che corrisponde al nome specificato.

L'opzione MQIMPO_INQ_PROP_UNDER_CURSOR può essere successivamente utilizzata con una chiamata MQINQMP, se necessario, per analizzare nuovamente la stessa proprietà.

Se la proprietà sotto il cursore è stata eliminata, MQINQMP restituisce la proprietà corrispondente successiva a quella che è stata eliminata.

Se viene aggiunta una proprietà che corrisponde al carattere jolly, mentre è in corso un'iterazione, la proprietà potrebbe essere restituita o meno durante il completamento dell'iterazione. La proprietà viene restituita quando l'iterazione viene riavviata utilizzando MQIMPO_INQ_FIRST.

Una proprietà corrispondente al carattere jolly che è stato eliminato, mentre l'iterazione era in corso, non viene restituita dopo la sua eliminazione.

Questa opzione non è valida con una delle seguenti opzioni:

MQIMPO_INQ_FIRST
MQIMPO_INQ_PROP_UNDER_CURSOR

MQIMPO_INQ_PROP_UNDER_CURSOR
Richiamare il valore della proprietà indicata dal cursore della proprietà. La proprietà a cui punta il cursore della proprietà è quella che è stata interrogata l'ultima volta, utilizzando l'opzione MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

Il cursore della proprietà viene reimpostato quando viene riutilizzato l'handle del messaggio, quando l'handle del messaggio viene specificato nel campo *MsgHandle* di MQGMO su una chiamata MQGET

o quando l'handle del messaggio viene specificato nei campi *OriginalMsgHandle* o *NewMsgHandle* della struttura MQPMO su una chiamata MQPUT.

Se questa opzione viene utilizzata quando il cursore della proprietà non è ancora stato stabilito o se la proprietà indicata dal cursore della proprietà è stata eliminata, la chiamata ha esito negativo con codice di completamento MQCC_FAILED e motivo MQRC_PROPERTY_NOT_AVAILABLE.

Questa opzione non è valida con una delle seguenti opzioni:

MQIMPO_INQ_FIRST
MQIMPO_INQ_NEXT

Se nessuna delle opzioni precedentemente descritte è richiesta, è possibile utilizzare la seguente opzione:

MQIMPO_NONE
Utilizzare questo valore per indicare che non sono state specificate altre opzioni; tutte le opzioni assumono i propri valori predefiniti.

MQIMPO_NONE supporta la documentazione del programma; non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQIMPO_INQ_FIRST.

RequestedEncoding (MQLONG) per MQIMPO
Struttura delle opzioni della proprietà del messaggio di interrogazione - campo RequestedEncoding

Questa è la codifica in cui il valore della proprietà interrogata deve essere convertito quando viene specificato MQIMPO_CONVERT_VALUE o MQIMPO_CONVERT_TYPE.

Il valore iniziale di questo campo è MQENC_NATIVE.

RequestedCCSID (MQLONG) per MQIMPO
Struttura delle opzioni della proprietà del messaggio di interrogazione - Campo RequestedCCSID

La serie di caratteri in cui deve essere convertito il valore della proprietà richiesta se il valore è una stringa di caratteri. Questo è anche il set di caratteri in cui *ReturnedName* deve essere convertito quando viene specificato MQIMPO_CONVERT_VALUE o MQIMPO_CONVERT_TYPE.

Il valore iniziale di questo campo è MQCCSI_APPL.

ReturnedEncoding (MQLONG) per MQIMPO
Interroga la struttura delle opzioni della proprietà del messaggio - Campo ReturnedEncoding

Nell'output, questa è la codifica del valore restituito.

Se è stata specificata l'opzione MQIMPO_CONVERT_VALUE e la conversione ha avuto esito positivo, il campo *ReturnedEncoding* , alla restituzione, è lo stesso valore del valore trasmesso.

Il valore iniziale di questo campo è MQENC_NATIVE.

ReturnedCCSID (MQLONG) per MQIMPO
Interroga la struttura delle opzioni della proprietà del messaggio - Campo ReturnedCCSID

In fase di output, questa è la serie di caratteri del valore restituito se il parametro **Type** della chiamata MQINQMP è MQTYPE_STRING.

Se è stata specificata l'opzione MQIMPO_CONVERT_VALUE e la conversione ha avuto esito positivo, il campo *ReturnedCCSID* , alla restituzione, è lo stesso valore del valore trasmesso.

Il valore iniziale di questo campo è zero.

Reserved1 (MQCHAR) per MQIMPO

Questo è un campo riservato. Il valore iniziale di questo campo è un carattere vuoto (campo a 4 byte).

ReturnedName (MQCHARV) per MQIMPO

Struttura delle opzioni delle proprietà del messaggio - Campo ReturnedName

Il nome effettivo della proprietà interrogata.

In fase di input, è possibile passare un buffer di stringa utilizzando il campo *VSPtr* o *VSOffset* della struttura MQCHARV . La lunghezza del buffer della stringa viene specificato utilizzando il campo *VSBufsize* della struttura MQCHARV.

Al ritorno dalla chiamata MQINQMP, il buffer di stringa viene completato con il nome della proprietà che è stata interrogata, a condizione che il buffer di stringa sia stato sufficientemente lungo per contenere completamente il nome. Il campo *VSLength* della struttura MQCHARV viene riempito con la lunghezza del nome della proprietà. Il campo *VSCCSID* della struttura MQCHARV viene compilato per indicare la serie di caratteri del nome restituito, se la conversione del nome non è riuscita o meno.

Questo è un campo di immissione / emissione. Il valore iniziale di questo campo è MQCHARV_DEFAULT.

TypeString (MQCHAR8) per MQIMPO

Interroga la struttura delle opzioni della proprietà del messaggio - Campo TypeString

Una rappresentazione stringa del tipo di dati della proprietà.

Se la proprietà è stata specificata in un'intestazione MQRFH2 e l'attributo MQRFH2 `dt` non è riconosciuto, questo campo può essere utilizzato per determinare il tipo di dati della proprietà. *TypeString* viene restituito nella serie di caratteri codificata 1208 (UTF-8) ed è i primi otto byte del valore dell'attributo `dt` della proprietà che non è stato possibile riconoscere

Questo è sempre un campo di output. Il valore iniziale di questo campo è la stringa nulla nel linguaggio di programmazione C e 8 caratteri vuoti in altri linguaggi di programmazione.

MQMD - Descrittore messaggi

La struttura MQMD contiene le informazioni di controllo che accompagnano i dati dell'applicazione quando un messaggio viaggia tra le applicazioni mittente e ricevente. La struttura è un parametro di input/output nelle chiamate MQGET, MQPUT e MQPUT1 .

Disponibilità

Tutti i sistemi IBM MQ , più IBM MQ MQI clients connessi a tali sistemi.

Versione

La versione corrente di MQMD è MQMD_VERSION_2. Le applicazioni che devono essere portabili tra diversi ambienti devono garantire che la versione richiesta di MQMD sia supportata in tutti gli ambienti interessati. I campi che esistono solo nelle versioni più recenti della struttura sono identificati come tali nelle descrizioni che seguono.

I file di intestazione, COPY e INCLUDE forniti per i linguaggi di programmazione supportati, contengono la versione più recente di MQMD supportata dall'ambiente, ma con il valore iniziale del campo *Version* impostato su MQMD_VERSION_1. Per utilizzare i campi che non sono presenti nella struttura version-1 , l'applicazione deve impostare il campo *Version* sul numero di versione della versione richiesta.

È disponibile una dichiarazione per la struttura version-1 con nome MQMD1.

Serie di caratteri e codifica

I dati in MQMD devono essere nella serie di caratteri e nella codifica del gestore code locale; tali dati sono forniti dall'attributo del gestore code **CodedCharSetId** e MQENC_NATIVE. Tuttavia, se l'applicazione è in esecuzione come IBM MQ MQI client, la struttura deve essere nella serie di caratteri e nella codifica del client.

Se i gestori code di invio e di ricezione utilizzano diverse serie di caratteri o codifiche, i dati in MQMD vengono convertiti automaticamente. Non è necessario che l'applicazione converta MQMD.

Utilizzo di diverse versioni di MQMD

Un MQMD version-2 è equivalente all'utilizzo di un MQMD version-1 e anteponendo ai dati del messaggio una struttura MQMDE. Tuttavia, se tutti i campi nella struttura MQMDE hanno i valori predefiniti, MQMDE può essere omesso. Un MQMD version-1 più MQMDE vengono utilizzati come descritto:

- Nelle chiamate MQPUT e MQPUT1 , se l'applicazione fornisce un MQMD version-1 , l'applicazione può facoltativamente aggiungere un prefisso ai dati del messaggio con MQMDE, impostando il campo *Format* in MQMD su MQFMT_MD_EXTENSION per indicare che è presente un MQMDE. Se l'applicazione non fornisce un MQMDE, il gestore code assume i valori predefiniti per i campi in MQMDE.

  Nota: Molti dei campi presenti in MQMD version-2 ma non in MQMD version-1 sono campi di input / output nelle chiamate MQPUT e MQPUT1 . Tuttavia, il gestore code non restituisce alcun valore nei campi equivalenti in MQMDE sull'output delle chiamate MQPUT e MQPUT1 ; se l'applicazione richiede tali valori di output, deve utilizzare un MQMD version-2 .
- Nella chiamata MQGET, se l'applicazione fornisce un MQMD version-1 , il gestore code prefissa il messaggio restituito con un MQMDE, ma solo se uno o più campi in MQMDE hanno un valore non predefinito. Il campo *Format* in MQMD avrà il valore MQFMT_MD_EXTENSION per indicare che è presente un MQMDE.

I valori predefiniti che il gestore code utilizza per i campi in MQMDE sono uguali ai valori iniziali di tali campi, mostrati in Tabella 503 a pagina 486.

Quando un messaggio si trova su una coda di trasmissione, alcuni dei campi in MQMD sono impostati su valori particolari; consultare “MQXQH - Intestazione coda di trasmissione” a pagina 635 per i dettagli.

Contesto messaggio

Alcuni campi in MQMD contengono il contesto del messaggio. Esistono due tipi di contesto del messaggio: *contesto identità* e *contesto origine*. In genere:

- Il contesto di identità è correlato all'applicazione che *originariamente* ha inserito il messaggio
- Il contesto di origine si correla all'applicazione che *più recentemente* ha inserito il messaggio.

Queste due applicazioni possono essere la stessa applicazione, ma possono anche essere applicazioni diverse (ad esempio, quando un messaggio viene inoltrato da un'applicazione a un'altra).

Sebbene l'identità e il contesto di origine in genere abbiano i significati descritti, il contenuto di entrambi i tipi di campi di contesto in MQMD dipende dalle opzioni MQPMO_ * _CONTEXT specificate quando il messaggio viene inserito. Di conseguenza, il contesto di identità non è necessariamente correlato all'applicazione che ha originariamente inserito il messaggio e il contesto di origine non è necessariamente correlato all'applicazione che ha inserito più di recente il messaggio; dipende dalla progettazione della suite di applicazioni.

MCA (message channel agent) non modifica mai il contesto del messaggio. Gli MCA che ricevono i messaggi dai gestori code remoti utilizzano l'opzione di contesto MQPMO_SET_ALL_ALL_CONTEXT sulla chiamata MQPUT o MQPUT1 . Ciò consente all'MCA ricevente di conservare esattamente il contesto del messaggio che ha viaggiato con il messaggio dall'MCA mittente. Tuttavia, il risultato è che il contesto di origine non si riferisce a nessuno degli MCA che hanno inviato e ricevuto il messaggio. Il contesto di origine fa riferimento a un'applicazione precedente che ha inserito il messaggio. Se tutte le applicazioni intermedie hanno inoltrato il contesto del messaggio, il contesto di origine si riferisce all'applicazione di origine stessa.

Nelle descrizioni, i campi di contesto sono descritti come se fossero utilizzati come descritto precedentemente. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 500. Campi in MQMD per MQMD

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQMD	'MD'
Versione (numero versione struttura)	MQMD_VERSION_1	1
Report (opzioni per i messaggi di report)	MQRO_NONE	0
MsgType (tipo di messaggio)	MQMT_DATAGRAM	8
MQMD - Campo scadenza (durata messaggio)	MQEI_UNLIMITED	-1
MQMD - campo Feedback (feedback o codice motivo)	MQFB_NONE	0
Codifica (codifica numerica dei dati del messaggio)	MQEN_NATIVE	Dipende dall'ambiente
CodedCharSetId (identificativo della serie di caratteri dei dati del messaggio)	MQCCSI_Q_MGR	0
Formato (nome formato dei dati del messaggio)	MQFMT_NONE	Spazi
Priorità (priorità del messaggio)	MQPRI_PRIORITY_AS_Q_DEF	-1
Persistenza (Persistenza messaggio)	MQPER_PERSISTENCE_AS_Q_DEF	2
Campo MQMD - MsgId (identificativo del messaggio)	MQMI_NONE	Valori null
CorrelId (identificativo di correlazione)	MQCI_NONE	Valori null
BackoutCount (contatore backout)	Nessuna	0
ReplyToQ (nome della coda di risposta)	Nessuna	Stringa null o spazi vuoti
ReplyTo (nome del gestore code di risposta)	Nessuna	Stringa null o spazi vuoti
UserIdentifier (identificativo utente)	Nessuna	Stringa null o spazi vuoti
AccountingToken (token di account)	MQACT_NONE	Valori null
ApplIdentityData (dati dell'applicazione relativi all'identità)	Nessuna	Stringa null o spazi vuoti
PutApplTipo (tipo di applicazione che inserisce il messaggio)	MQAT_NO_CONTEXT	0
PutApplName (nome dell'applicazione che ha inserito il messaggio)	Nessuna	Stringa null o spazi vuoti
PutDate (data in cui è stato inserito il messaggio)	Nessuna	Stringa null o spazi vuoti

Tabella 500. Campi in MQMD per MQMD (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
PutTime (ora in cui è stato inserito il messaggio)	Nessuna	Stringa null o spazi vuoti
ApplOriginData (dati dell'applicazione relativi all'origine)	Nessuna	Stringa null o spazi vuoti
Nota: I restanti campi vengono ignorati se *Version* è minore di MQMD_VERSION_2.		
GroupId (identificativo gruppo)	MQGI_NONE	Valori null
MsgSeqNumero (numero di sequenza del messaggio logico all'interno del gruppo)	Nessuna	1
Offset (offset dei dati nel messaggio fisico dall'inizio del messaggio logico)	Nessuna	0
Campo MQMD - MsgFlags (indicatori di messaggio)	MQMF_NONE	0
OriginalLength (lunghezza del messaggio originale)	MQOL_NON DEFINITO	-1

Note:

1. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.
2. Nel linguaggio di programmazione C, la variabile macroMQMD_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura:

```
MQMD MyMD = {MQMD_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQMD

```
typedef struct tagMQMD MQMD;
struct tagMQMD {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    Report;            /* Options for report messages */
  MQLONG    MsgType;           /* Message type */
  MQLONG    Expiry;            /* Message lifetime */
  MQLONG    Feedback;          /* Feedback or reason code */
  MQLONG    Encoding;          /* Numeric encoding of message data */
  MQLONG    CodedCharSetId;    /* Character set identifier of message
                                  data */
  MQCHAR8   Format;            /* Format name of message data */
  MQLONG    Priority;          /* Message priority */
  MQLONG    Persistence;       /* Message persistence */
  MQBYTE24  MsgId;             /* Message identifier */
  MQBYTE24  CorrelId;          /* Correlation identifier */
  MQLONG    BackoutCount;      /* Backout counter */
  MQCHAR48  ReplyToQ;          /* Name of reply queue */
  MQCHAR48  ReplyToQMgr;       /* Name of reply queue manager */
  MQCHAR12  UserIdentifier;    /* User identifier */
  MQBYTE32  AccountingToken;   /* Accounting token */
  MQCHAR32  ApplIdentityData;  /* Application data relating to
                                  identity */
  MQLONG    PutApplType;       /* Type of application that put the
                                  message */
  MQCHAR28  PutApplName;       /* Name of application that put the
                                  message */
  MQCHAR8   PutDate;           /* Date when message was put */
```

```
  MQCHAR8   PutTime;           /* Time when message was put */
  MQCHAR4   ApplOriginData;    /* Application data relating to origin */
  MQBYTE24  GroupId;           /* Group identifier */
  MQLONG    MsgSeqNumber;      /* Sequence number of logical message
                                  within group */
  MQLONG    Offset;            /* Offset of data in physical message
                                  from start of logical message */
  MQLONG    MsgFlags;          /* Message flags */
  MQLONG    OriginalLength;    /* Length of original message */
};
```

Dichiarazione COBOL per MQMD

```
**   MQMD structure
  10 MQMD.
**    Structure identifier
   15 MQMD-STRUCID          PIC X(4).
**    Structure version number
   15 MQMD-VERSION          PIC S9(9) BINARY.
**    Options for report messages
   15 MQMD-REPORT           PIC S9(9) BINARY.
**    Message type
   15 MQMD-MSGTYPE          PIC S9(9) BINARY.
**    Message lifetime
   15 MQMD-EXPIRY           PIC S9(9) BINARY.
**    Feedback or reason code
   15 MQMD-FEEDBACK         PIC S9(9) BINARY.
**    Numeric encoding of message data
   15 MQMD-ENCODING         PIC S9(9) BINARY.
**    Character set identifier of message data
   15 MQMD-CODEDCHARSETID   PIC S9(9) BINARY.
**    Format name of message data
   15 MQMD-FORMAT           PIC X(8).
**    Message priority
   15 MQMD-PRIORITY         PIC S9(9) BINARY.
**    Message persistence
   15 MQMD-PERSISTENCE      PIC S9(9) BINARY.
**    Message identifier
   15 MQMD-MSGID            PIC X(24).
**    Correlation identifier
   15 MQMD-CORRELID         PIC X(24).
**    Backout counter
   15 MQMD-BACKOUTCOUNT     PIC S9(9) BINARY.
**    Name of reply queue
   15 MQMD-REPLYTOQ         PIC X(48).
**    Name of reply queue manager
   15 MQMD-REPLYTOQMGR      PIC X(48).
**    User identifier
   15 MQMD-USERIDENTIFIER   PIC X(12).
**    Accounting token
   15 MQMD-ACCOUNTINGTOKEN  PIC X(32).
**    Application data relating to identity
   15 MQMD-APPLIDENTITYDATA PIC X(32).
**    Type of application that put the message
   15 MQMD-PUTAPPLTYPE      PIC S9(9) BINARY.
**    Name of application that put the message
   15 MQMD-PUTAPPLNAME      PIC X(28).
**    Date when message was put
   15 MQMD-PUTDATE          PIC X(8).
**    Time when message was put
   15 MQMD-PUTTIME          PIC X(8).
**    Application data relating to origin
   15 MQMD-APPLORIGINDATA   PIC X(4).
**    Group identifier
   15 MQMD-GROUPID          PIC X(24).
**    Sequence number of logical message within group
   15 MQMD-MSGSEQNUMBER     PIC S9(9) BINARY.
**    Offset of data in physical message from start of logical message
   15 MQMD-OFFSET           PIC S9(9) BINARY.
**    Message flags
   15 MQMD-MSGFLAGS         PIC S9(9) BINARY.
**    Length of original message
   15 MQMD-ORIGINALLENGTH   PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQMD

```
dcl
```

```
 1 MQMD based,
  3 StrucId          char(4),       /* Structure identifier */
  3 Version          fixed bin(31), /* Structure version number */
  3 Report           fixed bin(31), /* Options for report messages */
  3 MsgType          fixed bin(31), /* Message type */
  3 Expiry           fixed bin(31), /* Message lifetime */
  3 Feedback         fixed bin(31), /* Feedback or reason code */
  3 Encoding         fixed bin(31), /* Numeric encoding of message
                                       data */
  3 CodedCharSetId   fixed bin(31), /* Character set identifier of
                                       message data */
  3 Format           char(8),       /* Format name of message data */
  3 Priority         fixed bin(31), /* Message priority */
  3 Persistence      fixed bin(31), /* Message persistence */
  3 MsgId            char(24),      /* Message identifier */
  3 CorrelId         char(24),      /* Correlation identifier */
  3 BackoutCount     fixed bin(31), /* Backout counter */
  3 ReplyToQ         char(48),      /* Name of reply queue */
  3 ReplyToQMgr      char(48),      /* Name of reply queue manager */
  3 UserIdentifier   char(12),      /* User identifier */
  3 AccountingToken  char(32),      /* Accounting token */
  3 ApplIdentityData char(32),      /* Application data relating to
                                       identity */
  3 PutApplType      fixed bin(31), /* Type of application that put the
                                       message */
  3 PutApplName      char(28),      /* Name of application that put the
                                       message */
  3 PutDate          char(8),       /* Date when message was put */
  3 PutTime          char(8),       /* Time when message was put */
  3 ApplOriginData   char(4),       /* Application data relating to
                                       origin */
  3 GroupId          char(24),      /* Group identifier */
  3 MsgSeqNumber     fixed bin(31), /* Sequence number of logical
                                       message within group */
  3 Offset           fixed bin(31), /* Offset of data in physical
                                       message from start of logical
                                       message */
  3 MsgFlags         fixed bin(31), /* Message flags */
  3 OriginalLength   fixed bin(31); /* Length of original message */
```

Dichiarazione High Level Assembler per MQMD

```
MQMD                  DSECT
MQMD_STRUCID          DS   CL4   Structure identifier
MQMD_VERSION          DS   F     Structure version number
MQMD_REPORT           DS   F     Options for report messages
MQMD_MSGTYPE          DS   F     Message type
MQMD_EXPIRY           DS   F     Message lifetime
MQMD_FEEDBACK         DS   F     Feedback or reason code
MQMD_ENCODING         DS   F     Numeric encoding of message data
MQMD_CODEDCHARSETID   DS   F     Character set identifier of message
*                                data
MQMD_FORMAT           DS   CL8   Format name of message data
MQMD_PRIORITY         DS   F     Message priority
MQMD_PERSISTENCE      DS   F     Message persistence
MQMD_MSGID            DS   XL24  Message identifier
MQMD_CORRELID         DS   XL24  Correlation identifier
MQMD_BACKOUTCOUNT     DS   F     Backout counter
MQMD_REPLYTOQ         DS   CL48  Name of reply queue
MQMD_REPLYTOQMGR      DS   CL48  Name of reply queue manager
MQMD_USERIDENTIFIER   DS   CL12  User identifier
MQMD_ACCOUNTINGTOKEN  DS   XL32  Accounting token
MQMD_APPLIDENTITYDATA DS   CL32  Application data relating to identity
MQMD_PUTAPPLTYPE      DS   F     Type of application that put the
*                                message
MQMD_PUTAPPLNAME      DS   CL28  Name of application that put the
*                                message
MQMD_PUTDATE          DS   CL8   Date when message was put
MQMD_PUTTIME          DS   CL8   Time when message was put
MQMD_APPLORIGINDATA   DS   CL4   Application data relating to origin
MQMD_GROUPID          DS   XL24  Group identifier
MQMD_MSGSEQNUMBER     DS   F     Sequence number of logical message
*                                within group
MQMD_OFFSET           DS   F     Offset of data in physical message
*                                from start of logical message
MQMD_MSGFLAGS         DS   F     Message flags
MQMD_ORIGINALLENGTH   DS   F     Length of original message
*
MQMD_LENGTH           EQU  *-MQMD
```

```
                        ORG  MQMD
MQMD_AREA               DS   CL(MQMD_LENGTH)
```

Dichiarazione Visual Basic per MQMD

```
Type MQMD
  StrucId          As String*4  'Structure identifier'
  Version          As Long      'Structure version number'
  Report           As Long      'Options for report messages'
  MsgType          As Long      'Message type'
  Expiry           As Long      'Message lifetime'
  Feedback         As Long      'Feedback or reason code'
  Encoding         As Long      'Numeric encoding of message data'
  CodedCharSetId   As Long      'Character set identifier of message'
                                'data'
  Format           As String*8  'Format name of message data'
  Priority         As Long      'Message priority'
  Persistence      As Long      'Message persistence'
  MsgId            As MQBYTE24  'Message identifier'
  CorrelId         As MQBYTE24  'Correlation identifier'
  BackoutCount     As Long      'Backout counter'
  ReplyToQ         As String*48 'Name of reply queue'
  ReplyToQMgr      As String*48 'Name of reply queue manager'
  UserIdentifier   As String*12 'User identifier'
  AccountingToken  As MQBYTE32  'Accounting token'
  ApplIdentityData As String*32 'Application data relating to identity'
  PutApplType      As Long      'Type of application that put the'
                                'message'
  PutApplName      As String*28 'Name of application that put the'
                                'message'
  PutDate          As String*8  'Date when message was put'
  PutTime          As String*8  'Time when message was put'
  ApplOriginData   As String*4  'Application data relating to origin'
  GroupId          As MQBYTE24  'Group identifier'
  MsgSeqNumber     As Long      'Sequence number of logical message'
                                'within group'
  Offset           As Long      'Offset of data in physical message'
                                'from start of logical message'
  MsgFlags         As Long      'Message flags'
  OriginalLength   As Long      'Length of original message'
End Type
```

StrucId (MQCHAR4) per MQMD

Questo è l'identificativo della struttura del descrittore del messaggio. È sempre un campo di immissione. Il valore è MQMD_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQMD
Identificativo per la struttura descrittore del messaggio.

Per il linguaggio di programmazione C, viene anche definita la costante MQMD_STRUC_ID_ARRAY. Questo ha lo stesso valore di MQMD_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQMD

Questo è il numero di versione della struttura e deve essere uno dei seguenti:

MQMD_VERSION_1
Struttura descrittore del messaggio Version-1 .

Questa versione è supportata in tutti gli ambienti.

MQMD_VERSION_2
Struttura del descrittore del messaggio Version-2 .

Questa versione è supportata in tutti gli ambienti IBM MQ V6.0 e successivi, oltre a IBM MQ MQI clients connessi a questi sistemi.

Nota: Quando viene utilizzato un MQMD version-2 , il gestore code esegue ulteriori controlli su qualsiasi struttura di intestazione MQ che potrebbe essere presente all'inizio dei dati del messaggio dell'applicazione; per ulteriori informazioni, consultare le note di utilizzo per la chiamata MQPUT.

I campi esistenti solo nella versione più recente della struttura vengono identificati come tali nelle descrizioni dei campi. La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQMD_CURRENT_
Versione corrente della struttura del descrittore del messaggio.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQMD_VERSION_1.

Report (MQLONG) per MQMD

Un messaggio di report è un messaggio relativo ad un altro messaggio, utilizzato per informare un'applicazione di eventi previsti o imprevisti correlati al messaggio originale. Il campo *Report* consente all'applicazione che invia il messaggio originale di specificare quali messaggi di report sono richiesti, se i dati del messaggio dell'applicazione devono essere inclusi in essi e (sia per i report che per le risposte) come devono essere impostati gli identificativi del messaggio e della correlazione nel report o nel messaggio di risposta. È possibile richiedere uno o tutti (o nessuno) dei seguenti tipi di messaggi di report:

- Eccezione
- Scadenza
- Conferma all'arrivo (COA)
- Conferma alla consegna (COD)
- PAN (positive action notification)
- Notifica azione negativa (NAN)

È possibile specificare una o più di queste opzioni. Per specificare più di un'opzione, aggiungere i valori insieme (non aggiungere la stessa costante più di una volta) o combinare i valori utilizzando l'operazione OR bit per bit (se il linguaggio di programmazione supporta le operazioni bit).

L'applicazione che riceve il messaggio di report può determinare il motivo per cui il report è stato generato esaminando il campo *Feedback* in MQMD; consultare il campo *Feedback* per ulteriori dettagli.

L'utilizzo delle opzioni del report quando si immette un messaggio in un argomento può causare la generazione e l'invio all'applicazione di zero, uno o più messaggi di report. Ciò è dovuto al fatto che il messaggio di pubblicazione può essere inviato a zero, una o molte applicazioni di sottoscrizione.

Opzioni di eccezione: specificare una delle opzioni elencate per richiedere un messaggio di report di eccezione.

MQRO_ECCEZIONE
Un agent del canale dei messaggi genera questo tipo di report quando un messaggio viene inviato a un altro gestore code e non è possibile consegnare il messaggio alla coda di destinazione specificata. Ad esempio, la coda di destinazione o una coda di trasmissione intermedia potrebbe essere piena oppure il messaggio potrebbe essere troppo grande per la coda.

La creazione del messaggio di report di eccezioni dipende dalla persistenza del messaggio originale e dalla velocità del canale del messaggio (normale o veloce) attraverso cui il messaggio originale viaggia:

- Per tutti i messaggi persistenti e per i messaggi non persistenti che viaggiano attraverso i normali canali di messaggi, il report di eccezione viene generato solo se l'azione specificata dall'applicazione mittente per la condizione di errore può essere completata correttamente. L'applicazione mittente può specificare una delle seguenti azioni per controllare la disposizione del messaggio originale quando si verifica la condizione di errore:
  - MQRO_DEAD_LETTER_Q (posiziona il messaggio originale nella coda di messaggi non instradabili).
  - MQRO_DISCARD_MSG (elimina il messaggio originale).

  Se l'azione specificata dall'applicazione mittente non può essere completata correttamente, il messaggio originale viene lasciato nella coda di trasmissione e non viene generato alcun messaggio di report di eccezione.

- Per i messaggi non persistenti che viaggiano attraverso i canali di messaggi veloci, il messaggio originale viene rimosso dalla coda di trasmissione e il report di eccezione generato *anche se* l'azione specificata per la condizione di errore non può essere completata correttamente. Ad esempio, se viene specificato MQRO_DEAD_LETTER_Q, ma il messaggio originale non può essere inserito nella coda di messaggi non recapitabili perché tale coda è piena, viene generato il messaggio di report di eccezione e il messaggio originale viene eliminato.

  Per ulteriori informazioni sui canali di messaggi normali e veloci, consultare Velocità dei messaggi non persistenti (NPMSPEED).

Non viene generato un report di eccezione se l'applicazione che ha inserito il messaggio originale può ricevere una notifica sincrona del problema mediante il codice motivo restituito dalla chiamata MQPUT o MQPUT1 .

Le applicazioni possono anche inviare report di eccezione, per indicare che un messaggio non può essere elaborato (ad esempio, perché si tratta di una transazione di addebito che causerebbe il superamento del limite di credito del conto).

I dati del messaggio originale non sono inclusi nel messaggio di report.

Non specificare più di uno tra MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA e MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_EXCEPTION_WITH_DATA
È uguale a MQRO_EXCEPTION, tranne che i primi 100 byte dei dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio del report. Se il messaggio originale contiene una o più strutture di intestazione MQ , vengono incluse nel messaggio di report, in aggiunta ai 100 byte dei dati dell'applicazione.

Non specificare più di uno tra MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA e MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_EXCEPTION_WITH_FULL_DATA
Sono richiesti report di eccezione con dati completi.

È uguale a MQRO_EXCEPTION, tranne che tutti i dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio del report.

Non specificare più di uno tra MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA e MQRO_EXCEPTION_WITH_FULL_DATA.

Opzioni di scadenza: specificare una delle opzioni elencate per richiedere un messaggio di report di scadenza.

SCADENZA_MQRO
Questo tipo di report viene generato dal gestore code se il messaggio viene eliminato prima del recapito a un'applicazione perché è trascorso il tempo di scadenza (consultare il campo *Expiry*). Se questa opzione non è impostata, non viene generato alcun messaggio di report se un messaggio viene eliminato per questo motivo (anche se si specifica una delle opzioni MQRO_EXCEPTION_ *).

I dati del messaggio originale non sono inclusi nel messaggio di report.

Non specificare più di uno tra MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA e MQRO_EXPIRATION_WITH_FULL_DATA.

MQRO_EXPIRATION_WITH_DATA
È uguale a MQRO_EXPIRATION, tranne che i primi 100 byte dei dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio del prospetto. Se il messaggio originale contiene una o più strutture di intestazione MQ , vengono incluse nel messaggio di report, in aggiunta ai 100 byte dei dati dell'applicazione.

Non specificare più di uno tra MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA e MQRO_EXPIRATION_WITH_FULL_DATA.

MQRO_EXPIRATION_WITH_FULL_DATA
È uguale a MQRO_EXPIRATION, tranne che tutti i dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio di report.

Non specificare più di uno tra MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA e MQRO_EXPIRATION_WITH_FULL_DATA.

Opzioni di conferma all'arrivo: specificare una delle opzioni elencate per richiedere un messaggio di report di conferma all'arrivo.

COA MQRO

Questo tipo di report viene generato dal gestore code proprietario della coda di destinazione quando il messaggio viene inserito nella coda di destinazione. I dati del messaggio originale non sono inclusi nel messaggio di report.

Se il messaggio viene inserito come parte di un'unità di lavoro e la coda di destinazione è una coda locale, il messaggio di report COA generato dal gestore code può essere richiamato solo se è stato eseguito il commit dell'unità di lavoro.

Un report COA non viene generato se il campo *Format* nel descrittore del messaggio è MQFMT_XMIT_Q_HEADER o MQFMT_DEAD_LETTER_HEADER. Ciò impedisce la creazione di un report COA se il messaggio viene inserito in una coda di trasmissione o non è distribuibile e viene inserito in una coda di messaggi non recapitabili.

Nel caso di una coda bridge IMS , il report COA viene generato quando il messaggio raggiunge la coda IMS (riconoscimento ricevuto da IMS) e non quando il messaggio viene inserito nella coda bridge MQ . Ciò significa che se IMS non è attivo, non viene generato alcun report COA fino a quando IMS non viene avviato e un messaggio non viene accodato nella coda IMS .

L'utente che esegue un programma che inserisce un messaggio con MQMD.Report= MQRO_COA deve avere l'autorizzazione + passid sulla coda di risposte. Se l'utente non dispone dell'autorizzazione + passid, il messaggio di report COA non raggiunge la coda di risposta. Si è tentato di inserire il messaggio di report nella coda dei messaggi non recapitabili.

Non specificare più di uno tra MQRO_COA, MQRO_COA_WITH_DATA e MQRO_COA_WITH_FULL_DATA.

DATA_COA_WITH_MQRO
È uguale a MQRO_COA, tranne che i primi 100 byte dei dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio del report. Se il messaggio originale contiene una o più strutture di intestazione MQ , vengono incluse nel messaggio di report, in aggiunta ai 100 byte dei dati dell'applicazione.

Non specificare più di uno tra MQRO_COA, MQRO_COA_WITH_DATA e MQRO_COA_WITH_FULL_DATA.

DATI_COA_WITH_MQRO_FULL_DATA
È uguale a MQRO_COA, tranne che tutti i dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio del report.

Non specificare più di uno tra MQRO_COA, MQRO_COA_WITH_DATA e MQRO_COA_WITH_FULL_DATA.

Opzioni di conferma consegna: specificare una delle opzioni elencate per richiedere un messaggio di report di conferma consegna.

COD MQRO
Questo tipo di report viene generato dal gestore code quando un'applicazione richiama il messaggio dalla coda di destinazione in modo da eliminare il messaggio dalla coda. I dati del messaggio originale non sono inclusi nel messaggio di report.

Se il messaggio viene richiamato come parte di un'unità di lavoro, il messaggio di report viene generato all'interno della stessa unità di lavoro, in modo che il report non sia disponibile fino a quando non viene eseguito il commit dell'unità di lavoro. Se viene eseguito il backout dell'unità di lavoro, il report non viene inviato.

Un report COD non viene sempre generato se un messaggio viene richiamato con l'opzione MQGMO_MARK_SKIP_BACKOUT. Se viene eseguito il backout dell'unità di lavoro principale ma viene

eseguito il commit dell'unità di lavoro secondaria, il messaggio viene rimosso dalla coda, ma non viene generato un report COD.

Non viene generato un prospetto COD se il campo *Format* nel descrittore del messaggio è MQFMT_DEAD_LETTER_HEADER. Ciò impedisce la creazione di un report COD se il messaggio non è distribuibile e viene inserito in una coda di messaggi non recapitabili.

MQRO_COD non è valido se la coda di destinazione è una coda XCF.

Non specificare più di uno tra MQRO_COD, MQRO_COD_WITH_DATA e MQRO_COD_WITH_FULL_DATA.

DATI MQRO_COD_WITH_
È uguale a MQRO_COD, ad eccezione del fatto che i primi 100 byte dei dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio del report. Se il messaggio originale contiene una o più strutture di intestazione MQ , vengono incluse nel messaggio di report, in aggiunta ai 100 byte dei dati dell'applicazione.

Se MQGMO_ACCEPT_TRUNCATED_MSG è specificato nella chiamata MQGET per il messaggio originale e il messaggio richiamato è troncato, la quantità di dati del messaggio dell'applicazione inseriti nel messaggio di report dipende dall'ambiente:

- Su z/OS, è il minimo di:
  - La lunghezza del messaggio originale
  - La lunghezza del buffer utilizzato per richiamare il messaggio
  - 100 byte.
- In altri ambienti, è il minimo di:
  - La lunghezza del messaggio originale
  - 100 byte.

MQRO_COD_WITH_DATA non è valido se la coda di destinazione è una coda XCF.

Non specificare più di uno tra MQRO_COD, MQRO_COD_WITH_DATA e MQRO_COD_WITH_FULL_DATA.

DAD_COD MQRO_WITH_FULL_DATA
È uguale a MQRO_COD, ad eccezione del fatto che tutti i dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio di report.

MQRO_COD_WITH_FULL_DATA non è valido se la coda di destinazione è una coda XCF.

Non specificare più di uno tra MQRO_COD, MQRO_COD_WITH_DATA e MQRO_COD_WITH_FULL_DATA.

Opzioni di notifica azione: specificare una o entrambe le opzioni elencate per richiedere che l'applicazione ricevente invii un messaggio di report azione positiva o azione negativa.

MQRO_PAN
Questo tipo di report viene generato dall'applicazione che richiama il messaggio e agisce su di esso. Indica che l'azione richiesta nel messaggio è stata eseguita correttamente. L'applicazione che genera il report determina se i dati devono essere inclusi nel report.

Oltre a trasmettere questa richiesta all'applicazione che richiama il messaggio, il gestore code non intraprende alcuna azione basata su questa opzione. L'applicazione di richiamo deve generare il report, se appropriato.

MQRO_NAN
Questo tipo di report viene generato dall'applicazione che richiama il messaggio e agisce su di esso. Indica che l'azione richiesta nel messaggio non è stata eseguita correttamente. L'applicazione che genera il report determina se i dati devono essere inclusi nel report. Ad esempio, si potrebbe voler includere alcuni dati che indicano il motivo per cui non è stato possibile eseguire la richiesta.

Oltre a trasmettere questa richiesta all'applicazione che richiama il messaggio, il gestore code non intraprende alcuna azione basata su questa opzione. L'applicazione di richiamo deve generare il report, se appropriato.

La domanda deve determinare quali condizioni corrispondono ad un'azione positiva e quali ad un'azione negativa. Tuttavia, se la richiesta è stata eseguita solo parzialmente, generare un report NAN piuttosto che un report PAN, se richiesto. Ogni possibile condizione deve corrispondere a un'azione positiva o a un'azione negativa, ma non a entrambe.

Opzioni identificativo messaggio: specificare una delle opzioni elencate per controllare il modo in cui deve essere impostato il *MsgId* del messaggio di report (o del messaggio di risposta).

ID_MSG_NEW_MQRO
Questa è l'azione predefinita e indica che se un report o una risposta viene generato come risultato di questo messaggio, viene generato un nuovo *MsgId* per il report o il messaggio di risposta.

MQRO_PASS_MSG_ID
Se viene generato un report o una risposta come risultato di questo messaggio, il *MsgId* di questo messaggio viene copiato nel *MsgId* del report o del messaggio di risposta.

Il *MsgId* di un messaggio di pubblicazione sarà diverso per ogni sottoscrittore che riceve una copia della pubblicazione e quindi il *MsgId* copiato nel report o nel messaggio di risposta sarà diverso per ogni sottoscrittore.

Se questa opzione non è specificata, viene utilizzato MQRO_NEW_MSG_ID.

Opzioni identificativo di correlazione: specificare una delle opzioni elencate per controllare il modo in cui deve essere impostato il *CorrelId* del messaggio di report (o del messaggio di risposta).

ID_COPY_MQRO_MSG_TO_CORREL_ID
Questa è l'azione predefinita e indica che se viene generato un report o una risposta come risultato di questo messaggio, il *MsgId* di questo messaggio viene copiato nel *CorrelId* del report o del messaggio di risposta.

Il *MsgId* di un messaggio di pubblicazione sarà diverso per ogni sottoscrittore che riceve una copia della pubblicazione e quindi il *MsgId* copiato in *CorrelId* del messaggio di report o di risposta sarà diverso per ciascuno di essi.

ID CORREL_PASS_MQRO_
Se viene generato un report o una risposta come risultato di questo messaggio, il *CorrelId* di questo messaggio viene copiato nel *CorrelId* del report o del messaggio di risposta.

Il *CorrelId* di un messaggio di pubblicazione sarà specifico per un sottoscrittore a meno che non utilizzi l'opzione MQSO_SET_CORREL_ID e non imposti il campo ID SubCorrelin MQSD su MQCI_NONE. Pertanto, è possibile che il *CorrelId* copiato nel *CorrelId* del messaggio di report o di risposta sia diverso per ciascuno di essi.

Se questa opzione non viene specificata, viene utilizzato MQRO_COPY_MSG_ID_TO_CORREL_ID.

I server che rispondevano alle richieste o che generavano i messaggi di report devono controllare se le opzioni MQRO_PASS_MSG_ID o MQRO_PASS_CORREL_ID erano impostate nel messaggio originale. In tal caso, i server devono eseguire l'azione descritta per tali opzioni. Se nessuno dei due è impostato, i server devono eseguire l'azione predefinita corrispondente.

Opzioni di disposizione: specificare una delle opzioni elencate per controllare la disposizione del messaggio originale quando non può essere consegnato alla coda di destinazione. L'applicazione può impostare le opzioni di disposizione indipendentemente dalla richiesta di report di eccezioni.

MQRO_DEAD_LETTER_Q
Questa è l'azione predefinita e colloca il messaggio nella coda di messaggi non recapitabili se il messaggio non può essere consegnato alla coda di destinazione. Ciò si verifica nelle situazioni seguenti:

- Quando l'applicazione che ha inserito il messaggio originale non può essere notificata in modo sincrono del problema mediante il codice motivo restituito dalla chiamata MQPUT o MQPUT1 . Viene generato un messaggio di report di eccezione, se richiesto dal mittente.
- Quando l'applicazione che ha inserito il messaggio originale stava inserendo un argomento

MQRO_DISCARD_MSG
Questo elimina il messaggio se non può essere consegnato alla coda di destinazione. Ciò si verifica nelle situazioni seguenti:

- Quando l'applicazione che ha inserito il messaggio originale non può essere notificata in modo sincrono del problema mediante il codice motivo restituito dalla chiamata MQPUT o MQPUT1 . Viene generato un messaggio di report di eccezione, se richiesto dal mittente.
- Quando l'applicazione che ha inserito il messaggio originale stava inserendo un argomento

Se si desidera restituire il messaggio originale al mittente, senza che il messaggio originale venga inserito nella coda di messaggi non instradabili, il mittente deve specificare MQRO_DISCARD_MSG con MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_PASS_DISCARD_E_SCADENZA
Se questa opzione è impostata su un messaggio e viene generato un report o una risposta, il descrittore del messaggio del report eredita:

- MQRO_DISCARD_MSG se è stato impostato.
- Il tempo di scadenza rimanente del messaggio (se non si tratta di un report di scadenza). Se si tratta di un report di scadenza, il tempo di scadenza è impostato su 60 secondi.

Opzione Attività

ATTIVITÀ MQRO
L'utilizzo di questo valore consente la traccia dell'instradamento di **qualsiasi** messaggio in una rete del gestore code. L'opzione di report può essere specificata su qualsiasi messaggio utente corrente, consentendo all'utente di iniziare immediatamente a calcolare l'instradamento del messaggio attraverso la rete.

Se l'applicazione che genera il messaggio non può abilitare la generazione di report di attività, la creazione di report può essere abilitata utilizzando un'uscita incrociata API fornita dagli amministratori del gestore code.

Nota:

1. Minore è il numero di gestori code nella rete in grado di generare report di attività, minore è il dettaglio dell'instradamento.
2. I report di attività potrebbero essere difficili da collocare nell'ordine corretto per determinare il percorso intrapreso.
3. I report di attività potrebbero non essere in grado di trovare un instradamento alla destinazione richiesta.
4. I messaggi con questa serie di opzioni di report devono essere accettati da qualsiasi gestore code, anche se non comprendono l'opzione. Ciò consente all'opzione di report di essere impostata su qualsiasi messaggio utente, anche se elaborati da un gestore code nonIBM WebSphere MQ 6.0 o successivo.
5. Se un processo, un gestore code o un processo utente, esegue un'attività su un messaggio con questa opzione impostata, può scegliere di generare e inserire un report di attività.

Opzione predefinita: specificare quanto segue se non sono richieste opzioni di report:

MQRO_NONE
Utilizzare questo valore per indicare che non sono state specificate altre opzioni. MQRO_NONE è definito per aiutare la documentazione del programma. Non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Informazioni generali:

1. Tutti i tipi di report richiesti devono essere specificamente richiesti dall'applicazione che invia il messaggio originale. Ad esempio, se viene richiesto un report COA ma non un report di eccezione, viene generato un report COA quando il messaggio viene inserito nella coda di destinazione, ma non viene generato alcun report di eccezione se la coda di destinazione è piena quando arriva il messaggio.

Se non è impostata alcuna opzione *Report* , non viene generato alcun messaggio di report dal gestore code o dall'MCA (message channel agent).

Alcune opzioni di prospetto possono essere specificate anche se il gestore code locale non le riconosce; ciò è utile quando l'opzione deve essere elaborata dal gestore code *di destinazione* . Per ulteriori dettagli, vedere "Opzioni di report e indicatori di messaggi" a pagina 928.

Se viene richiesto un messaggio di report, il nome della coda a cui inviare il report deve essere specificato nel campo *ReplyToQ* . Quando viene ricevuto un messaggio di report, è possibile determinare la natura del report esaminando il campo *Feedback* nel descrittore del messaggio.

2. Se il gestore code o l'MCA che genera un messaggio di report non è in grado di inserire il messaggio di report nella coda di risposta (ad esempio, perché la coda di risposta o la coda di trasmissione è piena), il messaggio di report viene posizionato nella coda di messaggi non recapitabili. Se *anche* ha esito negativo o non vi è alcuna coda di messaggi non instradabili, l'azione intrapresa dipende dal tipo di messaggio di report:
   - Se il messaggio di report è un report di eccezione, il messaggio che ha generato il report di eccezione viene lasciato nella relativa coda di trasmissione; ciò garantisce che il messaggio non venga perso.
   - Per tutti gli altri tipi di report, il messaggio di report viene eliminato e l'elaborazione continua normalmente. Questa operazione viene eseguita perché il messaggio originale è già stato consegnato in modo sicuro (per i messaggi di report COA o COD) o non è più di alcun interesse (per un messaggio di report di scadenza).

   Una volta che un messaggio di report è stato posizionato correttamente su una coda (la coda di destinazione o una coda di trasmissione intermedia), il messaggio non è più soggetto a un'elaborazione speciale; viene trattato come qualsiasi altro messaggio.
3. Quando il report viene generato, la coda *ReplyToQ* viene aperta e il messaggio di report viene inserito utilizzando l'autorizzazione di *UserIdentifier* nell'MQMD del messaggio che causa il report, ad eccezione dei seguenti casi:
   - I report di eccezioni generati da un MCA ricevente vengono inseriti con qualsiasi autorizzazione utilizzata da MCA quando tenta di inserire il messaggio che causa il report.
   - I report COA generati dal gestore code vengono inseriti con qualsiasi autorizzazione utilizzata quando il messaggio che causa il report è stato inserito sul gestore code che genera il report. Ad esempio, se il messaggio è stato inserito da un MCA di ricezione utilizzando l'identificativo utente di MCA, il gestore code inserisce il report COA utilizzando l'identificativo utente di MCA.

   Le applicazioni che generano i report devono utilizzare la stessa autorizzazione utilizzata per generare una risposta; questa è di solito l'autorizzazione dell'identificativo utente nel messaggio originale.

   Se il report deve viaggiare verso una destinazione remota, i mittenti e i destinatari possono decidere se accettarlo o meno, allo stesso modo in cui lo fanno per altri messaggi.
4. Se è richiesto un messaggio di report con dati:
   - Il messaggio di report viene sempre generato con la quantità di dati richiesti dal mittente del messaggio originale. Se il messaggio di report è troppo grande per la coda di risposta, si verifica l'elaborazione sopra descritta; il messaggio di report non viene mai troncato per adattarsi alla coda di risposta.
   - Se *Format* del messaggio originale è MQFMT_XMIT_Q_HEADER, i dati inclusi nel report non includono MQXQH. I dati del report iniziano con il primo byte dei dati oltre MQXQH nel messaggio originale. Ciò si verifica indipendentemente dal fatto che la coda sia o meno una coda di trasmissione.
5. Se un messaggio di report COA, COD o di scadenza viene ricevuto nella coda di risposta, è garantito che il messaggio originale è arrivato, è stato consegnato o è scaduto, come appropriato. Tuttavia, se uno o più di questi messaggi di report viene richiesto e non viene ricevuto, non è possibile presumere l'inverso, poiché potrebbe essersi verificato uno dei seguenti casi:
   a. Il messaggio di report viene trattenuto perché un collegamento è inattivo.

b. Il messaggio di report viene congelato perché esiste una condizione di blocco in una coda di trasmissione intermedia o nella coda di risposta (ad esempio, la coda è piena o inibita per le inserzioni).

c. Il messaggio di report si trova su una coda di messaggi non recapitabili.

d. Quando il gestore code ha tentato di creare il messaggio di report, non è stato possibile inserirlo nella coda appropriata né nella coda di messaggi non recapitabili, quindi non è stato possibile generare il messaggio di report.

e. Si è verificato un errore del gestore code tra l'azione riportata (arrivo, consegna o scadenza) e la generazione del messaggio di report corrispondente. Ciò non si verifica per i messaggi di report COD se l'applicazione richiama il messaggio originale all'interno di un'unità di lavoro, poiché il messaggio di report COD viene generato all'interno della stessa unità di lavoro.

I messaggi di report di eccezione possono essere trattenuti nello stesso modo per i motivi 1, 2 e 3 precedenti. Tuttavia, quando un MCA non può generare un messaggio di report di eccezione (il messaggio di report non può essere inserito nella coda di risposta o nella coda di messaggi non recapitabili), il messaggio originale rimane nella coda di trasmissione al mittente e il canale viene chiuso. Ciò si verifica indipendentemente dal fatto che il messaggio di report debba essere generato all'estremità di invio o di ricezione del canale.

6. Se il messaggio originale è temporaneamente bloccato (con conseguente generazione di un messaggio di report di eccezione e inserimento del messaggio originale in una coda di messaggi non recapitabili), ma il blocco viene cancellato e un'applicazione legge il messaggio originale dalla coda di messaggi non recapitabili e lo inserisce nuovamente nella sua destinazione, potrebbe verificarsi quanto segue:
   - Anche se è stato generato un messaggio di report di eccezioni, il messaggio originale alla fine arriva correttamente alla sua destinazione.
   - Viene generato più di un messaggio di report di eccezione rispetto a un singolo messaggio originale, poiché il messaggio originale potrebbe incontrare un altro blocco in un secondo momento.

Messaggi di report durante l'inserimento in un argomento:

1. I report possono essere generati durante l'inserimento di un messaggio in un argomento. Questo messaggio verrà inviato a tutti i sottoscrittori dell'argomento, che potrebbe essere zero, uno o molti. È necessario tenerne conto quando si sceglie di utilizzare le opzioni di report, poiché di conseguenza potrebbero essere generati molti messaggi di report.
2. Quando si inserisce un messaggio in un argomento, è possibile che vi siano molte code di destinazione a cui deve essere fornita una copia del messaggio. Se alcune di queste code di destinazione presentano un problema, ad esempio la coda piena, il corretto completamento di MQPUT dipende dall'impostazione di NPMSGDLV o PMSGDLV (a seconda della persistenza del messaggio). Se l'impostazione è tale che la consegna del messaggio alla coda di destinazione deve essere eseguita correttamente (ad esempio, è un messaggio persistente per un sottoscrittore durevole e PMSGDLV è impostato su ALL o ALLDUR), l'esito positivo viene definito come uno dei seguenti criteri:
   - Inserimento riuscito nella coda del sottoscrittore
   - Utilizzo di MQRO_DEAD_LETTER_Q e inserimento riuscito nella coda di messaggi non instradabili se la coda del sottoscrittore non può accettare il messaggio
   - Utilizzare MQRO_DISCARD_MSG se la coda del sottoscrittore non può accettare il messaggio.

Messaggi di report per segmenti di messaggi:

1. I messaggi di prospetto possono essere richiesti per i messaggi per cui è consentita la segmentazione (consultare la descrizione dell'indicatore MQMF_SEGMENTATION_ALLOWED). Se il gestore code ritiene necessario segmentare il messaggio, è possibile generare un messaggio di report per ciascuno dei segmenti che successivamente rilevano la condizione pertinente. Le applicazioni devono essere preparate a ricevere più messaggi di report per ogni tipo di messaggio di report richiesto. Utilizzare il campo *GroupId* nel messaggio di report per correlare più report con l'identificativo del gruppo del messaggio originale e il campo *Feedback* identifica il tipo di ciascun messaggio di report.
2. Se MQGMO_LOGICAL_ORDER viene utilizzato per richiamare i messaggi di report per i segmenti, tenere presente che i report di *diversi tipi* potrebbero essere restituiti dalle successive chiamate

MQGET. Ad esempio, se vengono richiesti entrambi i report COA e COD per un messaggio segmentato dal gestore code, le chiamate MQGET per i messaggi di report potrebbero restituire i messaggi di report COA e COD intercalati in modo imprevedibile. Evitare questa situazione utilizzando l'opzione MQGMO_COMPLETE_MSG (facoltativamente con MQGMO_ACCEPT_TRUNCATED_MSG). MQGMO_COMPLETE_MSG fa sì che il gestore code riassembli i messaggi di report che hanno lo stesso tipo di report. Ad esempio, la prima chiamata MQGET potrebbe riassemblare tutti i messaggi COA relativi al messaggio originale e la seconda chiamata MQGET potrebbe riassemblare tutti i messaggi COD. Il primo riassemblato dipende dal tipo di messaggio di report che si verifica per primo sulla coda.

3. Le applicazioni che inseriscono i segmenti possono specificare diverse opzioni di report per ciascun segmento. Tuttavia, notare i seguenti punti:
   - Se i segmenti vengono richiamati utilizzando l'opzione MQGMO_COMPLETE_MSG, solo le opzioni del report nel *primo* segmento vengono rispettate dal gestore code.
   - Se i segmenti vengono richiamati uno per uno e la maggior parte di essi dispone di una delle opzioni MQRO_COD_ *, ma almeno un segmento non lo fa, non è possibile utilizzare l'opzione MQGMO_COMPLETE_MSG per richiamare i messaggi di report con una singola chiamata MQGET oppure utilizzare l'opzione MQGMO_ALL_SEGMENTS_AVAILABLE per rilevare quando sono arrivati tutti i messaggi di report.
4. In una rete MQ , i gestori code possono avere diverse funzionalità. Se un messaggio di report per un segmento viene generato da un gestore code o da un MCA che non supporta la segmentazione, per impostazione predefinita il gestore code o l'MCA non includono le necessarie informazioni sul segmento nel messaggio di report e ciò potrebbe rendere difficile identificare il messaggio originale che ha causato la generazione del report. Evitare questa difficoltà richiedendo i dati con il messaggio di report, ovvero specificando le opzioni MQRO_ * _WITH_DATA o MQRO_ * _WITH_FULL_DATA appropriate. Tuttavia, tenere presente che, se viene specificato MQRO_ * _WITH_DATA, *meno di* 100 byte di dati del messaggio dell'applicazione potrebbero essere restituiti all'applicazione che richiama il messaggio di report, se il messaggio di report viene generato da un gestore code o da un MCA che non supporta la segmentazione.

Contenuto del descrittore del messaggio per un messaggio di report: quando il gestore code o MCA (message channel agent) genera un messaggio di report, imposta i campi nel descrittore del messaggio sui seguenti valori e inserisce il messaggio nel modo normale.

Tabella 501. I valori utilizzati per i campi MQMD quando un messaggio di report viene generato dal sistema

Campo in MQMD	Valore utilizzato
StrucId	ID_STRUC_MQMD
Version	MQMD_VERSION_2
Report	MQRO_NONE
MsgType	REPORT MQMT
Expiry	MQEI_UNLIMITED
Feedback	Come appropriato per la natura del report (MQFB_COA, MQFB_COD, MQFB_EXPIRATION o un valore MQRC_ *)
Encoding	Copiato dal descrittore del messaggio originale
CodedCharSetId	Copiato dal descrittore del messaggio originale
Format	Copiato dal descrittore del messaggio originale
Priority	Copiato dal descrittore del messaggio originale
Persistence	Copiato dal descrittore del messaggio originale
MsgId	Come specificato dalle opzioni del prospetto nel descrittore del messaggio originale

Tabella 501. I valori utilizzati per i campi MQMD quando un messaggio di report viene generato dal sistema (Continua)

Campo in MQMD	Valore utilizzato
CorrelId	Come specificato dalle opzioni del prospetto nel descrittore del messaggio originale
BackoutCount	0
ReplyToQ	Spazi
ReplyToQMgr	Nome del gestore code
UserIdentifier	Come impostato dall'opzione MQPMO_PASS_IDENTITY_CONTEXT
AccountingToken	Come impostato dall'opzione MQPMO_PASS_IDENTITY_CONTEXT
ApplIdentityData	Come impostato dall'opzione MQPMO_PASS_IDENTITY_CONTEXT
PutApplType	MQAT_QMGR o come appropriato per l'agent del canale dei messaggi
PutApplName	Primi 28 byte del nome del gestore code o del nome dell'agent del canale dei messaggi. Per i messaggi di report generati dal bridge IMS , questo campo contiene i nomi del gruppo XCF e del membro XCF del sistema IMS a cui è correlato il messaggio.
PutDate	Data di invio del messaggio di report
PutTime	Ora di invio del messaggio di report
ApplOriginData	Spazi
GroupId	Copiato dal descrittore del messaggio originale
MsgSeqNumber	Copiato dal descrittore del messaggio originale
Offset	Copiato dal descrittore del messaggio originale
MsgFlags	Copiato dal descrittore del messaggio originale
OriginalLength	Copiato dal descrittore del messaggio originale se non MQOL_UNDEFINED e impostato sulla lunghezza dei dati del messaggio originale altrimenti

Un'applicazione che genera un report è consigliata per impostare valori simili, ad eccezione dei seguenti:

- Il campo *ReplyToQMgr* può essere impostato su spazi vuoti (il gestore code lo modifica con il nome del gestore code locale quando il messaggio viene inserito).
- Impostare i campi di contesto utilizzando l'opzione che sarebbe stata utilizzata per una risposta, normalmente MQPMO_PASS_IDENTITY_CONTEXT.

Analisi del campo del report: il campo *Report* contiene campi secondari; per questo motivo, le applicazioni devono verificare se il mittente del messaggio ha richiesto un determinato report deve utilizzare una delle tecniche descritte in "Analisi del campo del report" a pagina 930.

Questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è MQRO_NONE.

MsgType (MQLONG) per MQMD

Indica il tipo di messaggio. I tipi di messaggio sono raggruppati come segue:

MQMT_SYSTEM_FIRST
Il valore più basso per i tipi di messaggio definiti dal sistema.

SYSTEM_MQMT_LAST
Il valore più alto per i tipi di messaggi definiti dal sistema.

I seguenti valori sono attualmente definiti nell'intervallo di sistema:

MQMT_DATAGRAM
Il messaggio non richiede una risposta.

MQMT_REQUEST
Il messaggio richiede una risposta.

Specificare il nome della coda a cui inviare la risposta nel campo *ReplyToQ* . Il campo *Report* indica come impostare *MsgId* e *CorrelId* della risposta.

MQMT_REPLY
Il messaggio è la risposta a un messaggio di richiesta precedente (MQMT_REQUEST). Il messaggio deve essere inviato alla coda indicata dal campo *ReplyToQ* del messaggio di richiesta. Utilizzare il campo *Report* della richiesta per controllare come impostare *MsgId* e *CorrelId* della risposta.

Nota: Il gestore code non applica la relazione richiesta - risposta; questa è una responsabilità dell'applicazione.

REPORT MQMT
Il messaggio riporta alcune ricorrenze previste o non previste, di solito correlate ad altri messaggi (ad esempio, è stato ricevuto un messaggio di richiesta che conteneva dati non validi). Inviare il messaggio alla coda indicata dal campo *ReplyToQ* del descrittore del messaggio originale. Impostare i campi *Feedback* per indicare la natura del prospetto. Utilizzare il campo *Report* del messaggio originale per controllare la modalità di impostazione di *MsgId* e *CorrelId* del messaggio di report.

I messaggi di report generati dal gestore code o dall'agent del canale dei messaggi vengono sempre inviati alla coda *ReplyToQ* , con i campi *Feedback* e *CorrelId* impostati come descritto in precedenza.

Possono essere utilizzati anche valori definiti dall'applicazione. Devono essere compresi nel seguente intervallo:

MQMT_APPL_FIRST
Valore più basso per i tipi di messaggio definiti dall'applicazione.

APPL_MQMT_LAST
Il valore più alto per i tipi di messaggio definiti dall'applicazione.

Per le chiamate MQPUT e MQPUT1 , il valore *MsgType* deve essere compreso nell'intervallo definito dal sistema o nell'intervallo definito dall'applicazione; in caso contrario, la chiamata ha esito negativo con codice motivo MQRC_MSG_TYPE_ERROR.

Questo è un campo di output per la chiamata MQGET e un campo di immissione per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è MQMT_DATAGRAM.

Scadenza (MQLONG) per MQMD

Si tratta di un periodo di tempo espresso in decimi di secondo, impostato dall'applicazione che inserisce il messaggio. Il messaggio diventa idoneo per l'eliminazione se viene rimosso dalla coda di destinazione prima che trascorra questo periodo di tempo.

Ad esempio, per impostare un minuto per la scadenza, è necessario impostare **MQMD**.Da **Expiry** a 600.

Il valore viene ridotto per riflettere il tempo che il messaggio trascorre sulla coda di destinazione e anche su eventuali code di trasmissione intermedie se l'inserimento avviene su una coda remota. Può anche essere ridotto dagli agent del canale dei messaggi per riflettere i tempi di trasmissione, se sono significativi. Allo stesso modo, un'applicazione che inoltra questo messaggio a un'altra coda potrebbe diminuire il valore, se necessario, se ha conservato il messaggio per un periodo di tempo significativo. Tuttavia, il tempo di scadenza viene trattato come approssimativo e il valore non deve essere ridotto per riflettere piccoli intervalli di tempo.

Quando il messaggio viene richiamato da un'applicazione utilizzando la chiamata MQGET, il campo *Expiry* rappresenta il tempo di scadenza che rimane ancora.

Una volta trascorso il tempo di scadenza di un messaggio, diventa idoneo per essere eliminato dal gestore code. Il messaggio viene eliminato quando si verifica una chiamata MQGET browse o non browse che avrebbe restituito il messaggio se non fosse già scaduto. Ad esempio, una chiamata MQGET non - browse con il campo *MatchOptions* in MQGMO impostato su MQMO_NONE che legge da una coda ordinata FIFO elimina tutti i messaggi scaduti fino al primo messaggio non scaduto. Con una coda ordinata con priorità, la stessa chiamata eliminerà i messaggi scaduti di priorità superiore e i messaggi di priorità uguale che sono arrivati sulla coda prima del primo messaggio non scaduto.

Un messaggio scaduto non viene mai restituito a un'applicazione (da una chiamata MQGET di ricerca o non di ricerca), quindi il valore nel campo *Expiry* del descrittore del messaggio dopo una chiamata MQGET riuscita è maggiore di zero o il valore speciale MQEI_UNLIMITED.

Se un messaggio viene inserito su una coda remota, il messaggio potrebbe scadere (ed essere eliminato) mentre si trova su una coda di trasmissione intermedia, prima che il messaggio raggiunga la coda di destinazione.

Un report viene generato quando un messaggio scaduto viene eliminato, se il messaggio ha specificato una delle opzioni di report MQRO_EXPIRATION_ *. Se nessuna di queste opzioni viene specificata, non viene generato alcun report di questo tipo; si presume che il messaggio non sia più rilevante dopo questo periodo di tempo (forse perché un messaggio successivo lo ha sostituito).

Per un messaggio inserito nel punto di sincronizzazione, l'intervallo di scadenza inizia nel momento in cui il messaggio viene inserito, non nel momento in cui viene eseguito il commit del punto di sincronizzazione. È possibile che l'intervallo di scadenza passi prima che venga eseguito il commit del punto di sincronizzazione. In tal caso, il messaggio verrà eliminato in un momento successivo all'operazione di commit e non verrà restituito a un'applicazione in risposta a un'operazione MQGET.

Qualsiasi altro programma che elimina i messaggi in base all'ora di scadenza deve anche inviare un messaggio di report appropriato, se ne è stato richiesto uno.

Note:

1. Se un messaggio viene inserito con un'ora *Expiry* zero o un numero maggiore di 999 999 999, la chiamata MQPUT o MQPUT1 non riesce con codice motivo MQRC_EXPIRY_ERROR; in questo caso non viene generato alcun messaggio di report.

   Per abilitare il codice motivo 2013, MQRC_EXPIRY_ERROR, è necessario abilitare la variabile di ambiente AMQ_ENFORCE_MAX_EXPIRY_ERROR.

   Quanto segue utilizza un esempio per Linux:

   ```
   $ export AMQ_ENFORCE_MAX_EXPIRY_ERROR=True
   ```

   Notare che:

   - La cosa importante è esportare la variabile
   - Il valore effettivo viene ignorato, tuttavia, l'uso di `True` potrebbe essere utile quando si esamina la configurazione.
2. Poiché un messaggio con un'ora di scadenza trascorsa potrebbe non essere eliminato fino a un momento successivo, potrebbero essere presenti messaggi su una coda che hanno superato l'ora di scadenza e che non sono quindi idonei per il richiamo. Tuttavia, questi messaggi vengono conteggiati per il numero di messaggi sulla coda per tutti gli scopi, incluso il trigger di profondità.

   Se un sottoscrittore / consumatore (client) tenta di ottenere un messaggio e tale messaggio è scaduto, il client non riceve nulla poiché il messaggio è stato eliminato poiché era troppo vecchio. Inoltre, il client non riceverà alcun messaggio di errore.
3. Un report di scadenza viene generato, se richiesto, quando il messaggio viene eliminato, non quando diventa idoneo per l'eliminazione.
4. L'eliminazione di un messaggio scaduto e la generazione di un report di scadenza, se richiesto, non fanno mai parte dell'unità di lavoro dell'applicazione, anche se il messaggio era pianificato per l'eliminazione come risultato di una chiamata MQGET che opera all'interno di un'unità di lavoro.

5. Se un messaggio quasi scaduto viene richiamato da una chiamata MQGET all'interno di un'unità di lavoro e l'unità di lavoro viene successivamente ripristinata, il messaggio potrebbe diventare idoneo per essere eliminato prima di poter essere richiamato nuovamente.
6. Se un messaggio quasi scaduto è bloccato da una chiamata MQGET con MQGMO_LOCK, il messaggio potrebbe diventare idoneo per essere eliminato prima di poter essere richiamato da una chiamata MQGET con MQGMO_MSG_UNDER_CURSOR; il codice motivo MQRC_NO_MSG_UNDER_CURSOR viene restituito su questa chiamata MQGET successiva, se ciò si verifica.
7. Quando viene richiamato un messaggio di richiesta con una scadenza maggiore di zero, l'applicazione può intraprendere una delle seguenti azioni quando invia il messaggio di risposta:
   - Copiare il tempo di scadenza rimanente dal messaggio di richiesta al messaggio di risposta.
   - Impostare la scadenza nel messaggio di risposta su un valore esplicito maggiore di zero.
   - Impostare l'ora di scadenza nel messaggio di risposta su MQEI_UNLIMITED.

   L'azione da intraprendere dipende dalla progettazione dell'applicazione. Tuttavia, l'azione predefinita per l'inserimento di messaggi in una coda di messaggi non recapitabili (messaggi non recapitati) deve essere quella di conservare il tempo di scadenza rimanente del messaggio e di continuare a diminuirlo.
8. I messaggi trigger vengono sempre generati con MQEI_UNLIMITED.
9. Un messaggio (di solito su una coda di trasmissione) che ha un nome *Format* MQFMT_XMIT_Q_HEADER ha un secondo descrittore di messaggi all'interno di MQXQH. Pertanto, dispone di due campi *Expiry* associati. In questo caso vanno evidenziati i seguenti punti aggiuntivi:
   - Quando un'applicazione inserisce un messaggio su una coda remota, il gestore code inserisce il messaggio inizialmente su una coda di trasmissione locale e prefissa i dati del messaggio dell'applicazione con una struttura MQXQH. Il gestore code imposta i valori dei due campi *Expiry* in modo che siano uguali a quelli specificati dall'applicazione.

     Se un'applicazione inserisce un messaggio direttamente in una coda di trasmissione locale, i dati del messaggio devono già iniziare con una struttura MQXQH e il nome del formato deve essere MQFMT_XMIT_Q_HEADER. In questo caso, l'applicazione non deve impostare i valori di questi due campi *Expiry* in modo che siano uguali. (Il gestore code verifica che il campo *Expiry* all'interno di MQXQH contenga un valore valido e che i dati del messaggio siano sufficientemente lunghi da includerlo). Per un'applicazione che può scrivere direttamente nella coda di trasmissione, l'applicazione deve creare un'intestazione della coda di trasmissione con il descrittore del messaggio incorporato. Tuttavia, se il valore di scadenza nel descrittore del messaggio scritto nella coda di trasmissione non è coerente con il valore nel descrittore del messaggio incorporato, si verifica un errore di scadenza.
   - Quando un messaggio con un *Format* nome MQFMT_XMIT_Q_HEADER viene richiamato da una coda (normale o di trasmissione), il gestore code decresce *entrambi* questi campi *Expiry* con il tempo trascorso in attesa sulla coda. Non viene generato alcun errore se i dati del messaggio non sono abbastanza lunghi da includere il campo *Expiry* in MQXQH.
   - Il gestore code utilizza il campo *Expiry* nel descrittore del messaggio separato (ossia, non quello nel descrittore del messaggio integrato nella struttura MQXQH) per verificare se il messaggio è idoneo per l'eliminazione.
   - Se i valori iniziali dei due campi *Expiry* sono diversi, il tempo *Expiry* nel descrittore del messaggio separato quando il messaggio viene richiamato potrebbe essere maggiore di zero (quindi il messaggio non è idoneo per l'eliminazione), mentre è trascorso il tempo in base al campo *Expiry* in MQXQH. In questo caso, il campo *Expiry* in MQXQH è impostato su zero.
10. Il tempo di scadenza su un messaggio di risposta restituito dal bridge IMS è illimitato, a meno che MQIIH_PASS_EXPIRATION non sia impostato nel campo Indicatori di MQIIH. Per ulteriori informazioni, consultare Indicatori .

Viene riconosciuto il seguente valore speciale:

MQEI_UNLIMITED
Il messaggio ha una scadenza illimitata.

Questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è MQEI_UNLIMITED.

Expired messages on z/OS
On IBM MQ for z/OS, messages that have expired are discarded by the next appropriate MQGET call.

However, if no such call occurs, the expired message is not discarded, and, for some queues, a large number of expired messages can accumulate. To remedy this, set the queue manager to scan queues periodically and discard expired messages on one or more queues in one of the following ways:

Periodic scan
You can specify a period using the EXPRYINT (expiry interval) queue manager attribute. Each time the expiry interval is reached, the queue manager looks for candidate queues that are worth scanning to discard expired messages.

The queue manager maintains information about the expired messages on each queue, and knows whether a scan for expired messages is worthwhile. So, only a selection of queues is scanned at any time.

Shared queues are scanned by only one queue manager in a queue sharing group. Generally, it is the first queue manager to restart, or the first to have EXPRYINT set. If this queue manager terminates, another queue manager in the queue sharing group takes over the queue scanning. Set the expiry interval value for all queue managers within a queue sharing group to the same value.

Note that expiry processing takes place for every queue when a queue manager restarts, regardless of the EXPRYINT setting.

Explicit request
Issue the REFRESH QMGR TYPE(EXPIRY) command, specifying the queue or queues that you want scanned.

Applicazione di tempi di scadenza più bassi
Gli amministratori possono limitare la scadenza di qualsiasi messaggio inserito in una coda o in un argomento utilizzando l'attributo **CAPEXPRY** .

Ci sono due modi per impostare CAPEXPRY:

- V 9.4.0 V 9.4.0 Specifica dell'attributo **CAPEXPRY**
- Specifica dell'attributo **CAPEXPRY** nell'attributo **CUSTOM**

V 9.4.0 V 9.4.0 Da IBM MQ 9.4.0, è necessario utilizzare l'attributo **CAPEXPRY** .

Le seguenti informazioni si applicano indipendentemente da come è stato impostato l'attributo **CAPEXPRY** :

- Il valore di CAPEXPRY è espresso in decimi di secondi, quindi il valore 600 rappresenta un minuto.
- Una data / ora di scadenza specificata nel campo **Expiry** della struttura MQMD, da parte di un'applicazione, che è maggiore del valore **CAPEXPRY** specificato nella coda o nell'argomento viene sostituito da tale valore **CAPEXPRY** . Un tempo di scadenza specificato da un'applicazione, che è inferiore al valore **CAPEXPRY** utilizzato.
- Se nel percorso di risoluzione viene utilizzato più di un oggetto, ad esempio, quando un messaggio viene inserito in un alias o in una coda remota, il valore più basso di tutti i **CAPEXPRY** viene utilizzato come limite superiore per la scadenza del messaggio.
- Le modifiche ai valori **CAPEXPRY** diventano immediatamente effettive. Il valore di scadenza viene valutato per ogni inserimento in una coda o in un argomento e quindi è sensibile alla risoluzione dell'oggetto, che potrebbe differire tra ogni operazione di inserimento.

  Tuttavia, i messaggi esistenti nella coda, prima di una modifica in **CAPEXPRY**, non sono influenzati dalla modifica (ovvero, la loro scadenza rimane la stessa). Solo i nuovi messaggi inseriti nella coda o nell'argomento dopo la modifica in **CAPEXPRY** hanno la nuova ora di scadenza.

- In un cluster in cui un'operazione di inserimento viene eseguita su una coda aperta con MQOO_BIND_NOT_FIXED, ai messaggi possono essere assegnati valori di scadenza differenti per ogni operazione, a seconda del valore **CAPEXPRY** impostato per la coda di trasmissione, utilizzato dal canale, che invia il messaggio al gestore code di destinazione selezionato.

Attenzione: CAPEXPRY non deve essere utilizzato su code che contengono IBM MQ messaggi generati internamente come qualsiasi SYSTEM.CLUSTER.* e SYSTEM.PROTECTION.POLICY.QUEUE.

Interazioni tra l'attributo CAPEXPRY e l'attributo CAPEXPRY nell'attributo CUSTOM

V 9.4.0 V 9.4.0

Quando si migra un oggetto da una versione precedente del prodotto, il valore **CAPEXPRY** è impostato sul valore predefinito NOLIMIT (per code) e ASPARENT (per argomenti). Pertanto, se impostato, l'attributo **CAPEXPRY** all'interno dell'attributo **CUSTOM** ha la precedenza.

Se una coda o un argomento dispone già dell'attributo CAPEXPRY impostato all'interno dell'attributo CUSTOM, è necessario rimuovere l'attributo CAPEXPRY dall'attributo CUSTOM prima che il nuovo attributo CAPEXPRY sia impostato su un valore non predefinito. È possibile eseguire questa operazione in un singolo comando, ad esempio:

```
ALTER QLOCAL(Q1) CAPEXPRY(1000) CUSTOM('')
```

L'attributo CAPEXPRY impostato direttamente su code o argomenti (non all'interno dell'attributo CUSTOM) è un attributo cluster. Tutte le istanze di una coda cluster o di un argomento cluster devono utilizzare lo stesso valore CAPEXPRY. È ancora possibile che una coda di trasmissione riduca il tempo di scadenza di un messaggio se CAPEXPRY è stato impostato sulla coda di trasmissione e il valore è inferiore all'attributo CAPEXPRY della coda cluster o dell'argomento.

Riferimenti correlati

code DEFINE

DEFINE, argomento

Feedback (MQLONG) per MQMD

Il campo Feedback viene utilizzato con un messaggio di tipo MQMT_REPORT per indicare il tipo di report ed è significativo solo con quel tipo di messaggio.

Il campo può contenere uno dei valori MQFB_ * o uno dei valori MQRC_ *. I codici di feedback sono raggruppati come segue:

MQFB_NONE
Nessun feedback fornito.

MQFB_SYSTEM_FIRST
Valore più basso per il feedback generato dal sistema.

MQFB_SYSTEM_LAST
Il valore più alto per il feedback generato dal sistema.

L'intervallo di codici di feedback generati dal sistema da MQFB_SYSTEM_FIRST a MQFB_SYSTEM_LAST include i codici di feedback generici elencati in questo argomento (MQFB_ *) e anche i codici di errore (MQRC_ *) che possono verificarsi quando non è possibile inserire il messaggio nella coda di destinazione.

MQFB_APPL_FIRST
Valore più basso per il feedback generato dall'applicazione.

LAST APPL_MQFB
Valore massimo per il feedback generato dall'applicazione.

Le applicazioni che generano i messaggi di report non devono utilizzare i codici di feedback nell'intervallo di sistema (diverso da MQFB_QUIT), a meno che non vogliano simulare i messaggi di report generati dal gestore code o dall'agent del canale dei messaggi.

Nelle chiamate MQPUT o MQPUT1 , il valore specificato deve essere MQFB_NONE o compreso nell'intervallo di sistema o nell'intervallo dell'applicazione. Questa opzione è selezionata indipendentemente dal valore di *MsgType*.

Codici di feedback generali

COA MQFB
Conferma di arrivo sulla coda di destinazione (vedere MQRO_COA).

COD MQFB
Conferma della consegna all'applicazione ricevente (vedere MQRO_COD).

SCADENZA_MQFB
Il messaggio è stato eliminato perché non è stato rimosso dalla coda di destinazione prima della scadenza.

PAN MQFB
Notifica azione positiva (vedere MQRO_PAN).

NAN_MQFB
Notifica azione negativa (vedere MQRO_NAN).

MQFB_QUIT
Termina applicazione.

Questo può essere utilizzato da un programma di pianificazione del workload per controllare il numero di istanze di un programma applicativo in esecuzione. L'invio di un messaggio MQMT_REPORT con questo codice di feedback a un'istanza del programma applicativo indica a tale istanza che deve interrompere l'elaborazione. Tuttavia, l'aderenza a questa convenzione è una questione per l'applicazione; non viene applicata dal gestore code.

Codici di feedback del canale

MQFB_CHANNEL_COMPLETED
Un canale è terminato normalmente.

MQFB_CHANNEL_NON RIUSCITO
Un canale è terminato in maniera anomala e passa allo stato STOPPED.

MQFB_CHANNEL_FAIL_RETRY
Un canale è terminato in maniera anomala e passa allo stato RETRY.

IMS-codici di feedback bridge

Questi codici vengono utilizzati quando viene ricevuto un codice di rilevamento IMS-OTMA non previsto. Il sense code o, quando il sense code è 0x1A il codice motivo associato a tale sense code, è indicato in *Feedback*.

1. Per i codici *Feedback* compresi nell'intervallo tra MQFB_IMS_FIRST (300) e MQFB_IMS_LAST (399), è stato ricevuto un codice di condizione diverso da 0x1A . Il *sense code* è fornito dall'espressione (*Feedback* - MQFB_IMS_FIRST+1)
2. Per i codici *Feedback* compresi nell'intervallo tra MQFB_IMS_NACK_1A_REASON_FIRST (600) e MQFB_IMS_NACK_1A_REASON_LAST (855), è stato ricevuto un sense code di 0x1A . Il *codice motivo* associato al codice di rilevamento viene fornito dall'espressione (*Feedback* - MQFB_IMS_NACK_1A_REASON_FIRST)

Il significato dei codici di condizione IMS-OTMA e i codici di errore corrispondenti sono descritti in *Open Transaction Manager Access Guide and Reference*.

I seguenti codici di feedback possono essere generati dal bridge IMS :

LENGTH_ZERO MQFB_DATA_
La lunghezza del segmento era zero nei dati dell'applicazione del messaggio.

MQFB_DATA_LENGTH_NEGATIVE
Una lunghezza del segmento era negativa nei dati dell'applicazione del messaggio.

LENGTH_TOO_BIG MQFB_DATA_BIG
Una lunghezza di segmento era troppo grande nei dati dell'applicazione del messaggio.

MQFB_BUFFER_OVERFLOW
Il valore di uno dei campi di lunghezza potrebbe causare l'overflow dei dati nel buffer dei messaggi.

MQFB_LENGTH_OFF_BY_ONE
Il valore di uno dei campi di lunghezza era 1 byte troppo breve.

ERRORE MQFB_IIH
Il campo *Format* in MQMD specifica MQFMT_IMS, ma il messaggio non inizia con una struttura MQIIH valida.

MQFB_NOT_AUTHORIZED_FOR_IMS
L'ID utente contenuto nel descrittore del messaggio MQMD o la parola d'ordine contenuta nel campo *Authenticator* nella struttura MQIIH, non ha superato la convalida eseguita dal bridge IMS . Di conseguenza, il messaggio non è stato trasmesso a IMS.

ERRORE MQFB_IMS
IMSha restituito un errore non previsto. Per ulteriori informazioni sull'errore, consultare il log degli errori IBM MQ sul sistema su cui si trova il bridge IMS .

MQFB_IMS_FIRST
Quando il codice di rilevamento IMS-OTMA non è 0x1A, i codici di feedback generati da IMSsono compresi tra MQFB_IMS_FIRST (300) e MQFB_IMS_LAST (399). Il codice di rilevamento IMS-OTMA è *Feedback* meno MQFB_IMS_ERROR.

MQFB_IMS_LAST
Valore massimo per il feedback generato da IMSquando il codice di rilevamento non è 0x1A.

MQFB_IMS_NACK_1A_REASON_FIRST
Quando il codice di rilevamento è 0x1A, i codici di ritorno generati da IMSsono compresi nell'intervallo tra MQFB_IMS_NACK_1A_REASON_FIRST (600) e MQFB_IMS_NACK_1A_REASON_LAST (855).

MQFB_IMS_NACK_1A_REASON_LAST
Valore massimo per il feedback generato da IMSquando il codice di rilevamento è 0x1A

CICS-bridge feedback codes: i seguenti codici di feedback possono essere generati da CICS bridge:

MQFB_CICS_APPL_ABENDED
Il programma applicativo specificato nel messaggio è terminato in modo anomalo. Questo codice di feedback si verifica solo nel campo *Reason* della struttura MQDLH.

MQFB_CICS_APPL_NOT_STARTED
EXEC CICS LINK per il programma applicativo specificato nel messaggio non è riuscito. Questo codice di feedback si verifica solo nel campo *Reason* della struttura MQDLH.

MQFB_CICS_BRIDGE_FAILURE
CICS bridge è terminato in modo anomalo senza completare la normale elaborazione degli errori.

ERRORE CCSID MQFB_CICS_
Identificativo della serie di caratteri non valido.

ERRORE MQFB_CICS_CIH
Struttura dell'intestazione delle informazioni CICS mancante o non valida.

ERRORE MQFB_CICS_COMMAREA_
Lunghezza di CICS COMMAREA non valida.

ERRORE MQFB_CICS_CORREL_ID_
Identificativo di correlazione non valido.

ERRORE MQFB_CICS_DLQ
L'attività CICS bridge non è stata in grado di copiare una risposta a questa richiesta nella coda di messaggi non instradabili. La richiesta è stata ripristinata.

ERRORE MQFB_CICS_ENCODING_ERROR
Codifica non valida.

ERRORE INTERNO MQFB_CICS_
CICS bridge ha rilevato un errore non previsto.

Questo codice di feedback si verifica solo nel campo *Reason* della struttura MQDLH.

MQFB_CICS_NOT_AUTHORIZED
Identificativo utente non autorizzato o parola d'ordine non valida.

Questo codice di feedback si verifica solo nel campo *Reason* della struttura MQDLH.

MQFB_CICS_UOW_BACKED_OUT
L'unità di lavoro è stata ripristinata per uno dei seguenti motivi:

- È stato rilevato un errore durante l'elaborazione di un'altra richiesta all'interno della stessa unità di lavoro.
- Si è verificata una fine anomala CICS mentre l'unità di lavoro era in corso.

ERRORE MQFB_CICS_UOW_
Campo di controllo dell'unità di lavoro *UOWControl* non valido.

Codici di feedback del messaggio di instradamento traccia:

MQFB_ATTIVITÀ
Utilizzato con il formato MQFMT_EMBDED_PCF per consentire l'opzione dei dati utente che seguono i report di attività.

MQFB_MAX_ATTIVITÀ
Restituito quando il messaggio di trace - route viene eliminato perché il numero di attività in cui il messaggio è stato coinvolto supera il limite massimo di attività.

MQFB_NOT_FORWARDED
Viene restituito quando il messaggio di instradamento della traccia viene eliminato perché sta per essere inviato a un gestore code remoto che non supporta i messaggi di instradamento della traccia.

MQFB_NON_CONSEGNATO
Restituito quando il messaggio di trace - route viene eliminato perché sta per essere inserito su una coda locale.

MQFB_UNSUPPORT_FORWARDING
Restituito quando il messaggio traceroute viene eliminato perché un valore nel parametro di inoltro non è riconosciuto e si trova nella maschera di bit rifiutata.

MQFB_UNSUPPORTED_DELIVERY
Restituito quando il messaggio di traceroute viene eliminato perché un valore nel parametro di consegna non è riconosciuto e si trova nella maschera di bit rifiutata.

IBM MQ codici di errore: per i messaggi di report di eccezioni, *Feedback* contiene un codice di errore IBM MQ . Tra i codici di errore possibili sono:

MQRC_PUT_INIBITO
(2051, X'803 ') Chiamate Put inibite per la coda.

MQRC_Q_FULL
(2053, X'805 ') La coda contiene già il numero massimo di messaggi.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808 ') Nessuno spazio disponibile sul disco per la coda.

MQRC_PERSIST_NOT_ALLOWED
(2048, X'800 ') La coda non supporta i messaggi persistenti.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') La lunghezza del messaggio è maggiore del massimo consentito per il gestore code.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') La lunghezza del messaggio è maggiore del massimo consentito per la coda.

Per un elenco completo dei codici di errore, vedere:

- Per IBM MQ for z/OS, vedi Codici di completamento API e motivo.

- Per tutte le altre piattaforme, vedi Codici di completamento e motivo API.

.

Questo è un campo di output per la chiamata MQGET e un campo di immissione per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è MQFB_NONE.

Codifica (MQLONG) per MQMD

Specifica la codifica numerica dei dati numerici nel messaggio; non si applica ai dati numerici nella stessa struttura MQMD. La codifica numerica definisce la rappresentazione utilizzata per numeri interi binari, interi decimali compressi e numeri a virgola mobile.

Su z/OS, la parte numero intero binario del campo `Encoding` viene utilizzata anche per specificare la codifica numero intero dei dati del carattere nel corpo del messaggio quando l'identificativo della serie di caratteri corrispondente indica che la rappresentazione della serie di caratteri dipende dalla codifica utilizzata per i numeri interi binari. Ciò influenza solo alcune serie di caratteri multibyte (ad esempio, le serie di caratteri UTF-16).

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Il gestore code non verifica la validità del campo. Viene definito il seguente valore speciale:

MQEN_NATIVE
La codifica è il valore predefinito per il linguaggio di programmazione e la macchina su cui è in esecuzione l'applicazione.

Nota: Il valore di questa costante dipende dal linguaggio di programmazione e dall'ambiente. Per questo motivo, le applicazioni devono essere compilate utilizzando i file di intestazione, macro, COPY o INCLUDE appropriati all'ambiente in cui verrà eseguita l'applicazione.

Le applicazioni che inseriscono i messaggi generalmente specificano MQENC_NATIVE. Le applicazioni che richiamano i messaggi devono confrontare questo campo con il valore MQENC_NATIVE; se i valori differiscono, l'applicazione potrebbe dover convertire i dati numerici nel messaggio. Utilizzare l'opzione MQGMO_CONVERT per richiedere al gestore code di convertire il messaggio come parte dell'elaborazione della chiamata MQGET. Consultare “Codifiche macchina” a pagina 925 per i dettagli su come viene creato il campo `Encoding` .

Se si specifica l'opzione MQGMO_CONVERT nella chiamata MQGET, questo campo è un campo di input/output. Il valore specificato dall'applicazione è la codifica in cui convertire i dati del messaggio, se necessario. Se la conversione ha esito positivo o non è necessario, il valore non viene modificato. Se la conversione ha esito negativo, il valore dopo la chiamata MQGET rappresenta la codifica del messaggio non convertito restituito all'applicazione.

In altri casi, si tratta di un campo di output per la chiamata MQGET e di un campo di input per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è MQENC_NATIVE.

CodedCharSetId (MQLONG) per MQMD

Questo campo specifica l'identificativo della serie di caratteri dei dati del carattere nel corpo del messaggio.

Nota: I dati dei caratteri in MQMD e nelle altre strutture di dati MQ che sono parametri sulle chiamate devono trovarsi nella serie di caratteri del gestore code. Questo è definito dall'attributo **CodedCharSetId** del gestore code; consultare “Attributi per il gestore code” a pagina 818 per i dettagli di questo attributo.

Se questo campo è impostato su MQCCSI_Q_MGR quando si richiama MQGET con MQGMO_CONVERT nelle opzioni, il comportamento è diverso tra le applicazioni client e server. Per le applicazioni server, la codepage utilizzata per la conversione dei caratteri è `CodedCharSetId` del gestore code; per le applicazioni client, la codepage utilizzata per la conversione dei caratteri è la codepage locale corrente.

Per le applicazioni client, MQCCSI_Q_MGR viene compilato in base alla locale del client e non a quella del gestore code. L'eccezione a questa regola è quando si inserisce un messaggio in una coda bridge IMS ; ciò che viene restituito, nel campo *`CodedCharSetId`* di MQMD, è il CCSID del gestore code.

Non utilizzare il seguente valore speciale:

APPL MQCCSI
Ciò determina un valore non corretto nel campo CodedCharSetId di MQMD e causa un codice di ritorno di MQRC_SOURCE_CCSID_ERROR (o MQRC_FORMAT_ERROR per z/OS) quando il messaggio viene ricevuto utilizzando la chiamata MQget con l'opzione MQGMO_CONVERT.

È possibile utilizzare i seguenti valori speciali:

MQCCSI_Q_MGR
I dati carattere nel messaggio si trovano nella serie di caratteri del gestore code.

Nelle chiamate MQPUT e MQPUT1 , il gestore code modifica questo valore nell'MQMD inviato con il messaggio nell'identificativo della serie di caratteri true del gestore code. Di conseguenza, il valore MQCCSI_Q_MGR non viene mai restituito dalla chiamata MQGET.

MQCCSI_DEULT
Il CodedCharSetId dei dati nel campo *String* viene definito dal campo CodedCharSetId nella struttura dell'intestazione che precede la struttura MQCFH o dal campo CodedCharSetId in MQMD se MQCFH si trova all'inizio del messaggio.

MQCCSI_INHERIT
I dati carattere nel messaggio si trovano nella stessa serie di caratteri di questa struttura; questa è la serie di caratteri del gestore code. (Solo per MQMD, MQCCSI_INHERIT ha lo stesso significato di MQCCSI_Q_MGR).

Il gestore code modifica questo valore in MQMD che viene inviato con il messaggio all'effettivo identificativo della serie di caratteri MQMD. Se non si verifica alcun errore, il valore MQCCSI_INHERIT non viene restituito dalla chiamata MQGET.

Non utilizzare MQCCSI_INHERIT se il valore del campo PutApplType in MQMD è MQAT_BROKER.

MQCCSI_EMBEDDED
I dati carattere nel messaggio si trovano in una serie di caratteri con l'identificativo contenuto nei dati messaggio stessi. È possibile che vi sia un numero qualsiasi di identificativi della serie di caratteri incorporati nei dati del messaggio, che si applicano a parti differenti dei dati. Questo valore deve essere utilizzato per i messaggi PCF (con un formato MQFMT_ADMIN, MQFMT_EVENT o MQFMT_PCF) che contengono dati in una combinazione di serie di caratteri. Per ciascuna struttura MQCFST, MQCFSL e MQCFSF contenuta nel messaggio PCF deve essere specificato un identificativo della serie di caratteri esplicito e non MQCCSI_DEFAULT.

Se un messaggio in formato MQFMT_EMBEDDED_PCF deve contenere dati in una combinazione di serie di caratteri, non utilizzare MQCCSI_EMBEDDED. Impostare invece MQEPH_CCSID_EMBEDDED nel campo Indicatori nella struttura MQEPH. Ciò equivale all'impostazione di MQCCSI_EMBEDDED nella struttura precedente. Ogni struttura MQCFST, MQCFSL e MQCFSF contenuta nel messaggio PCF deve avere un identificativo della serie di caratteri esplicito specificato e non MQCCSI_DEFAULT. Per ulteriori informazioni sulla struttura MQEPH, vedere “MQEPH - Intestazione PCF integrata” a pagina 371.

Specificare questo valore solo sulle chiamate MQPUT e MQPUT1 . Se viene specificato nella chiamata MQGET, impedisce la conversione del messaggio.

Nelle chiamate MQPUT e MQPUT1 , il gestore code modifica i valori MQCCSI_Q_MGR e MQCCSI_INHERIT nell'MQMD inviato con il messaggio come descritto in precedenza, ma non modifica l'MQMD specificato nella chiamata MQPUT o MQPUT1 . Nessun altro controllo viene eseguito sul valore specificato.

Le applicazioni che richiamano i messaggi devono confrontare questo campo con il valore previsto dall'applicazione; se i valori differiscono, l'applicazione potrebbe dover convertire i dati carattere nel messaggio.

Su z/OS, il campo Encoding di MQMD viene utilizzato per specificare la codifica intera dei dati carattere nel corpo del messaggio, quando il campo CodedCharSetId di MQMD indica che la rappresentazione della serie di caratteri dipende dalla codifica utilizzata per i numeri interi binari. Su Multipiattaforme, l'ordine di byte dei dati di caratteri si presume sia lo stesso della codifica di numeri interi nativa per la piattaforma su cui è in esecuzione il gestore code. Ciò influenza solo alcune serie di caratteri multibyte (ad esempio, le serie di caratteri UTF-16).

Se si specifica l'opzione MQGMO_CONVERT nella chiamata MQGET, questo campo è un campo di input/output. Il valore specificato dall'applicazione è il CCSID (coded character set identifier) in cui convertire i dati del messaggio, se necessario. Se la conversione ha esito positivo o non è necessaria, il valore non viene modificato (ad eccezione del fatto che il valore MQCCSI_Q_MGR o MQCCSI_INHERIT viene convertito nel valore effettivo). Se la conversione ha esito negativo, il valore dopo la chiamata MQGET rappresenta il CCSID (coded character set identifier) del messaggio non convertito restituito all'applicazione.

Altrimenti, questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è MQCCSI_Q_MGR.

Formato (MQCHAR8) per MQMD

È un nome che il mittente del messaggio utilizza per indicare al ricevente la natura dei dati nel messaggio. Tutti i caratteri che si trovano nella serie di caratteri del gestore code possono essere specificati per il nome, ma è necessario limitare il nome a quanto segue:

- Maiuscolo da A a Z
- Cifre numeriche da 0 a 9

Se vengono utilizzati altri caratteri, potrebbe non essere possibile tradurre il nome tra le serie di caratteri dei gestori code di invio e di ricezione.

Riempire il nome con spazi vuoti fino alla lunghezza del campo oppure utilizzare un carattere null per terminare il nome prima della fine del campo; i caratteri null e gli eventuali caratteri successivi vengono trattati come spazi vuoti. Non specificare un nome con spazi vuoti iniziali o intermedi. Per la chiamata MQGET, il gestore code restituisce il nome riempito con spazi vuoti alla lunghezza del campo.

Il gestore code non controlla che il nome sia conforme ai suggerimenti sopra descritti.

I nomi che iniziano con MQ in lettere maiuscole, minuscole e miste hanno un significato definito dal gestore code; non utilizzare nomi che iniziano con queste lettere per i propri formati. I formati integrati del gestore code sono:

MQFMT_NONE
La natura dei dati non è definita: i dati non possono essere convertiti quando il messaggio viene richiamato da una coda utilizzando l'opzione MQGMO_CONVERT.

Se si specifica MQGMO_CONVERT nella chiamata MQGET e la serie di caratteri o la codificazione dei dati nel messaggio differisce da quella specificata nel parametro **MsgDesc** , il messaggio viene restituito con i seguenti codici di completamento e motivo (presupponendo che non vi siano altri errori):

- Codice di completamento MQCC_WARNING e codice motivo MQRC_FORMAT_ERROR se i dati MQFMT_NONE si trova all'inizio del messaggio.
- Codice di completamento MQCC_OK e codice motivo MQRC_NONE se i dati MQFMT_NONE si trova alla fine del messaggio (ovvero, preceduti da una o più strutture di intestazione MQ). Le strutture dell'intestazione MQ vengono convertite nella serie di caratteri richiesta e codificate in questo caso.

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_NONE_ARRAY, che ha lo stesso valore di MQFMT_NONE, ma è un array di caratteri invece di una stringa.

MMQFMT_ADMIN
Il messaggio è una richiesta del server dei comandi o un messaggio di risposta in formato PCF (programmable command format). I messaggi di questo formato possono essere convertiti se l'opzione MQGMO_CONVERT viene specificata sulla chiamata MQGET. Consultare Utilizzo dei formati di comando programmabili per ulteriori informazioni sull'utilizzo dei messaggi del formato di comandi programmabili.

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_ADMIN_ARRAY, che ha lo stesso valore di MQFMT_ADMIN, ma è un array di caratteri anziché una stringa.

MQFMT_CICS
I dati del messaggio iniziano con l'intestazione delle informazioni CICS MQCIH, seguita dai dati dell'applicazione. Il nome del formato dei dati dell'applicazione è fornito dal campo `Format` nella struttura MQCIH.

z/OS Su z/OS, specificare l'opzione MQGMO_CONVERT nella chiamata MQGET per convertire i messaggi con formato MQFMT_CICS.

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_CICS_ARRAY; ha lo stesso valore di MQFMT_CICS, ma è un array di caratteri invece di una stringa.

MQFMT_COMMAND_1
Il messaggio è un messaggio di risposta del server di comandi MQSC contenente il conteggio oggetti, il codice di completamento e il codice motivo. I messaggi di questo formato possono essere convertiti se l'opzione MQGMO_CONVERT viene specificata sulla chiamata MQGET.

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_COMMAND_1_ARRAY , che ha lo stesso valore di MQFMT_COMMAND_1, ma è un array di caratteri anziché una stringa.

MQFMT_COMMAND_2
Il messaggio è un messaggio di risposta del server di comandi MQSC contenente informazioni sugli oggetti richiesti. I messaggi di questo formato possono essere convertiti se l'opzione MQGMO_CONVERT viene specificata sulla chiamata MQGET.

Per il linguaggio di programmazione C, è definita anche la costante MQFMT_COMMAND_2_ARRAY ; ha lo stesso valore di MQFMT_COMMAND_2, ma è un array di caratteri invece di una stringa.

MQFMT_DEAD_LETTER_HEADER
I dati del messaggio iniziano con l'intestazione non instradabile MQDLH. I dati del messaggio originale seguono immediatamente la struttura MQDLH. Il nome del formato dei dati del messaggio originale viene fornito dal campo `Format` nella struttura MQDLH; consultare "MQDLH - Intestazione lettera non instradabile" a pagina 359 per i dettagli di questa struttura. I messaggi di questo formato possono essere convertiti se l'opzione MQGMO_CONVERT viene specificata sulla chiamata MQGET.

I report COA e COD non vengono generati per i messaggi che hanno un `Format` di MQFMT_DEAD_LETTER_HEADER.

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_DEAD_LETTER_HEADER_ARRAY, che ha lo stesso valore di MQFMT_DEAD_LETTER_HEADER, ma è un array di caratteri anziché una stringa.

INTESTAZIONE_DIST_MQFM
I dati del messaggio iniziano con l'intestazione dell'elenco di distribuzione MQDH; ciò include gli array dei record MQOR e MQPMR. L'intestazione dell'elenco di distribuzione può essere seguita da ulteriori dati. Il formato dei dati aggiuntivi (se presenti) viene fornito dal campo *`Format`* nella struttura MQDH; consultare "MQDH - Intestazione di distribuzione" a pagina 352 per i dettagli di questa struttura. I messaggi con formato MQFMT_DIST_HEADER possono essere convertiti se l'opzione MQGMO_CONVERT viene specificata nella chiamata MQGET.

Questo formato è supportato nei seguenti ambienti:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Windows Windows

e per IBM MQ MQI clients collegati a questi sistemi.

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_DIST_HEADER_ARRAY, che ha lo stesso valore di MQFMT_DIST_HEADER, ma è un array di caratteri anziché una stringa.

MQFMT_EMBEDDED_PCF
Formato per un messaggio trace - route, purché il valore del comando PCF sia impostato su MQCMD_TRACE_ROUTE. L'uso di questo formato consente l'invio dei dati utente insieme al messaggio di traccia - instradamento, a condizione che le loro applicazioni possano far fronte ai parametri PCF precedenti.

L'intestazione PCF deve essere la prima intestazione oppure il messaggio non verrà considerato come un messaggio di instradamento traccia. Ciò significa che il messaggio non può essere in un gruppo e che i messaggi di trace - route non possono essere segmentati. Se un messaggio di trace - route viene inviato in un gruppo, il messaggio viene rifiutato con il codice motivo MQRC_MSG_NOT_ALLOWED_IN_GROUP.

Tenere presente che MQFMT_ADMIN può essere utilizzato anche per il formato di un messaggio traceroute, ma in tal caso non è possibile inviare dati utente insieme al messaggio traceroute.

EVENTO MQFMT
Il messaggio è un messaggio di eventi di MQ che riporta un evento che si è verificato. I messaggi di evento hanno la stessa struttura dei comandi programmabili; consultare PCF command messages per ulteriori informazioni su questa struttura e Event monitoring per informazioni sugli eventi.

I messaggi di evento Version-1 possono essere convertiti in tutti gli ambienti se l'opzione MQGMO_CONVERT viene specificata nella chiamata MQGET. I messaggi di eventi Version-2 possono essere convertiti solo su z/OS.

Per il linguaggio di programmazione C, viene definita anche la costanti MQFMT_EVENT_ARRAY, che ha lo stesso valore di MQFMT_EVENT, ma è un array di caratteri anziché una stringa.

IMS MQFMT
I dati del messaggio iniziano con l'intestazione delle informazioni IMS MQIIH, seguita dai dati applicazione. Il nome del formato dei dati dell'applicazione viene fornito dal campo `Format` nella struttura MQIIH.

Per dettagli su come viene gestita la struttura MQIIH quando si utilizza MQGET con MQGMO_CONVERT, consultare “Formato (MQCHAR8) per MQIIH” a pagina 420 e “ReplyTo(MQCHAR8) per MQIIH” a pagina 421.

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_IMS_ARRAY; ha lo stesso valore di MQFMT_IMS, ma è un array di caratteri invece di una stringa.

MQFMT_IMS_VAR_STRING
Il messaggio è una stringa variabile IMS , che è una stringa nel formato `ll`*`zz`*`ccc`, dove:

`ll`
è un campo di lunghezza di 2 byte che specifica la lunghezza totale dell'elemento stringa della variabile IMS . Questa lunghezza è uguale alla lunghezza di `ll` (2 byte), più la lunghezza di *`zz`* (2 byte), più la lunghezza della stringa di caratteri stessa. `ll` è un numero intero binario a 2 byte nella codifica specificata dal campo `Encoding` .

`zz`
è un campo a 2 byte contenente indicatori significativi per IMS. *`zz`* è una stringa di byte composta da due campi MQBYTE e viene trasmessa senza modifiche dal mittente al destinatario (ovvero, *`zz`* non è soggetto ad alcuna conversione).

`ccc`
è una stringa di caratteri a lunghezza variabile contenente `ll-4` caratteri. *`ccc`* si trova nella serie di caratteri specificata dal campo `CodedCharSetId` .

Su z/OS, i dati del messaggio possono essere costituiti da una sequenza di IMS stringhe di variabili unite, con ogni stringa nel formato `llzzccc`. Non devono essere ignorati byte tra stringhe di variabili IMS successive. Ciò significa che se la prima stringa ha una lunghezza dispari, la seconda stringa sarà

disallineata, cioè non inizierà su un limite che è un multiplo di due. Prestare attenzione quando si costruiscono tali stringhe su macchine che richiedono l'allineamento di tipi di dati elementari.

Utilizzare l'opzione MQGMO_CONVERT sulla chiamata MQGET per convertire i messaggi con formato MQFMT_IMS_VAR_STRING.

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_IMS_VAR_STRING_ARRAY, che ha lo stesso valore di MQFMT_IMS_VAR_STRING, ma è un array di caratteri anziché una stringa.

MQFMT_MD_EXTENSIONE
I dati del messaggio iniziano con l'estensione MQMDE del descrittore del messaggio e sono facoltativamente seguiti da altri dati (di solito i dati del messaggio dell'applicazione). Il nome formato, la serie di caratteri e la codifica dei dati che seguono MQMDE sono forniti dai campi `Format`, `CodedCharSetId` e `Encoding` in MQMDE. Consultare "MQMDE - Estensione descrittore messaggio" a pagina 484 per i dettagli di questa struttura. I messaggi di questo formato possono essere convertiti se l'opzione MQGMO_CONVERT viene specificata sulla chiamata MQGET.

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_MD_EXTENSION_ARRAY, che ha lo stesso valore di MQFMT_MD_EXTENSION, ma è un array di caratteri invece di una stringa.

MQFMT_PCF
Il messaggio è un messaggio definito dall'utente che è conforme alla struttura di un messaggio PCF (programmable command format). I messaggi di questo formato possono essere convertiti se l'opzione MQGMO_CONVERT viene specificata sulla chiamata MQGET. Consultare Utilizzo dei formati di comando programmabili per ulteriori informazioni sull'utilizzo dei messaggi del formato di comandi programmabili.

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_PCF_ARRAY, che ha lo stesso valore di MQFMT_PCF, ma è un array di caratteri anziché una stringa.

MQFMT_REF_MSG_HEADER
I dati del messaggio iniziano con l'intestazione del messaggio di riferimento MQRMH e sono facoltativamente seguiti da altri dati. Il nome formato, la serie di caratteri e la codifica dei dati sono forniti dai campi `Format`, `CodedCharSetId` e `Encoding` in MQRMH. Consultare "MQRMH - Intestazione messaggio di riferimento" a pagina 562 per i dettagli di questa struttura. I messaggi di questo formato possono essere convertiti se l'opzione MQGMO_CONVERT viene specificata sulla chiamata MQGET.

Questo formato è supportato nei seguenti ambienti:

- AIX
- IBM i
- Linux
- Windows

e per IBM MQ MQI clients collegati a questi sistemi.

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_REF_MSG_HEADER_ARRAY; ha lo stesso valore di MQFMT_REF_MSG_HEADER, ma è una schiera di caratteri invece di una stringa.

MQFMT_RF_HEADER
I dati del messaggio iniziano con le regole e l'intestazione di formattazione MQRFH ed è facoltativamente seguita da altri dati. Il nome del formato, la serie di caratteri e la codifica dei dati (se presenti) vengono forniti dai campi `Format`, `CodedCharSetId` e `Encoding` in MQRFH. I messaggi di questo formato possono essere convertiti se l'opzione MQGMO_CONVERT viene specificata sulla chiamata MQGET.

Per il linguaggi di programmazione C, viene definita anche la costante MQFMT_RF_HEADER_ARRAY, che ha lo stesso valore di MQFMT_RF_HEADER, ma è un array di caratteri anziché una stringa.

MQFMT_RF_HEADER_2
I dati del messaggio iniziano con le regole version-2 e l'intestazione di formattazione MQRFH2ed è facoltativamente seguita da altri dati. Il nome del formato, la serie di caratteri e la codifica dei dati facoltativi (se presenti) sono forniti dai campi `Format`, `CodedCharSetIde` `Encoding` in MQRFH2. I messaggi di questo formato possono essere convertiti se l'opzione MQGMO_CONVERT viene specificata sulla chiamata MQGET.

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_RF_HEADER_2_ARRAY ; ha lo stesso valore di MQFMT_RF_HEADER_2, ma è un array di caratteri anziché una stringa.

MQFMT_STRING
I dati del messaggio dell'applicazione possono essere una stringa SBCS (single - byte character set) o una stringa DBCS (double - byte character set). I messaggi di questo formato possono essere convertiti se l'opzione MQGMO_CONVERT viene specificata sulla chiamata MQGET.

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_STRING_ARRAY, che ha lo stesso valore di MQFMT_STRING, ma è un array di caratteri invece di una stringa.

MQFMT_TRIGGER
Il messaggio è un messaggio di trigger, descritto dalla struttura MQTM; consultare “MQTM - Messaggio trigger” a pagina 616 per dettagli su questa struttura. I messaggi di questo formato possono essere convertiti se l'opzione MQGMO_CONVERT viene specificata sulla chiamata MQGET.

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_TRIGGER_ARRAY, che ha lo stesso valore di MQFMT_TRIGGER, ma è un array di caratteri invece di una stringa.

Intestazione MQFMT_WORK_INFO_HEADER
I dati del messaggio iniziano con l'intestazione delle informazioni di lavoro MQWIH, seguita dai dati dell'applicazione. Il nome del formato dei dati dell'applicazione viene fornito dal campo `Format` nella struttura MQWIH.

z/OS Su z/OS, specificare l'opzione MQGMO_CONVERT nella chiamata MQGET per convertire i dati utente in messaggi con formato MQFMT_WORK_INFO_HEADER. Tuttavia, la stessa struttura MQWIH viene sempre restituita nella serie di caratteri e nella codifica del gestore code (ovvero, la struttura MQWIH viene convertita indipendentemente dalla specifica o meno dell'opzione MQGMO_CONVERT).

Per il linguaggio di programmazione C, viene definita anche la costante MQFMT_WORK_INFO_HEADER_ARRAY, che ha lo stesso valore di MQFMT_WORK_INFO_HEADER, ma è un array di caratteri anziché una stringa.

MQFMT_XMIT_Q_HEADER
I dati del messaggio iniziano con l'intestazione della coda di trasmissione MQXQH. I dati provenienti dal messaggio originale seguono immediatamente la struttura MQXQH. Il nome formato dei dati del messaggio originale viene fornito dal campo `Format` della struttura MQMD, che fa parte dell'intestazione della coda di trasmissione MQXQH. Consultare “MQXQH - Intestazione coda di trasmissione” a pagina 635 per i dettagli di questa struttura.

I report COA e COD non sono generati per i messaggi che hanno un `Format` di MQFMT_XMIT_Q_HEADER.

Per il linguaggio di programmazione C, viene definito anche il valore costante MQFMT_XMIT_Q_HEADER_ARRAY, che ha lo stesso valore di MQFMT_XMIT_Q_HEADER, ma è un array di caratteri anziché una stringa.

Questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . La lunghezza di questo campo è fornita da MQ_FORMAT_LENGTH. Il valore iniziale di questo campo è MQFMT_NONE.

Priorità (MQLONG) per MQMD

Per le chiamate MQPUT e MQPUT1 , il valore deve essere maggiore o uguale a zero; zero è la priorità più bassa. È possibile utilizzare anche il valore speciale seguente:

MQPRI_PRIORITY_AS_Q_DEF

- Se la coda è una coda cluster, la priorità per il messaggio viene presa dall'attributo **DefPriority** definito nel gestore code *di destinazione* che possiede la particolare istanza della coda in cui è collocato il messaggio.

  Quando esistono più istanze della coda cluster e differiscono in questo attributo, viene selezionato il valore di uno di essi e non è possibile prevedere quale sarà utilizzato. È quindi necessario impostare questo attributo sullo stesso valore per tutte le istanze. In caso contrario, viene emesso il messaggio di errore AMQ9407 nei log del gestore code. Consultare anche How are destination object attributes resolved for aliases, remote and cluster queues?

  Il valore di *DefPriority* viene copiato nel campo *Priority* quando il messaggio viene inserito nella coda di destinazione. Se *DefPriority* viene modificato successivamente, i messaggi che sono già stati inseriti nella coda non vengono influenzati.
- Se la coda non è una coda cluster, la priorità per il messaggio viene presa dall'attributo **DefPriority** definito nel gestore code *locale* , anche se il gestore code di destinazione è remoto.

  Se è presente più di una definizione nel percorso di risoluzione del nome della coda, la priorità predefinita viene presa dal valore di questo attributo nella *prima* definizione nel percorso. Questo può essere:

  – Una coda alias
  – Una coda locale
  – Una definizione locale di una coda remota
  – Un alias del gestore code
  – Una coda di trasmissione (ad esempio, la coda *DefXmitQName*)

  Il valore *DefPriority* viene copiato nel campo *Priority* quando il messaggio viene inserito. Se *DefPriority* viene modificato successivamente, i messaggi già inseriti non vengono influenzati.

Il valore restituito dalla chiamata MQGET è sempre maggiore o uguale a zero; il valore MQPRI_PRIORITY_AS_Q_DEF non viene mai restituito.

Se un messaggio viene inserito con una priorità superiore al massimo supportato dal gestore code locale (questo massimo viene fornito dall'attributo del gestore code **MaxPriority**), il messaggio viene accettato dal gestore code, ma posizionato sulla coda alla massima priorità del gestore code; la chiamata MQPUT o MQPUT1 viene completata con MQCC_WARNING e il codice motivo MQRC_PRIORITY_EXCEEDS_MAXIMUM. Tuttavia, il campo *Priority* conserva il valore specificato dall'applicazione che ha inserito il messaggio.

Su z/OS, se un messaggio con un numero MsgSeqpari a 1 viene inserito in una coda che ha una sequenza di distribuzione del messaggio MQMDS_PRIORITY e un tipo di indice MQIT_GROUP_ID, la coda potrebbe trattare il messaggio con una priorità differente. Se il messaggio è stato inserito nella coda con una priorità 0 o 1, viene elaborato come se avesse una priorità 2. Questo perché l'ordine dei messaggi posizionati su questo tipo di coda è ottimizzato per abilitare test di completezza del gruppo efficienti. Per ulteriori informazioni sulla sequenza di consegna dei messaggi MQMDS_PRIORITY e sul tipo di indice MQIT_GROUP_ID, consultare MsgDeliveryMsgDelivery.

Quando si risponde a un messaggio, le applicazioni devono utilizzare la priorità del messaggio di richiesta per il messaggio di risposta. In altre situazioni, specificando MQPRI_PRIORITY_AS_Q_DEF è possibile eseguire l'ottimizzazione della priorità senza modificare l'applicazione.

Questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è MQPRI_PRIORITY_AS_Q_DEF.

Persistenza (MQLONG) per MQMD

Indica se il messaggio sopravvive agli errori di sistema e riavvia il gestore code. Per le chiamate MQPUT e MQPUT1 , il valore deve essere uno dei seguenti:

PERSISTORA_MQPER_
Il messaggio sopravvive agli errori di sistema e al riavvio del gestore code. Una volta che il messaggio è stato inserito e l'unità di lavoro in cui è stato inserito è stata sottoposta a commit (se il messaggio è inserito come parte di un'unità di lavoro), il messaggio viene conservato nella memoria ausiliaria. Rimane lì fino a quando il messaggio non viene rimosso dalla coda e l'unità di lavoro in cui è stato eseguito il commit (se il messaggio viene richiamato come parte di un'unità di lavoro).

Quando un messaggio persistente viene inviato a una coda remota, un meccanismo di memorizzazione e inoltro conserva il messaggio in ciascun gestore code lungo l'instradamento alla destinazione, fino a quando non si sa che il messaggio è arrivato al gestore code successivo.

I messaggi persistenti non possono essere posizionati su:

- Code dinamiche temporanee
- Code condivise che si associano a un oggetto CFSTRUCT a CFLEVEL (2) o inferiore o dove l'oggetto CFSTRUCT è definito come RECOVER (NO).

I messaggi persistenti possono essere posizionati su code dinamiche permanenti e code predefinite.

MQPER_NOT_PERSISTENT
Il messaggio di solito non sopravvive agli errori di sistema o al riavvio del gestore code. Ciò si applica anche se una copia intatta del messaggio viene trovata nella memoria ausiliaria al riavvio del gestore code.

Nel caso di NPMCLASS (HIGH), i messaggi non persistenti sopravvivono a un normale arresto e riavvio del gestore code.

Nel caso di code condivise, i messaggi non persistenti sopravvivono ai riavvii del gestore code nel gruppo di condivisione code, ma non sopravvivono agli errori della CF utilizzata per memorizzare i messaggi nelle code condivise.

MQPER_PERSISTENCE_AS_Q_DEF

- Se la coda è una coda cluster, la persistenza del messaggio viene ricavata dall'attributo **DefPersistence** definito nel gestore code *di destinazione* che possiede la particolare istanza della coda in cui è collocato il messaggio.

  Quando esistono più istanze della coda cluster e differiscono in questo attributo, viene selezionato il valore di uno di essi e non è possibile prevedere quale sarà utilizzato. È quindi necessario impostare questo attributo sullo stesso valore per tutte le istanze. In caso contrario, viene emesso il messaggio di errore AMQ9407 nei log del gestore code. Consultare anche How are destination object attributes resolved for aliases, remote and cluster queues?

  Il valore di *DefPersistence* viene copiato nel campo *Persistence* quando il messaggio viene inserito nella coda di destinazione. Se *DefPersistence* viene modificato successivamente, i messaggi che sono già stati inseriti nella coda non vengono influenzati.
- Se la coda non è una coda cluster, la persistenza del messaggio viene ricavata dall'attributo **DefPersistence** definito nel gestore code *locale* , anche se il gestore code di destinazione è remoto.

  Se nel percorso di risoluzione del nome coda è presente più di una definizione, la persistenza predefinita viene ricavata dal valore di questo attributo nella *prima* definizione nel percorso. Questo può essere:

  – Una coda alias
  – Una coda locale
  – Una definizione locale di una coda remota
  – Un alias del gestore code
  – Una coda di trasmissione (ad esempio, la coda *DefXmitQName*)

Il valore *DefPersistence* viene copiato nel campo *Persistence* quando il messaggio viene inserito. Se *DefPersistence* viene modificato successivamente, i messaggi già inseriti non vengono influenzati.

Sia i messaggi persistenti che quelli non persistenti possono esistere nella stessa coda.

Quando si risponde ad un messaggio, le applicazioni devono utilizzare la persistenza del messaggio di richiesta per il messaggio di risposta.

Per una chiamata MQGET, il valore restituito è MQPER_PERSISTENT o MQPER_NOT_PERSISTENT.

Questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è MQPER_PERSISTENCE_AS_Q_DEF.

MsgId (MQBYTE24) per MQMD

È una stringa di byte utilizzata per distinguere un messaggio da un altro. Generalmente, due messaggi non devono avere lo stesso identificativo di messaggio, sebbene ciò non sia consentito dal gestore code. L'identificativo del messaggio è una proprietà permanente del messaggio e persiste durante i riavvii del gestore code. Poiché l'identificativo del messaggio è una stringa di byte e non una stringa di caratteri, l'identificativo del messaggio non viene convertito tra le serie di caratteri quando il messaggio passa da un gestore code all'altro.

Per le chiamate MQPUT e MQPUT1 , se MQMI_NONE o MQPMO_NEW_MSG_ID è specificato dall'applicazione, il gestore code genera un identificativo del messaggio univoco[3]quando il messaggio viene inserito e lo inserisce nel descrittore del messaggio inviato con il messaggio. Il gestore code restituisce questo identificativo del messaggio anche nel descrittore del messaggio appartenente all'applicazione mittente. L'applicazione può utilizzare questo valore per registrare informazioni su messaggi particolari e per rispondere alle query provenienti da altre parti dell'applicazione.

Se il messaggio viene inserito in un argomento, il gestore code genera identificatori di messaggi univoci come necessario per ogni messaggio pubblicato. Se MQPMO_NEW_MSG_ID è specificato dall'applicazione, il gestore code genera un identificativo di messaggio univoco da restituire all'output. Se MQMI_NONE viene specificato dall'applicazione, il valore del campo *MsgId* in MQMD non viene modificato al ritorno dalla chiamata.

Consultare la descrizione di MQPMO_RETAIN in “Opzioni (MQLONG) per MQPMO” a pagina 518 per ulteriori informazioni sulle pubblicazioni conservate.

Se il messaggio viene inserito in un elenco di distribuzione, il gestore code genera identificativi di messaggi univoci come necessario, ma il valore del campo *MsgId* in MQMD non viene modificato al ritorno dalla chiamata, anche se è stato specificato MQMI_NONE o MQPMO_NEW_MSG_ID. Se l'applicazione deve conoscere gli identificatori dei messaggi generati dal gestore code, deve fornire i record MQPMR contenenti il campo *MsgId* .

L'applicazione di invio può specificare anche un valore per l'identificativo del messaggio diverso da MQMI_NONE; questo arresta il gestore code che genera un identificativo del messaggio univoco. Un'applicazione che sta inoltrando un messaggio può utilizzarlo per trasmettere l'identificativo del messaggio originale.

Il gestore code non utilizza questo campo se non per:

---

[3] Un *MsgId* generato dal gestore code è costituito da un identificativo del prodotto a 4 byte (AMQ¬ o CSQ¬ in ASCII o EBCDIC, dove ¬ rappresenta un carattere vuoto), seguito da un'implementazione specifica del prodotto di una stringa univoca. In IBM MQ contiene i primi 12 caratteri del nome del gestore code e un valore derivato dall'orologio di sistema. Tutti i gestori code che possono intercomunicare devono quindi avere nomi che differiscono nei primi 12 caratteri, al fine di garantire che gli identificatori dei messaggi siano univoci. La possibilità di generare una stringa univoca dipende anche dal fatto che l'orologio di sistema non venga modificato all'indietro. Per eliminare la possibilità che un identificativo di messaggio generato dal gestore code ne duplici uno generato dall'applicazione, l'applicazione deve evitare di creare identificativi con caratteri iniziali compresi nell'intervallo da A a I in ASCII o EBCDIC (da X'41 'a X'49' e da X'C1'a X'C9'). Tuttavia, all'applicazione non viene impedito di generare identificatori con caratteri iniziali in questi intervalli.

- Genera un valore univoco se richiesto, come descritto in precedenza
- Consegna il valore all'applicazione che emette la richiesta get per il messaggio
- Copiare il valore nel campo *CorrelId* di qualsiasi messaggio di report generato su questo messaggio (in base alle opzioni *Report*)

Quando il gestore code o un agent del canale dei messaggi genera un messaggio di report, imposta il campo *MsgId* nel modo specificato dal campo *Report* del messaggio originale, MQRO_NEW_MSG_ID o MQRO_PASS_MSG_ID. Anche le applicazioni che generano messaggi di report devono eseguire questa operazione.

Per la chiamata MQGET, *MsgId* è uno dei cinque campi che è possibile utilizzare per richiamare uno specifico messaggio dalla coda. Di solito la chiamata MQGET restituisce il messaggio successivo sulla coda, ma è possibile ottenere un particolare messaggio specificando uno o più dei cinque criteri di selezione, in qualsiasi combinazione; questi campi sono:

- *MsgId*
- *CorrelId*
- *GroupId*
- *MsgSeqNumber*
- *Offset*

L'applicazione imposta uno o più di questi campi sui valori richiesti, quindi imposta le opzioni di corrispondenza MQMO_ * corrispondenti nel campo *MatchOptions* in MQGMO per utilizzare tali campi come criteri di selezione. Solo i messaggi con i valori specificati in tali campi sono candidati per il richiamo. Il valore predefinito per il campo *MatchOptions* (se non modificato dall'applicazione) corrisponde sia all'identificativo del messaggio che all'identificativo di correlazione.

Su z/OS, il criterio di selezione che può essere utilizzato è limitato dal tipo di indice utilizzato per la coda. Consultare l'attributo della coda **IndexType** per ulteriori dettagli.

Normalmente, il messaggio restituito è il *primo* messaggio nella coda che soddisfa i criteri di selezione. Tuttavia, se viene specificato MQGMO_BROWSE_NEXT, il messaggio restituito è il messaggio *successivo* che soddisfa i criteri di selezione; la scansione per questo messaggio inizia con il messaggio *che segue* la posizione corrente del cursore.

Nota: La coda viene sottoposta a scansione in modo sequenziale per un messaggio che soddisfa i criteri di selezione, quindi i tempi di richiamo sono più lenti rispetto a quelli in cui non viene specificato alcun criterio di selezione, specialmente se è necessario eseguire la scansione di molti messaggi prima che ne venga trovato uno adatto. Le eccezioni sono:

- Multi una chiamata MQGET da parte di *CorrelId* su Multiplatforms a 64 bit dove l'indice *CorrelId* elimina la necessità di eseguire una scansione sequenziale true.
- z/OS una chiamata MQGET da *IndexType* su z/OS.

In entrambi i casi, le prestazioni di recupero sono migliorate.

Consultare Tabella 495 a pagina 403 per ulteriori informazioni su come vengono utilizzati i criteri di selezione in varie situazioni.

La specifica di MQMI_NONE come identificativo del messaggio ha lo stesso effetto di non specificare MQMO_MATCH_MSG_ID, ossia *qualsiasi* identificativo del messaggio corrisponde.

Questo campo viene ignorato se l'opzione MQGMO_MSG_UNDER_CURSOR è specificata nel parametro **GetMsgOpts** nella chiamata MQGET.

Alla restituzione da una chiamata MQGET, il campo *MsgId* viene impostato sull'identificativo del messaggio restituito (se presente).

È possibile utilizzare il seguente valore speciale:

MQMI_NONE
Nessun identificativo di messaggio specificato.

Il valore è zero binario per la lunghezza del campo.

Per il linguaggio di programmazione C, viene anche definita la costante MQMI_NONE_ARRAY, che ha lo stesso valore di MQMI_NONE, ma è un array di caratteri invece di una stringa.

Questo è un campo di input / output per le chiamate MQGET, MQPUT e MQPUT1 . La lunghezza di questo campo è fornita da MQ_MSG_ID_LENGTH. Il valore iniziale di questo campo è MQMI_NONE.

CorrelId (MQBYTE24) per MQMD

Il campo CorrelId è una proprietà nell'intestazione del messaggio che può essere utilizzata per identificare un messaggio specifico o un gruppo di messaggi.

Si tratta di una stringa di byte che l'applicazione può utilizzare per correlare un messaggio ad un altro o per correlare il messaggio ad un altro lavoro che l'applicazione sta eseguendo. L'identificativo di correlazione è una proprietà permanente del messaggio e persiste nei riavvii del gestore code. Poiché l'identificativo di correlazione è una stringa di byte e non una stringa di caratteri, l'identificativo di correlazione non viene convertito tra serie di caratteri quando il messaggio passa da un gestore code all'altro.

Per le chiamate MQPUT e MQPUT1 , l'applicazione può specificare qualsiasi valore. Il gestore code trasmette questo valore con il messaggio e lo consegna all'applicazione che emette la richiesta get per il messaggio.

Se l'applicazione specifica MQPMO_NEW_CORREL_ID, il gestore code genera un identificativo di correlazione univoco che viene inviato con il messaggio e restituito anche all'applicazione di invio all'output dalla chiamata MQPUT o MQPUT1 .

Un identificativo di correlazione generato dal gestore code è costituito da un identificativo di prodotto a 3 byte (AMQ o CSQ in ASCII o EBCDIC), seguito da un byte riservato e da un'implementazione specifica del prodotto di una stringa univoca. In IBM MQ , questa stringa di implementazione specifica del prodotto contiene i primi 12 caratteri del nome del gestore code e un valore derivato dall'orologio di sistema. Tutti i gestori code che possono intercomunicare devono quindi avere nomi diversi nei primi 12 caratteri per garantire che gli identificativi dei messaggi siano univoci. La possibilità di generare una stringa univoca dipende anche dal fatto che l'orologio di sistema non venga modificato all'indietro. Per eliminare la possibilità che un identificativo di messaggio generato dal gestore code ne duplici uno generato dall'applicazione, l'applicazione deve evitare di creare identificativi con caratteri iniziali compresi nell'intervallo da A a I in ASCII o EBCDIC (da X'41 'a X'49' e da X'C1'a X'C9'). Tuttavia, all'applicazione non viene impedito di generare identificatori con caratteri iniziali in questi intervalli.

Questo identificativo di correlazione generato viene conservato con il messaggio se viene conservato e viene utilizzato come identificativo di correlazione quando il messaggio viene inviato come pubblicazione ai sottoscrittori che specificano MQCI_NONE nel campo ID SubCorrelnell'MQSD passato sulla chiamata MQSUB. Consultare Opzioni MQPMO per ulteriori dettagli sulle pubblicazioni conservate.

Quando il gestore code o un agent del canale dei messaggi genera un messaggio di report, imposta il campo *CorrelId* nel modo specificato dal campo *Report* del messaggio originale, MQRO_COPY_MSG_ID_TO_CORREL_ID o MQRO_PASS_CORREL_ID. Anche le applicazioni che generano messaggi di report devono eseguire questa operazione.

Per la chiamata MQGET, *CorrelId* è uno dei cinque campi che è possibile utilizzare per selezionare un determinato messaggio da richiamare dalla coda. Consultare la descrizione del campo *MsgId* per dettagli su come specificare i valori per questo campo.

La specifica di MQCI_NONE come identificativo di correlazione ha lo stesso effetto di non specificare MQMO_MATCH_CORREL_ID, ossia *qualsiasi* identificativo di correlazione corrisponderà.

Se l'opzione MQGMO_MSG_UNDER_CURSOR è specificata nel parametro **GetMsgOpts** nella chiamata MQGET, questo campo viene ignorato.

Al ritorno da una chiamata MQGET, il campo *CorrelId* è impostato sull'identificativo di correlazione del messaggio restituito (se presente).

È possibile utilizzare i seguenti valori speciali:

MQCI_NONE
Non è stato specificato alcun identificatore di correlazione.

Il valore è zero binario per la lunghezza del campo.

Per il linguaggio di programmazione C, viene definita anche la costante MQCI_NONE_ARRAY, che ha lo stesso valore di MQCI_NONE, ma è un array di caratteri invece di una stringa.

SESSIONE MQCI_NEW_
Il messaggio è l'inizio di una nuova sessione.

Questo valore viene riconosciuto da CICS bridge come indica l'avvio di una nuova sessione, ossia l'avvio di una nuova sequenza di messaggi.

Per il linguaggio di programmazione C, viene definita anche la costante MQCI_NEW_SESSION_ARRAY, che ha lo stesso valore di MQCI_NEW_SESSION, ma è un array di caratteri invece di una stringa.

Per la chiamata MQGET, questo è un campo di input/output. Per chiamate MQPUT e MQPUT1 , questo è un campo di input se MQPMO_NEW_CORREL_ID non è specificato e un campo di output se MQPMO_NEW_CORREL_ID è specificato. La lunghezza di questo campo è fornita da MQ_CORREL_ID_LENGTH. Il valore iniziale di questo campo è MQCI_NONE.

Nota:

Non è possibile passare l'identificatore di correlazione di una pubblicazione in una gerarchia. Il campo viene utilizzato dal gestore code.

BackoutCount (MQLONG) per MQMD

Questo è un conteggio del numero di volte in cui il messaggio è stato precedentemente restituito dalla chiamata MQGET come parte di un'unità di lavoro e successivamente ne è stato eseguito il backout. Consente all'applicazione di rilevare gli errori di elaborazione basati sul contenuto del messaggio. Il conteggio esclude le chiamate MQGET che specificano le opzioni MQGMO_BROWSE_ *.

La precisione di questo conteggio è influenzata dall'attributo della coda **HardenGetBackout** ; consultare “Attributi per le code” a pagina 858.

Su z/OS, un valore di 255 indica che il messaggio è stato ripristinato 255 o più volte; il valore restituito non è mai maggiore di 255.

Questo è un campo di output per la chiamata MQGET. Viene ignorato per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è `0`.

ReplyToQ (MQCHAR48) per MQMD

Questo è il nome della coda messaggi a cui l'applicazione che ha emesso la richiesta get per il messaggio invia i messaggi MQMT_REPLY e MQMT_REPORT. Il nome è il nome locale di una coda definita nel gestore code identificato da *`ReplyToQMgr`*. Questa coda non deve essere una coda modello, anche se il gestore code di invio non lo verifica quando viene inserito il messaggio.

Per le chiamate MQPUT e MQPUT1 , questo campo non deve essere vuoto se il campo *`MsgType`* ha il valore MQMT_REQUEST o se i messaggi di report sono richiesti dal campo *`Report`* . Tuttavia, il valore specificato (o sostituito) viene passato all'applicazione che emette la richiesta get per il messaggio, indipendentemente dal tipo di messaggio.

Se il campo *`ReplyToQMgr`* è vuoto, il gestore code locale ricerca il nome *`ReplyToQ`* nelle proprie definizioni di coda. Se esiste una definizione locale di una coda remota con questo nome, il valore *`ReplyToQ`* nel messaggio trasmesso viene sostituito dal valore dell'attributo **RemoteQName** dalla definizione della coda remota e questo valore viene restituito nel descrittore del messaggio quando l'applicazione ricevente emette una chiamata MQGET per il messaggio. Se non esiste una definizione locale di una coda remota, *`ReplyToQ`* non viene modificato.

Se il nome viene specificato, può contenere spazi vuoti finali; il primo carattere null e i caratteri che lo seguono vengono trattati come spazi vuoti. In caso contrario, non viene effettuato alcun controllo che il nome soddisfi le regole di denominazione per le code; ciò è vero anche per il nome trasmesso, se il

ReplyToQ viene sostituito nel messaggio trasmesso. L'unico controllo effettuato è che è stato specificato un nome, se le circostanze lo richiedono.

Se non è richiesta una coda di risposta, impostare il campo *ReplyToQ* su spazi vuoti oppure (nel linguaggio di programmazione C) sulla stringa nulla oppure su uno o più spazi vuoti seguiti da un carattere null; non lasciare il campo non inizializzato.

Per la chiamata MQGET, il gestore code restituisce sempre il nome completato con spazi vuoti alla lunghezza del campo.

Se un messaggio che richiede un messaggio di report non può essere consegnato e anche il messaggio di report non può essere consegnato alla coda specificata, sia il messaggio originale che il messaggio di report vanno alla coda di messaggi non recapitabili (messaggio non recapitato) (consultare l'attributo **DeadLetterQName** descritto in "Attributi per il gestore code" a pagina 818).

Questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . La lunghezza di questo campo è fornita da MQ_Q_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

ReplyToGestore code (MQCHAR48) per MQMD

Questo è il nome del gestore code a cui inviare il messaggio di risposta o il messaggio di report. *ReplyToQ* è il nome locale di una coda definita su questo gestore code.

Se il campo *ReplyToQMgr* è vuoto, il gestore code locale ricerca il nome *ReplyToQ* nelle relative definizioni di coda. Se esiste una definizione locale di una coda remota con questo nome, il valore *ReplyToQMgr* nel messaggio trasmesso viene sostituito dal valore dell'attributo **RemoteQMgrName** dalla definizione della coda remota e questo valore viene restituito nel descrittore del messaggio quando l'applicazione ricevente emette una chiamata MQGET per il messaggio. Se una definizione locale di una coda remota non esiste, il *ReplyToQMgr* che viene trasmesso con il messaggio è il nome del gestore code locale.

Se il nome viene specificato, può contenere spazi vuoti finali; il primo carattere null e i caratteri che lo seguono vengono trattati come spazi vuoti. In caso contrario, non viene effettuato alcun controllo che il nome soddisfi le regole di denominazione per i gestori code o che questo nome sia noto al gestore code mittente; ciò è valido anche per il nome trasmesso, se *ReplyToQMgr* viene sostituito nel messaggio trasmesso.

Se non è richiesta una coda di risposta, impostare il campo *ReplyToQMgr* su spazi vuoti oppure (nel linguaggio di programmazione C) sulla stringa nulla oppure su uno o più spazi vuoti seguiti da un carattere null; non lasciare il campo non inizializzato.

Per la chiamata MQGET, il gestore code restituisce sempre il nome completato con spazi vuoti alla lunghezza del campo.

Questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . La lunghezza di questo campo viene fornita da MQ_Q_MGR_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

UserIdentifier (MQCHAR12) per MQMD

Fa parte del **contesto di identità** del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare "MQMD - Descrittore messaggi" a pagina 431 e Contesto del messaggio.

UserIdentifier specifica l'ID utente dell'applicazione che ha originato il messaggio. Il gestore code considera queste informazioni come dati carattere, ma non ne definisce il formato.

Una volta ricevuto un messaggio, utilizzare *UserIdentifier* nel campo *AlternateUserId* del parametro **ObjDesc** di una chiamata MQOPEN o MQPUT1 successiva per eseguire il controllo dell'autorizzazione per l'utente *UserIdentifier* invece che per l'applicazione che esegue l'apertura.

Quando il gestore code genera queste informazioni per una chiamata MQPUT o MQPUT1 :

- Su z/OS, il gestore code utilizza il parametro *AlternateUserId* dal parametro **ObjDesc** della chiamata MQOPEN o MQPUT1 se è stata specificata l'opzione MQOO_ALTERNATE_USER_AUTHORITY

o MQPMO_ALTERNATE_USER_AUTHORITY. Se l'opzione pertinente non è stata specificata, il gestore code utilizza un identificativo utente determinato dall'ambiente.

- In altri ambienti, il gestore code utilizza sempre un identificativo utente determinato dall'ambiente.

Quando l'identificativo utente viene determinato dall'ambiente:

- Su z/OS, il gestore code utilizza:
  - Per MVS (batch), l'identificativo utente dalla scheda JES JOB o dal task avviato
  - Per TSO, l'identificativo utente propagato al lavoro durante l'inoltro del lavoro
  - Per CICS, l'identificativo utente associato all'attività
  - Per IMS, l'identificativo utente dipende dal tipo di applicazione:
    - Per:
      - Regioni BMP non messaggio
      - Regioni IFP non messaggi
      - BMP messaggio e regioni IFP messaggio che non hanno emesso una chiamata GU con esito positivo

      il gestore code utilizza l'identificativo utente dalla scheda JES JOB della regione o l'identificativo utente TSO. Se questi sono vuoti o nulli, utilizza il nome del blocco di specifica del programma (PSB).
    - Per:
      - BMP del messaggio e regioni IFP del messaggio che *hanno* emesso una chiamata GU con esito positivo
      - Regioni MPP

      il gestore code utilizza uno dei seguenti:
      - L'identificativo utente collegato associato al messaggio
      - Il nome del terminale logico (LTERM)
      - L'identificativo utente della scheda JES JOB della regione
      - L'identificativo utente TSO
      - Il nome PSB
- Su IBM i, il gestore code utilizza il nome del profilo utente associato con il lavoro dell'applicazione.
- Su AIX and Linux, il gestore code utilizza:
  - Il nome di accesso dell'applicazione
  - L'identificativo utente effettivo del processo se non è disponibile alcun accesso
  - L'identificativo utente associato alla transazione, se l'applicazione è una transazione CICS
- Sui sistemi Windows , il gestore code utilizza i primi 12 caratteri del nome utente collegato.

Questo campo è normalmente un campo di output generato dal gestore code, ma per una chiamata MQPUT o MQPUT1 è possibile rendere questo campo un campo di input / output e specificare il campo UserIdentification invece di consentire al gestore code di generare queste informazioni. Specificare MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT nel parametro PutMsgOpts e specificare un ID utente nel campo UserIdentifier se non si desidera che il gestore code generi il campo UserIdentifier per una chiamata MQPUT o MQPUT1 .

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT è specificato nel parametro **PutMsgOpts** . Tutte le informazioni che seguono un carattere null all'interno del campo vengono scartate. Il gestore code converte il carattere null e i seguenti caratteri in spazi vuoti. Se MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT non è specificato, questo campo viene ignorato nell'input ed è un campo di sola emissione.

Dopo il corretto completamento di una chiamata MQPUT o MQPUT1 , questo campo contiene il *UserIdentifier* che è stato trasmesso con il messaggio se è stato inserito in una coda. Questo sarà il valore di *UserIdentifier* che viene conservato con il messaggio se viene conservato (vedere la descrizione di MQPMO_RETAIN per ulteriori dettagli sulle pubblicazioni conservate) ma non viene utilizzato come *UserIdentifier* quando il messaggio viene inviato come pubblicazione ai sottoscrittori perché forniscono un valore da sovrascrivere *UserIdentifier* in tutte le pubblicazioni a loro inviate. Se il messaggio non ha contesto, il campo è completamente vuoto.

Questo è un campo di output per la chiamata MQGET. La lunghezza di questo campo è fornita da MQ_USER_ID_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 12 caratteri vuoti in altri linguaggi di programmazione.

AccountingToken (MQBYTE32) per MQMD

Questo è il token di account, parte del *contesto identità* del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare "MQMD - Descrittore messaggi" a pagina 431 e Contesto del messaggio.

`AccountingToken` consente ad una applicazione di addebitare in maniera appropriata il lavoro eseguito come risultato del messaggio. Il gestore code tratta queste informazioni come una stringa di bit e non ne controlla il contenuto.

Il gestore code genera queste informazioni nel modo seguente:

- Il primo byte del campo è impostato sulla lunghezza delle informazioni di account presenti nei byte che seguono; questa lunghezza è compresa nell'intervallo tra zero e 30 e viene memorizzata nel primo byte come un numero intero binario.
- Il secondo byte e i byte successivi (come specificato dal campo della lunghezza) vengono impostati sulle informazioni di account appropriate per l'ambiente.
  - z/OS Su z/OS le informazioni di account sono impostate su:
    - Per il batch z/OS , le informazioni di account dalla scheda JES JOB o da un'istruzione JES ACCT nella scheda EXEC (i separatori virgola vengono modificati in X'FF '). Queste informazioni vengono troncate, se necessario, a 31 byte.
    - Per TSO, il numero di account dell'utente.
    - Per CICS, l'identificativo dell'unità di lavoro LU 6.2 (UEPUOWDS) (26 byte).
    - Per IMS, il nome PSB di 8 caratteri concatenato con il token di recupero IMS di 16 caratteri.
  - IBM i Su IBM i, le informazioni di account sono impostate sul codice di account per il lavoro.
  - Linux AIX Su AIX and Linux, le informazioni di account sono impostate sull'identificativo utente numerico, in caratteri ASCII.
  - Windows Su Windows, le informazioni di account sono impostate su un SID (security identifier) Windows in un formato compresso. Il SID identifica in modo univoco l'identificativo utente memorizzato nel campo *UserIdentifier* . Quando il SID viene memorizzato nel campo *AccountingToken* , l'autorità di identificazione a 6 byte (ubicata nel terzo byte e nei byte successivi del SID) viene omessa. Ad esempio, se il SID Windows ha una lunghezza di 28 byte, nel campo *AccountingToken* vengono memorizzati 22 byte di informazioni SID.
- L'ultimo byte (byte 32) del campo di account è impostato sul tipo di token di account (in questo caso MQACTT_NT_SECURITY_ID, x '0b'):

  ID_CICS_LUOW MQACTT_
  Identificativo LUOW CICS .

  Windows **ID_MQACTT_NT_SECURITY_**
  Identificativo di sicurezza Windows .

  IBM i **MQACTT_OS400_ACCOUNT_TOKEN**
  Token di account IBM i .

UNIX **ID_NUMERIC_UNIX_MQACTT_**
Identificativo numerico UNIX .

UTENTE MQACT
Token di account definito dall'utente.

MQACTT_SCONOSCIUTO
Tipo di token di conteggio sconosciuto.

Il tipo di token di account è impostato su un valore esplicito solo nei seguenti ambienti:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Windows Windows

e per IBM MQ MQI clients collegati a questi sistemi. In altri ambienti, il tipo di token di account è impostato sul valore MQACTT_UNKNOWN. In questi ambienti utilizzare il campo *PutApplType* per dedurre il tipo di token di account ricevuto.

- Tutti gli altri byte sono impostati su zero binario.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT è specificato nel parametro **PutMsgOpts** . Se non viene specificato né MQPMO_SET_IDENTITY_CONTEXT né MQPMO_SET_ALL_CONTEXT, questo campo viene ignorato nell'input ed è un campo di sola emissione. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio.

Dopo il corretto completamento di una chiamata MQPUT o MQPUT1 , questo campo contiene il `AccountingToken` che è stato trasmesso con il messaggio se è stato inserito in una coda. Questo sarà il valore di `AccountingToken` che viene conservato con il messaggio se viene conservato (consultare la descrizione di MQPMO_RETAIN in “Opzioni (MQLONG) per MQPMO” a pagina 518 per ulteriori dettagli sulle pubblicazioni conservate) ma non viene utilizzato come `AccountingToken` quando il messaggio viene inviato come una pubblicazione ai sottoscrittori poiché forniscono un valore da sovrascrivere `AccountingToken` in tutte le pubblicazioni a loro inviate. Se il messaggio non ha contesto, il campo è completamente binario zero.

Questo è un campo di output per la chiamata MQGET.

Questo campo non è soggetto ad alcuna conversione basata sulla serie di caratteri del gestore code; il campo viene trattato come una stringa di bit e non come una stringa di caratteri.

Il gestore code non esegue alcuna operazione con le informazioni in questo campo. L'applicazione deve interpretare le informazioni se desidera utilizzarle per scopi contabili.

È possibile utilizzare il seguente valore speciale per il campo `AccountingToken` :

MQACT_NONE
Nessun token di account specificato.

Il valore è zero binario per la lunghezza del campo.

Per il linguaggio di programmazione C, viene definita anche la costante MQACT_NONE_ARRAY, che ha lo stesso valore di MQACT_NONE, ma è un insieme di caratteri anziché una stringa.

La lunghezza di questo campo è fornita da MQ_ACCOUNTING_TOKEN_LENGTH. Il valore iniziale di questo campo è MQACT_NONE.

Dati ApplIdentity(MQCHAR32) per MQMD

Fa parte del **contesto di identità** del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare “MQMD - Descrittore messaggi” a pagina 431 e Contesto del messaggio.

ApplIdentityData sono informazioni definite dalla suite di applicazioni e possono essere utilizzate per fornire ulteriori informazioni sul messaggio o sul suo creatore. Il gestore code considera queste informazioni come dati carattere, ma non ne definisce il formato. Quando il gestore code genera queste informazioni, è completamente vuoto.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT è specificato nel parametro **PutMsgOpts** . Se è presente un carattere null, il valore null e i seguenti caratteri vengono convertiti in spazi vuoti dal gestore code. Se non viene specificato né MQPMO_SET_IDENTITY_CONTEXT né MQPMO_SET_ALL_CONTEXT, questo campo viene ignorato nell'input ed è un campo di sola emissione. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio.

Dopo il corretto completamento di una chiamata MQPUT o MQPUT1 , questo campo contiene il *ApplIdentityData* che è stato trasmesso con il messaggio se è stato inserito in una coda. Questo sarà il valore di *ApplIdentityData* che viene conservato con il messaggio se viene conservato (vedere la descrizione di MQPMO_RETAIN per ulteriori dettagli sulle pubblicazioni conservate) ma non viene utilizzato come *ApplIdentityData* quando il messaggio viene inviato come pubblicazione ai sottoscrittori perché forniscono un valore da sovrascrivere *ApplIdentityData* in tutte le pubblicazioni a loro inviate. Se il messaggio non ha contesto, il campo è completamente vuoto.

Questo è un campo di output per la chiamata MQGET. La lunghezza di questo campo è fornita da MQ_APPL_IDENTITY_DATA_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 32 caratteri vuoti in altri linguaggi di programmazione.

PutApplTipo (MQLONG) per MQMD

Questo è il tipo di applicazione che inserisce il messaggio e fa parte del **contesto di origine** del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare “MQMD - Descrittore messaggi” a pagina 431 e Contesto del messaggio.

PutApplType può avere uno dei seguenti tipi standard. È anche possibile definire i propri tipi, ma solo con valori compresi nell'intervallo tra MQAT_USER_FIRST e MQAT_USER_LAST.

AIX MQAT
Applicazione AIX (stesso valore di MQAT_UNIX).

MQAT_AMQP
Applicazione protocollo AMQP

MEDIA_MQAT_BROKER
Broker.

MQAT_CICS
Transazione CICS .

BRIDGE - MQAT_CICS_BRIDGE
CICS bridge.

CICS_VSE MQAT
Transazione CICS/VSE .

DOS MQAT
Applicazione IBM MQ MQI client su PC DOS.

DQM MQAT
Agent gestore code distribuito.

TUTORE_QAT_MQ
Applicazione Guardian tandem (stesso valore di MQAT_NSK).

IMS MQAT
Applicazione IMS .

MQAT_IM_Bridge
Bridge IMS .

JAVA MQAT
Java.

MVS MQAT
Applicazione MVS o TSO (stesso valore di MQAT_ZOS).

MQAT_NOTES_AGENT
Lotus Notes Applicazione agent.

MQAT_OS390
Applicazione OS/390 (stesso valore di MQAT_ZOS).

MQAT_OS400
Applicazione IBM i .

Gestore code MQAT
Gestore code.

UNIX MQAT
Applicazione UNIX .

VOS MQAT
Applicazione VOS Stratus.

WINDOWS MQAT
Applicazione Windows a 16 bit.

MQAT_WINDOWS_NT
Applicazione Windows a 32 bit.

WLM MQAT
Applicazione z/OS workload manager.

XCF MQAT
XCF.

ZOS MQAT
Applicazione z/OS .

MQAT_PREDEFINITO
Il tipo di applicazione predefinita.

Questo è il tipo di applicazione predefinito per la piattaforma su cui è in esecuzione l'applicazione.

Nota: Il valore di questa costante è specifico dell'ambiente. Per questo motivo, compilare sempre l'applicazione utilizzando i file di intestazione, di inclusione o COPY appropriati per la piattaforma su cui verrà eseguita l'applicazione.

MQAT_SCONOSCIUTO
Utilizzare questo valore per indicare che il tipo di applicazione è sconosciuto, anche se sono presenti altre informazioni di contesto.

MQAT_USER_FIRST
Il valore più basso per il tipo di applicazione definito dall'utente.

MQAT_USER_LAST
Il valore più alto per il tipo di applicazione definito dall'utente.

Può verificarsi anche il seguente valore speciale:

MQAT_NO_CONTEXT
Questo valore viene impostato dal gestore code quando un messaggio viene inserito senza contesto (ovvero, viene specificata l'opzione di contesto MQPMO_NO_CONTEXT).

Quando viene richiamato un messaggio, *PutApplType* può essere verificato per questo valore per stabilire se il messaggio ha un contesto (si raccomanda che *PutApplType* non sia mai impostato su MQAT_NO_CONTEXT, da un'applicazione che utilizza MQPMO_SET_ALL_CONTEXT, se uno degli altri campi di contesto non è vuoto).

Quando il gestore code genera queste informazioni come risultato di un inserimento di un'applicazione, il campo viene impostato su un valore determinato dall'ambiente. Su IBM i, è impostato su MQAT_OS400; ; il gestore code non utilizza MQAT_CICS su IBM i.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se MQPMO_SET_ALL_CONTEXT è specificato nel parametro **PutMsgOpts** . Se MQPMO_SET_ALL_CONTEXT non viene specificato, questo campo viene ignorato nell'input ed è un campo di sola emissione.

Questo è un campo di output per la chiamata MQGET. Il valore iniziale di questo campo è MQAT_NO_CONTEXT.

Nome PutAppl(MQCHAR28) per MQMD

Questo è il nome dell'applicazione che ha inserito il messaggio e fa parte del *contesto di origine* del messaggio. Il contenuto differisce tra le piattaforme e potrebbe differire tra le release.

Per ulteriori informazioni sul contesto del messaggio, consultare “MQMD - Descrittore messaggi” a pagina 431 e Contesto del messaggio.

È possibile specificare il nome dell'applicazione in ulteriori linguaggi di programmazione. Per ulteriori informazioni, consultare specifica del nome dell'applicazione nei linguaggi di programmazione supportati .

Il formato di *PutApplName* dipende dal valore *PutApplType* e può cambiare da una release all'altra. Le modifiche sono rare, ma si verificano se l'ambiente cambia.

Quando il gestore code imposta questo campo (ossia, per tutte le opzioni tranne MQPMO_SET_ALL_CONTEXT), imposta il campo su un valore determinato dall'ambiente:

- z/OS Su z/OS, il gestore code utilizza:
  - Per il batch z/OS , il nome lavoro di 8 caratteri dalla scheda JES JOB
  - Per TSO, l'identificativo utente TSO di 7 caratteri
  - Per CICS, l'applid di 8 caratteri, seguito dal tranid di 4 caratteri
  - Per IMS, l'identificativo di sistema IMS di 8 caratteri, seguito dal nome PSB di 8 caratteri
  - Per XCF, il nome del gruppo XCF di 8 caratteri, seguito dal nome del membro XCF di 16 caratteri
  - Per un messaggio generato da un gestore code, i primi 28 caratteri del nome del gestore code
  - Per l'accodamento distribuito senza CICS, il nome lavoro di 8 caratteri dell'iniziatore di canali, seguito dal nome di 8 caratteri del modulo che viene inserito nella coda di messaggi non recapitabili, seguito dall'identificativo di un'attività di 8 caratteri.

  Il nome o i nomi vengono riempiti a destra con spazi vuoti, come qualsiasi spazio nel resto del campo. Se è presente più di un nome, non vi è alcun separatore tra di essi.
- Windows Sui sistemi Windows , il gestore code utilizza i seguenti nomi:
  - Per un'applicazione CICS , il nome della transazione CICS
  - Per un'applicazione nonCICS , i 28 caratteri più a destra del nome completo dell'eseguibile
- IBM i Su IBM i, il gestore code utilizza il nome lavoro completo.
- Linux AIX Su AIX and Linux, il gestore code utilizza i seguenti nomi:
  - Per un'applicazione CICS , il nome della transazione CICS
  - Per un'applicazione nonCICS , MQ richiede al sistema operativo il nome del processo. Viene restituito come nome del file di programma, senza percorso completo. Quindi, MQ inserisce questo nome processo in MQMD MQMD.PutApplName come segue:

    AIX **AIX**

    Se il nome è inferiore o uguale a 28 byte, il nome viene inserito, riempito a destra con spazi.

    Se il nome è maggiore di 28 byte, vengono inseriti i 28 byte più a sinistra del nome.

Linux

Se il nome è inferiore o uguale a 15 byte, il nome viene inserito, riempito a destra con spazi.

Se il nome è maggiore di 15 byte, vengono inseriti i 15 byte più a sinistra del nome, riempiti a destra con spazi.

Ad esempio, se si esegue `/opt/mqm/samp/bin/amqsput QNAME QMNAME`, il nome PutApplè `'amqsput'`. Ci sono 21 caratteri spazio di riempimento in questo campo MQCHAR28 . Notare che il percorso completo che include `/opt/mqm/samp/bin` non è incluso nel nome PutAppl.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se MQPMO_SET_ALL_CONTEXT è specificato nel parametro **PutMsgOpts** . Tutte le informazioni che seguono un carattere null all'interno del campo vengono scartate. Il carattere null e i seguenti caratteri vengono convertiti in spazi dal gestore code. Se MQPMO_SET_ALL_CONTEXT non viene specificato, questo campo viene ignorato nell'input ed è un campo di sola emissione.

PutDate (MQCHAR8) per MQMD

Questa è la data in cui è stato inserito il messaggio e fa parte di **contesto origine** del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare “MQMD - Descrittore messaggi” a pagina 431 e Contesto del messaggio.

Il formato utilizzato per la data in cui questo campo viene generato dal gestore code è:

- AAAAMMGG

dove i caratteri rappresentano:

AAAA
anno (quattro cifre)

MI
mese dell'anno (da 01 a 12)

GG
giorno del mese (da 01 a 31)

GMT (Greenwich Mean Time) viene utilizzato per i campi *PutDate* e *PutTime* , in base all'orologio di sistema impostato in modo accurato su GMT.

Se il messaggio è stato inserito come parte di un'unità di lavoro, la data è quella in cui è stato immesso il messaggio e non la data in cui è stato eseguito il commit dell'unità di lavoro.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se MQPMO_SET_ALL_CONTEXT è specificato nel parametro **PutMsgOpts** . Il contenuto del campo non viene controllato dal gestore code, ad eccezione del fatto che tutte le informazioni che seguono un carattere null all'interno del campo vengono eliminate. Il gestore code converte il carattere null e i seguenti caratteri in spazi vuoti. Se MQPMO_SET_ALL_CONTEXT non viene specificato, questo campo viene ignorato nell'input ed è un campo di sola emissione.

Questo è un campo di output per la chiamata MQGET. La lunghezza di questo campo è fornita da MQ_PUT_DATE_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 8 caratteri vuoti in altri linguaggi di programmazione.

PutTime (MQCHAR8) per MQMD

Questa è l'ora in cui è stato inserito il messaggio e fa parte di **contesto origine** del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare “MQMD - Descrittore messaggi” a pagina 431 e Contesto del messaggio.

Il formato utilizzato per l'ora in cui questo campo viene generato dal gestore code è:

- HHMMSSTH

dove i caratteri rappresentano (in ordine):

OO
ore (da 00 a 23)

MI
minuti (da 00 a 59)

SS
secondi (da 00 a 59; vedere nota)

T
decimi di secondo (da 0 a 9)

H
centesimi di secondo (da 0 a 9)

Nota: Se l'orologio di sistema è sincronizzato con uno standard di tempo molto accurato, è possibile in rare occasioni che vengano restituiti 60 o 61 per i secondi in *PutTime*. Ciò si verifica quando i secondi bisestili vengono inseriti nello standard temporale globale.

GMT (Greenwich Mean Time) viene utilizzato per i campi *PutDate* e *PutTime* , in base all'orologio di sistema impostato in modo accurato su GMT.

Se il messaggio è stato inserito come parte di un'unità di lavoro, l'ora è quella in cui è stato immesso il messaggio e non quella in cui è stato eseguito il commit dell'unità di lavoro.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se MQPMO_SET_ALL_CONTEXT è specificato nel parametro **PutMsgOpts** . Il gestore code non controlla il contenuto del campo, ad eccezione del fatto che tutte le informazioni che seguono un carattere null all'interno del campo vengono eliminate. Il gestore code converte il carattere null e i seguenti caratteri in spazi vuoti. Se MQPMO_SET_ALL_CONTEXT non viene specificato, questo campo viene ignorato nell'input ed è un campo di sola emissione.

Questo è un campo di output per la chiamata MQGET. La lunghezza di questo campo è fornita da MQ_PUT_TIME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 8 caratteri vuoti in altri linguaggi di programmazione.

Dati ApplOrigin(MQCHAR4) per MQMD

Questo fa parte del *contesto di origine* del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare “MQMD - Descrittore messaggi” a pagina 431 e Contesto del messaggio.

`ApplOriginData` è un'informazione definita dalla suite di applicazioni che può essere utilizzata per fornire ulteriori informazioni sull'origine del messaggio. Ad esempio, potrebbe essere impostato dalle applicazioni in esecuzione con l'autorizzazione utente appropriata per indicare se i dati di identità sono attendibili.

Il gestore code considera queste informazioni come dati carattere, ma non ne definisce il formato. Quando il gestore code genera queste informazioni, è completamente vuoto.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se MQPMO_SET_ALL_CONTEXT è specificato nel parametro **PutMsgOpts** . Tutte le informazioni che seguono un carattere null all'interno del campo vengono scartate. Il gestore code converte il carattere null e i seguenti caratteri in spazi vuoti. Se MQPMO_SET_ALL_CONTEXT non viene specificato, questo campo viene ignorato nell'input ed è un campo di sola emissione.

Questo è un campo di output per la chiamata MQGET. La lunghezza di questo campo è fornita da MQ_APPL_ORIGIN_DATA_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 4 caratteri vuoti in altri linguaggi di programmazione.

Quando il messaggio viene pubblicato, anche se `ApplOriginData` è impostato, è vuoto nella sottoscrizione che riceve.

GroupId (MQBYTE24) per MQMD

Si tratta di una stringa di byte utilizzata per identificare il particolare gruppo di messaggi o messaggio logico a cui appartiene il messaggio fisico. *GroupId* viene utilizzato anche se la segmentazione è

consentita per il messaggio. In tutti questi casi, *GroupId* ha un valore non null e uno o più dei seguenti indicatori è impostato nel campo *MsgFlags* :

- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- ISCRIZIONE MQMF_SE
- MQMF_LAST_SEGMENT
- MQMF_SEGMENTAZIONE_CONSENTITA

Se nessuno di questi indicatori è impostato, *GroupId* ha il valore null speciale MQGI_NONE.

L'applicazione non ha bisogno di impostare questo campo sulla chiamata MQPUT o MQGET se:

- Nella chiamata MQPUT, viene specificato MQPMO_LOGICAL_ORDER.
- Nella chiamata MQGET, MQMO_MATCH_GROUP_ID non è specificato.

Questi sono i modi consigliati per utilizzare queste chiamate per i messaggi che non sono messaggi di report. Tuttavia, se l'applicazione richiede un maggiore controllo o la chiamata è MQPUT1, l'applicazione deve verificare che *GroupId* sia impostata su un valore appropriato.

I gruppi di messaggi e i segmenti possono essere elaborati correttamente solo se l'identificativo del gruppo è univoco. Per questo motivo, le applicazioni *non devono generare i propri identificatori di gruppo* ; invece, le applicazioni devono effettuare una delle seguenti operazioni:

- Se viene specificato MQPMO_LOGICAL_ORDER, il gestore code genera automaticamente un identificativo di gruppo univoco per il primo messaggio nel gruppo o segmento del messaggio logico e utilizza tale identificativo di gruppo per i restanti messaggi nel gruppo o segmenti del messaggio logico, in modo che l'applicazione non debba eseguire alcuna azione speciale. Questa è la procedura consigliata.
- Se MQPMO_LOGICAL_ORDER non viene specificato, l'applicazione deve richiedere al gestore code di generare l'identificativo del gruppo, impostando *GroupId* su MQGI_NONE sulla prima chiamata MQPUT o MQPUT1 per un messaggio nel gruppo o segmento del messaggio logico. L'identificativo del gruppo restituito dal gestore code all'output di tale chiamata deve essere quindi utilizzato per i restanti messaggi nel gruppo o nei segmenti del messaggio logico. Se un gruppo di messaggi contiene messaggi segmentati, è necessario utilizzare lo stesso identificativo di gruppo per tutti i segmenti e i messaggi nel gruppo.

  Quando MQPMO_LOGICAL_ORDER non è specificato, i messaggi in gruppi e segmenti di messaggi logici possono essere inseriti in qualsiasi ordine (ad esempio, in ordine inverso), ma l'identificativo del gruppo deve essere assegnato dalla *prima* chiamata MQPUT o MQPUT1 emessa per uno qualsiasi di tali messaggi.

All'input delle chiamate MQPUT e MQPUT1 , il gestore code utilizza il valore descritto in Ordine fisico su una coda. Nell'output delle chiamate MQPUT e MQPUT1 , il gestore code imposta questo campo sul valore che è stato inviato con il messaggio se l'oggetto aperto è una coda singola e non un elenco di distribuzione, ma lo lascia invariato se l'oggetto aperto è un elenco di distribuzione. In quest' ultimo caso, se l'applicazione deve conoscere gli identificativi di gruppo generati, l'applicazione deve fornire i record MQPMR contenenti il campo *GroupId* .

All'input della chiamata MQGET, il gestore code utilizza il valore descritto in Tabella 495 a pagina 403. Nell'output della chiamata MQGET, il gestore code imposta questo campo sul valore per il messaggio richiamato.

Viene definito il seguente valore speciale:

MQGI_NONE
Nessun identificativo di gruppo specificato.

Il valore è zero binario per la lunghezza del campo. Questo è il valore utilizzato per i messaggi che non sono in gruppi, non in segmenti di messaggi logici e per cui la segmentazione non è consentita.

Per il linguaggio di programmazione C, viene definita anche la costante MQGI_NONE_ARRAY, che ha lo stesso valore di MQGI_NONE, ma è un array di caratteri invece di una stringa.

La lunghezza di questo campo è fornita da MQ_GROUP_ID_LENGTH. Il valore iniziale di questo campo è MQGI_NONE. Questo campo viene ignorato se *Version* è minore di MQMD_VERSION_2.

Numero MsgSeq(MQLONG) per MQMD

È il numero di sequenza di un messaggio logico all'interno di un gruppo.

I numeri della sequenza cominciano con il valore 1 e aumentano di 1 per ogni nuovo messaggio logico nel gruppo, fino a un valore massimo pari a 999 999 999. Un messaggio fisico che non fa parte di un gruppo ha un numero di sequenza pari a 1.

L'applicazione non deve impostare questo campo nella chiamata MQPUT o MQGET se:

- Nella chiamata MQPUT, viene specificato MQPMO_LOGICAL_ORDER.
- Nella chiamata MQGET, MQMO_MATCH_MSG_SEQ_NUMBER non viene specificato.

Questi sono i modi consigliati per utilizzare queste chiamate per i messaggi che non sono messaggi di report. Tuttavia, se l'applicazione richiede un maggiore controllo o la chiamata è MQPUT1, l'applicazione deve verificare che *MsgSeqNumber* sia impostata su un valore appropriato.

All'input delle chiamate MQPUT e MQPUT1 , il gestore code utilizza il valore descritto in Ordine fisico su una coda. All'output delle chiamate MQPUT e MQPUT1 , il gestore code imposta questo campo sul valore che è stato inviato con il messaggio.

In fase di input per la chiamata MQGET, il gestore code utilizza il valore mostrato nella Tabella 495 a pagina 403. Nell'output della chiamata MQGET, il gestore code imposta questo campo sul valore per il messaggio richiamato.

Il valore iniziale di questo campo è uno. Questo campo viene ignorato se *Version* è minore di MQMD_VERSION_2.

Offset (MQLONG) per MQMD

Questo è l'offset in byte dei dati nel messaggio fisico dall'inizio del messaggio logico di cui fanno parte i dati. Questi dati sono denominati *segmento*. L'offset è compreso tra 0 e 999 999 999. Un messaggio fisico che non sia un segmento di un messaggio logico ha uno scostamento di zero.

L'applicazione non ha bisogno di impostare questo campo sulla chiamata MQPUT o MQGET se:

- Nella chiamata MQPUT, viene specificato MQPMO_LOGICAL_ORDER.
- Nella chiamata MQGET, MQMO_MATCH_OFFSET non viene specificata.

Questi sono i modi consigliati per utilizzare queste chiamate per i messaggi che non sono messaggi di report. Tuttavia, se l'applicazione non è conforme a queste condizioni o se la chiamata è MQPUT1, l'applicazione deve garantire che *Offset* sia impostato su un valore appropriato.

All'input delle chiamate MQPUT e MQPUT1 , il gestore code utilizza il valore descritto in Ordine fisico su una coda. All'output delle chiamate MQPUT e MQPUT1 , il gestore code imposta questo campo sul valore che è stato inviato con il messaggio.

Per un messaggio di report che riporta un segmento di un messaggio logico, il campo *OriginalLength* (purché non MQOL_UNDEFINED) viene utilizzato per aggiornare l'offset nelle informazioni del segmento conservate dal gestore code.

In fase di input per la chiamata MQGET, il gestore code utilizza il valore mostrato nella Tabella 495 a pagina 403. Nell'output della chiamata MQGET, il gestore code imposta questo campo sul valore per il messaggio richiamato.

Il valore iniziale di questo campo è zero. Questo campo viene ignorato se *Version* è minore di MQMD_VERSION_2.

MsgFlags (MQLONG) per MQMD

MsgFlags sono indicatori che specificano gli attributi del messaggio o ne controllano l'elaborazione.

MsgFlags sono suddivisi nelle categorie seguenti:

- Indicatori di segmentazione
- Indicatori di stato

Indicatori di segmentazione: quando un messaggio è troppo grande per una coda, un tentativo di inserire il messaggio nella coda in genere non riesce. La segmentazione è una tecnica con cui il gestore code o l'applicazione suddivide il messaggio in parti più piccole denominate segmenti e colloca ciascun segmento nella coda come un messaggio fisico separato. L'applicazione che richiama il messaggio può richiamare i segmenti uno per uno o richiedere al gestore code di riassemblare i segmenti in un singolo messaggio restituito dalla chiamata MQGET. Quest' ultimo si ottiene specificando l'opzione MQGMO_COMPLETE_MSG sulla chiamata MQGET e fornendo un buffer sufficientemente grande per contenere il messaggio completo. (Consultare “MQGMO - Opzioni Get - message” a pagina 376 per i dettagli dell'opzione MQGMO_COMPLETE_MSG.) Un messaggio può essere segmentato sul gestore code di invio, su un gestore code intermedio o sul gestore code di destinazione.

È possibile specificare una delle seguenti opzioni per controllare la segmentazione di un messaggio:

MQMF_SEGMENTATION_INIBITO
Questa opzione impedisce che il messaggio venga suddiviso in segmenti dal gestore code. Se specificata per un messaggio che è già un segmento, questa opzione impedisce che il segmento venga suddiviso in segmenti più piccoli.

Il valore di questo indicatore è zero binario. Questa è l'opzione predefinita.

MQMF_SEGMENTAZIONE_CONSENTITA
Questa opzione consente al gestore code di suddividere il messaggio in segmenti. Se specificata per un messaggio che è già un segmento, questa opzione consente di suddividere il segmento in segmenti più piccoli. MQMF_SEGMENTATION_ALLOWED può essere impostato senza MQMF_SEGMENT o MQMF_LAST_SEGMENT.

- Su z/OS, il gestore code non supporta la segmentazione dei messaggi. Se un messaggio è troppo grande per la coda, la chiamata MQPUT o MQPUT1 ha esito negativo con codice di errore MQRC_MSG_TOO_BIG_FOR_Q. Tuttavia, è ancora possibile specificare l'opzione MQMF_SEGMENTATION_ALLOWED e consentire la segmentazione del messaggio in un gestore code remoto.

Quando il gestore code segmenta un messaggio, il gestore code attiva l'indicatore MQMF_SEGMENT nella copia di MQMD inviato con ciascun segmento, ma non modifica le impostazioni di questi indicatori nell'MQMD fornito dall'applicazione nella chiamata MQPUT o MQPUT1 . Per l'ultimo segmento nel messaggio logico, il gestore code attiva anche il contrassegno MQMF_LAST_SEGMENT nell'MQMD inviato con il segmento.

Nota: Prestare attenzione durante l'inserimento dei messaggi con MQMF_SEGMENTATION_ALLOWED ma senza MQPMO_LOGICAL_ORDER. Se il messaggio è:

- Non è un segmento
- Non in un gruppo, e
- Non inoltrato,

L'applicazione deve reimpostare il campo *GroupId* su MQGI_NONE prima *di ogni chiamata* MQPUT o MQPUT1 , in modo tale che il gestore code possa generare un identificativo gruppo univoco per ogni messaggio. Se questa operazione non viene eseguita, i messaggi non correlati possono avere lo stesso identificativo del gruppo, il che potrebbe portare a un'elaborazione non corretta in seguito. Consultare le descrizioni del campo *GroupId* e dell'opzione MQPMO_LOGICAL_ORDER per ulteriori informazioni su quando reimpostare il campo *GroupId* .

Il gestore code suddivide i messaggi in segmenti, se necessario, in modo che i segmenti (più eventuali dati di intestazione richiesti) si adattino alla coda. Tuttavia, vi è un limite inferiore per la dimensione di un segmento generato dal gestore code e solo l'ultimo segmento creato da un messaggio può essere inferiore a questo limite (il limite inferiore per la dimensione di un segmento generato dall'applicazione è di un byte). I segmenti generati dal gestore code potrebbero avere una lunghezza diversa. Il gestore code elabora il messaggio nel modo seguente:

- I formati definiti dall'utente sono suddivisi in limiti che sono multipli di 16 byte; il gestore code non genera segmenti inferiori a 16 byte (diversi dall'ultimo segmento).
- I formati incorporati diversi da MQFMT_STRING sono suddivisi in punti appropriati alla natura dei dati presenti. Tuttavia, il gestore code non suddivide mai un messaggio nel mezzo di un'intestazione IBM MQ . Ciò significa che un segmento contenente una singola struttura di intestazione MQ non può essere ulteriormente suddiviso dal gestore code e, di conseguenza, la dimensione minima del segmento possibile per quel messaggio è maggiore di 16 byte.

  Il secondo o successivo segmento generato dal gestore code inizia con uno dei seguenti:

  – Una struttura di intestazione MQ
  – L'inizio dei dati del messaggio dell'applicazione
  – Parte del percorso attraverso i dati del messaggio dell'applicazione
- MQFMT_STRING è suddiviso senza tenere conto della natura dei dati presenti (SBCS, DBCS o SBCS/DBCS misti). Quando la stringa è DBCS o SBCS/DBCS misto, ciò potrebbe risultare in segmenti che non possono essere convertiti da una serie di caratteri ad un'altra. Il gestore code non suddivide mai i messaggi MQFMT_STRING in segmenti inferiori a 16 byte (diversi dall'ultimo segmento).
- Il gestore code imposta i campi *Format*, *CodedCharSetId*e *Encoding* nell'MQMD di ciascun segmento per descrivere correttamente i dati presenti all' *inizio* del segmento; il nome del formato è il nome di un formato integrato o il nome di un formato definito dall'utente.
- Il campo *Report* nell'MQMD dei segmenti con *Offset* maggiore di zero viene modificato. Per ogni tipo di report, se l'opzione del report è MQRO_ * _WITH_DATA, ma il segmento non può contenere nessuno dei primi 100 byte di dati dell'utente (ovvero, i dati che seguono le strutture di intestazione IBM MQ che possono essere presenti), l'opzione del report viene modificata in MQRO_ *.

Il gestore code segue le regole di cui sopra, ma altrimenti suddivide i messaggi in modo imprevedibile; non fare ipotesi su dove viene suddiviso un messaggio.

Per i messaggi *persistenti* , il gestore code può eseguire la segmentazione solo in un'unità di lavoro:

- Se la chiamata MQPUT o MQPUT1 è in funzione all'interno di un'unità di lavoro definita dall'utente, viene utilizzata tale unità di lavoro. Se la chiamata non riesce durante il processo di segmentazione, il gestore code rimuove tutti i segmenti che sono stati inseriti nella coda come risultato della chiamata non riuscita. Tuttavia, l'errore non impedisce il corretto commit dell'unità di lavoro.
- Se la chiamata opera al di fuori di un'unità di lavoro definita dall'utente e non esiste alcuna unità di lavoro definita dall'utente, il gestore code crea un'unità di lavoro solo per la durata della chiamata. Se la chiamata ha esito positivo, il gestore code esegue automaticamente il commit dell'unità di lavoro. Se la chiamata ha esito negativo, il gestore code esegue il backout dell'unità di lavoro.
- Se la chiamata opera all'esterno di un'unità di lavoro definita dall'utente, ma esiste un'unità di lavoro definita dall'utente, il gestore code non può eseguire la segmentazione. Se il messaggio non richiede la segmentazione, la chiamata può ancora avere esito positivo. Ma se il messaggio richiede la segmentazione, la chiamata ha esito negativo con il codice di errore MQRC_UOW_NOT_AVAILABLE.

Per i messaggi *non persistenti* , il gestore code non richiede che sia disponibile un'unità di lavoro per eseguire la segmentazione.

Prestare particolare attenzione quando si convertono i dati in messaggi che potrebbero essere segmentati:

- Se l'applicazione ricevente converte i dati nella chiamata MQGET e specifica l'opzione MQGMO_COMPLETE_MSG, l'exit di conversione dati viene passata al messaggio completo per l'exit da convertire e il fatto che il messaggio è stato segmentato è evidente all'exit.
- Se l'applicazione ricevente richiama un segmento alla volta, l'uscita di conversione dati viene richiamata per convertire un segmento alla volta. L'uscita deve quindi convertire i dati in un segmento indipendentemente dai dati in uno qualsiasi degli altri segmenti.

  Se la natura dei dati nel messaggio è tale che la segmentazione arbitraria dei dati sui limiti di 16 byte potrebbe risultare in segmenti che non possono essere convertiti dall'uscita, o il formato è MQFMT_STRING e la serie di caratteri è DBCS o SBCS/DBCS misti, l'applicazione di invio deve

creare e inserire i segmenti, specificando MQMF_SEGMENTATION_INIBITED per eliminare ulteriore segmentazione. In questo modo, l'applicazione mittente può garantire che ogni segmento contenga informazioni sufficienti per consentire all'exit di conversione dati di convertire correttamente il segmento.

- Se viene specificata la conversione del mittente per un MCA (message channel agent) di invio, l'MCA converte solo i messaggi che non sono segmenti di messaggi logici; l'MCA non tenta mai di convertire i messaggi che sono segmenti.

Questo indicatore è un indicatore di input nelle chiamate MQPUT e MQPUT1 e un indicatore di output nella chiamata MQGET. Nell'ultima chiamata, il gestore code ripete anche il valore dell'indicatore al campo *Segmentation* in MQGMO.

Il valore iniziale di questo indicatore è MQMF_SEGMENTATION_INIITED.

Indicatori di stato: sono indicatori che indicano se il messaggio fisico appartiene a un gruppo di messaggi, è un segmento di un messaggio logico, entrambi o nessuno dei due. Uno o più dei seguenti valori possono essere specificati nella chiamata MQPUT o MQPUT1 o restituiti dalla chiamata MQGET:

MQMF_MSG_IN_GROUP
Il messaggio è un membro di un gruppo.

MQMF_LAST_MSG_IN_GROUP
Il messaggio è l'ultimo messaggio logico in un gruppo.

Se questo indicatore è impostato, il gestore code attiva MQMF_MSG_IN_GROUP nella copia di MQMD inviata con il messaggio, ma non modifica le impostazioni di tali indicatori nell'MQMD fornito dall'applicazione nella chiamata MQPUT o MQPUT1 .

È valido per un gruppo composto da un solo messaggio logico. In questo caso, MQMF_LAST_MSG_IN_GROUP è impostato, ma il valore del campo *MsgSeqNumber* è uno.

ISCRIZIONE MQMF_SE
Il messaggio è un segmento di un messaggio logico.

Quando MQMF_SEGMENT viene specificato senza MQMF_LAST_SEGMENT, la lunghezza dei dati del messaggio dell'applicazione nel segmento (*escludendo* le lunghezze di qualsiasi IBM MQ struttura di intestazione che potrebbe essere presente) deve essere almeno uno. Se la lunghezza è zero, la chiamata MQPUT o MQPUT1 ha esito negativo con il codice motivo MQRC_SEGMENT_LENGTH_ZERO.

Su z/OS, questa opzione non viene supportata se il messaggio viene inserito in una coda con un tipo di indice MQIT_GROUP_ID.

MQMF_LAST_SEGMENT
Il messaggio è l'ultimo segmento di un messaggio logico.

Se questo indicatore è impostato, il gestore code attiva MQMF_SEGMENT nella copia di MQMD che viene inviata con il messaggio, ma non modifica le impostazioni di tali indicatori nell'MQMD fornito dall'applicazione nella chiamata MQPUT o MQPUT1 .

Un messaggio logico può essere costituito da un solo segmento. In tal caso, MQMF_LAST_SEGMENT è impostato, ma il valore del campo *Offset* è zero.

Quando viene specificato MQMF_LAST_SEGMENT, la lunghezza dei dati del messaggio dell'applicazione nel segmento (*escludendo* le lunghezze delle strutture di intestazione che potrebbero essere presenti) può essere zero.

Su z/OS, questa opzione non viene supportata se il messaggio viene inserito in una coda con un tipo di indice MQIT_GROUP_ID.

L'applicazione deve assicurarsi che questi indicatori siano impostati correttamente quando si inseriscono i messaggi. Se viene specificato MQPMO_LOGICAL_ORDER o se è stato specificato nella precedente chiamata MQPUT per l'handle della coda, le impostazioni degli indicatori devono essere congruenti con le informazioni sul gruppo e sul segmento conservate dal gestore code per l'handle della coda. Le seguenti condizioni si applicano a *successive* chiamate MQPUT per l'handle della coda quando viene specificato MQPMO_LOGICAL_ORDER:

- Se non è presente alcun gruppo corrente o messaggio logico, tutti questi indicatori (e le relative combinazioni) sono validi.
- Una volta specificato MQMF_MSG_IN_GROUP, è necessario che rimanga attivo fino a quando non viene specificato MQMF_LAST_MSG_IN_GROUP. La chiamata ha esito negativo con codice di errore MQRC_INCOMPLETE_GROUP se questa condizione non viene soddisfatta.
- Una volta specificato MQMF_SEGMENT, deve rimanere attivo fino a quando non viene specificato MQMF_LAST_SEGMENT. La chiamata ha esito negativo con codice motivo MQRC_INCOMPLETE_MSG se questa condizione non viene soddisfatta.
- Una volta specificato MQMF_SEGMENT senza MQMF_MSG_IN_GROUP, MQMF_MSG_IN_GROUP deve rimanere *off* fino a quando non viene specificato MQMF_LAST_SEGMENT. La chiamata ha esito negativo con codice motivo MQRC_INCOMPLETE_MSG se questa condizione non viene soddisfatta.

Ordine fisico in una coda mostra le combinazioni valide degli indicatori e i valori utilizzati per vari campi.

Questi indicatori sono indicatori di input per le chiamate MQPUT e MQPUT1 e indicatori di output per la chiamata MQGET. In quest' ultima chiamata, il gestore code ripete anche i valori degli indicatori ai campi *GroupStatus* e *SegmentStatus* in MQGMO.

Non è possibile utilizzare messaggi raggruppati o segmentati con Pubblicazione / Sottoscrizione.

Indicatori predefiniti: è possibile specificare quanto segue per indicare che il messaggio ha attributi predefiniti:

MQMF_NONE
Nessun indicatore di messaggio (attributi di messaggio predefiniti).

Ciò impedisce la segmentazione e indica che il messaggio non è in un gruppo e non è un segmento di un messaggio logico. MQMF_NONE è definito per la documentazione del programma. Non si intende utilizzare questo indicatore con altri, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Il campo *MsgFlags* è suddiviso in sottocampi; per i dettagli, vedere "Opzioni di report e indicatori di messaggi" a pagina 928.

Il valore iniziale di questo campo è MQMF_NONE. Questo campo viene ignorato se *Version* è minore di MQMD_VERSION_2.

OriginalLength (MQLONG) per MQMD

Questo campo è rilevante solo per i messaggi di report che sono segmenti. Specifica la lunghezza del segmento del messaggio a cui si riferisce il messaggio del report; non specifica la lunghezza del messaggio logico di cui fa parte il segmento o la lunghezza dei dati nel messaggio del report.

Nota: Quando si genera un messaggio di report per un messaggio che è un segmento, il gestore code e l'agent del canale dei messaggi copiano in MQMD per il messaggio di report i campi *GroupId*, *MsgSeqNumber*, *Offset*e *MsgFlags*, dal messaggio originale. Di conseguenza, il messaggio del report è anche un segmento. Le applicazioni che generano messaggi di report devono fare lo stesso e impostare correttamente il campo *OriginalLength* .

Viene definito il seguente valore speciale:

MQOL_NON DEFINITO
Lunghezza originale del messaggio non definita.

OriginalLength è un campo di input nelle chiamate MQPUT e MQPUT1 , ma il valore fornito dall'applicazione viene accettato solo in particolari circostanze:

- Se il messaggio inserito è un segmento ed è anche un messaggio di report, il gestore code accetta il valore specificato. Il valore deve essere:
  - Maggiore di zero se il segmento non è l'ultimo segmento
  - Non inferiore a zero se il segmento è l'ultimo segmento
  - Non inferiore alla lunghezza dei dati presenti nel messaggio

Se queste condizioni non vengono soddisfatte, la chiamata ha esito negativo con il codice motivo MQRC_ORIGINAL_LENGTH_ERROR.

- Se il messaggio inserito è un segmento ma non un messaggio di report, il gestore code ignora il campo e utilizza la lunghezza dei dati del messaggio dell'applicazione.
- In tutti gli altri casi, il gestore code ignora il campo e utilizza il valore MQOL_UNDEFINED.

Questo è un campo di output sulla chiamata MQGET.

Il valore iniziale di questo campo è MQOL_UNDEFINED. Questo campo viene ignorato se *Version* è minore di MQMD_VERSION_2.

MQMDE - Estensione descrittore messaggio

La struttura MQMDE descrive i dati che a volte si verificano prima dei dati del messaggio dell'applicazione. La struttura contiene quei campi MQMD che esistono in MQMD version-2 , ma non in MQMD version-1 .

Disponibilità

Tutti i sistemi IBM MQ , più IBM MQ MQI clients connessi a tali sistemi.

Nome formato

MQFMT_MD_ESTENSIONE

Serie di caratteri e codifica

I dati in MQMDE devono essere nella serie di caratteri e nella codifica del gestore code locale; tali dati sono forniti dall'attributo del gestore code **CodedCharSetId** e MQENC_NATIVE per il linguaggio di programmazione C.

Impostare la serie di caratteri e la codifica di MQMDE nei campi *CodedCharSetId* e *Encoding* in:

- MQMD (se la struttura MQMDE si trova all'inizio dei dati del messaggio) oppure
- La struttura dell'intestazione che precede la struttura MQMDE (tutti gli altri casi).

Se MQMDE non si trova nella serie di caratteri e nella codifica del gestore code, MQMDE viene accettato ma non viene rispettato, vale a dire, MQMDE viene considerato come dati del messaggio.

Nota: Su Windows, le applicazioni compilate con Micro Focus COBOL utilizzano un valore MQENC_NATIVE diverso dalla codifica del gestore code. Sebbene i campi numerici nella struttura MQMD nelle chiamate MQPUT, MQPUT1e MQGET debbano essere nella codifica Micro Focus COBOL, i campi numerici nella struttura MQMDE devono essere nella codifica del gestore code. Questo valore è fornito da MQENC_NATIVE per il linguaggio di programmazione C e ha il valore 546.

Utilizzo

Le applicazioni che utilizzano version-2 MQMD non incontreranno una struttura MQMDE. Tuttavia, le applicazioni specializzate e le applicazioni che continuano ad utilizzare un MQMD version-1 potrebbero rilevare un MQMDE in alcune situazioni. La struttura MQMDE può verificarsi nelle circostanze seguenti:

- Specificato nelle chiamate MQPUT e MQPUT1
- Restituito dalla chiamata MQGET
- Nei messaggi sulle code di trasmissione

MQMDE specificato sulle chiamate MQPUT e MQPUT1

Nelle chiamate MQPUT e MQPUT1 , se l'applicazione fornisce un MQMD version-1 , l'applicazione può facoltativamente aggiungere un prefisso ai dati del messaggio con un MQMDE, impostando il campo *Format* in MQMD su MQFMT_MD_EXTENSION per indicare che è presente un MQMDE. Se l'applicazione non fornisce un MQMDE, il gestore code assume i valori predefiniti per i campi in MQMDE. I valori

predefiniti utilizzati dal gestore code sono uguali ai valori iniziali per la struttura; consultare Tabella 503 a pagina 486.

Se l'applicazione fornisce un prefisso version-2 MQMD *e* ai dati del messaggio dell'applicazione viene anteposto un MQMDE, le strutture vengono elaborate come mostrato nella seguente tabella.

Tabella 502. Azione del gestore code quando MQMDE è specificato su MQPUT o MQPUT1 per MQMDE

Versione MQMD	Valori dei campi version-2	Valori dei corrispondenti campi in MQMDE	Azione eseguita dal gestore code
1	-	Valido	MQMDE è rispettato
2	Valore predefinito	Valido	MQMDE è rispettato
2	Non predefinito	Valido	MQMDE viene trattato come dati del messaggio
1 o 2	Qualsiasi	Non valido	La chiamata ha esito negativo con un codice di errore appropriato
1 o 2	Qualsiasi	MQMDE è nella serie di caratteri o nella codifica non corretta o è una versione non supportata	MQMDE viene trattato come dati del messaggio

Nota: Su z/OS, se l'applicazione specifica un MQMD version-1 con un MQMDE, il gestore code convalida MQMDE solo se la coda ha un *IndexType* di MQIT_GROUP_ID.

C'è un caso speciale. Se l'applicazione utilizza un MQMD version-2 per inserire un messaggio che è un segmento (ovvero, è impostato l'indicatore MQMF_SEGMENT o MQMF_LAST_SEGMENT) e il nome del formato in MQMD è MQFMT_DEAD_LETTER_HEADER, il gestore code genera una struttura MQMDE e la inserisce *tra* la struttura MQDLH e i dati che la seguono. In MQMD che il gestore code conserva con il messaggio, i campi version-2 sono impostati sui rispettivi valori predefiniti.

Molti dei campi presenti in MQMD version-2 ma non in MQMD version-1 sono campi di input / output in MQPUT e MQPUT1. Tuttavia, il gestore code non restituisce alcun valore nei campi equivalenti in MQMDE sull'output dalle chiamate MQPUT e MQPUT1 ; se l'applicazione richiede tali valori di output, deve utilizzare un MQMD version-2 .

MQMDE restituito dalla chiamata MQGET

Nella chiamata MQGET, se l'applicazione fornisce un MQMD version-1 , il gestore code prefissa il messaggio restituito con un MQMDE, ma solo se uno o più campi in MQMDE hanno un valore non predefinito. Il gestore code imposta il campo *Format* in MQMD sul valore MQFMT_MD_EXTENSION per indicare che è presente un MQMDE.

Se l'applicazione fornisce un MQMDE all'inizio del parametro **Buffer** , MQMDE viene ignorato. Al ritorno dalla chiamata MQGET, viene sostituito da MQMDE per il messaggio (se ne è necessario uno) o sovrascritto dai dati del messaggio dell'applicazione (se MQMDE non è necessario).

Se la chiamata MQGET restituisce un MQMDE, i dati in MQMDE si trovano generalmente nella serie di caratteri e nella codifica del gestore code. Tuttavia, MQMDE potrebbe trovarsi in un'altra serie di caratteri e codificare se:

- MQMDE è stato considerato come un dato nella chiamata MQPUT o MQPUT1 (consultare Tabella 502 a pagina 485 per le circostanze che possono causare ciò).
- Il messaggio è stato ricevuto da un gestore code remoto connesso tramite una connessione TCP e l'MCA (message channel agent) di ricezione non è stato configurato correttamente.

Nota: Su Windows, le applicazioni compilate con Micro Focus COBOL utilizzano un valore di MQENC_NATIVE diverso dalla codifica del gestore code (vedere sopra).

MQMDE nei messaggi sulle code di trasmissione

I messaggi sulle code di trasmissione hanno come prefisso la struttura MQXQH, che contiene al suo interno un MQMD version-1 . Potrebbe essere presente anche un MQMDE, posizionato tra la struttura MQXQH e i dati del messaggio dell'applicazione, ma di solito è presente solo se uno o più campi in MQMDE hanno un valore non predefinito.

Altre strutture di intestazione MQ possono verificarsi anche tra la struttura MQXQH e i dati del messaggio dell'applicazione. Ad esempio, quando l'intestazione MQDLH è presente e il messaggio non è un segmento, l'ordine è:

- MQXQH (contenente un MQMD version-1)
- MQMDE
- MQDLH
- dati dei messaggi dell'applicazione

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 503. Campi in MQMDE per MQMDE

Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
StrucId (identificativo della struttura)	ID_STRUC_MQMDE	'MDE¬'
Versione (numero versione struttura)	MQMDE_VERSION_2	2
StrucLength (lunghezza della struttura MQMDE)	MQMDE_LENGTH_2	72
Codifica (codifica numerica dei dati che seguono MQMDE)	MQEN_NATIVE	Dipende dall'ambiente
CodedCharSetId (identificativo della serie di caratteri dei dati che seguono MQMDE)	MQCCSI_UNDEFINED	0
Formato (nome formato dei dati che seguono MQMDE)	MQFMT_NONE	Spazi
Indicatori (indicatori generali)	MQMDEF_NONE	0
GroupId (identificativo gruppo)	MQGI_NONE	Valori null
MsgSeqNumero (numero di sequenza del messaggio logico all'interno del gruppo)	Nessuna	1
Offset (offset dei dati nel messaggio fisico dall'inizio del messaggio logico)	Nessuna	0
MsgFlags (indicatori di messaggio)	MQMF_NONE	0
OriginalLength (lunghezza del messaggio originale)	MQOL_NON DEFINITO	-1

Tabella 503. Campi in MQMDE per MQMDE (Continua)

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macroMQMDE_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura:

```
MQMDE MyMDE = {MQMDE_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQMDE

```
typedef struct tagMQMDE MQMDE;
struct tagMQMDE {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    StrucLength;     /* Length of MQMDE structure */
  MQLONG    Encoding;        /* Numeric encoding of data that follows
                                MQMDE */
  MQLONG    CodedCharSetId;  /* Character-set identifier of data that
                                follows MQMDE */
  MQCHAR8   Format;          /* Format name of data that follows
                                MQMDE */
  MQLONG    Flags;           /* General flags */
  MQBYTE24  GroupId;         /* Group identifier */
  MQLONG    MsgSeqNumber;    /* Sequence number of logical message
                                within group */
  MQLONG    Offset;          /* Offset of data in physical message from
                                start of logical message */
  MQLONG    MsgFlags;        /* Message flags */
  MQLONG    OriginalLength;  /* Length of original message */
};
```

Dichiarazione COBOL per MQMDE

```
**   MQMDE structure
  10 MQMDE.
**    Structure identifier
   15 MQMDE-STRUCID        PIC X(4).
**    Structure version number
   15 MQMDE-VERSION        PIC S9(9) BINARY.
**    Length of MQMDE structure
   15 MQMDE-STRUCLENGTH    PIC S9(9) BINARY.
**    Numeric encoding of data that follows MQMDE
   15 MQMDE-ENCODING       PIC S9(9) BINARY.
**    Character-set identifier of data that follows MQMDE
   15 MQMDE-CODEDCHARSETID PIC S9(9) BINARY.
**    Format name of data that follows MQMDE
   15 MQMDE-FORMAT         PIC X(8).
**    General flags
   15 MQMDE-FLAGS          PIC S9(9) BINARY.
**    Group identifier
   15 MQMDE-GROUPID        PIC X(24).
**    Sequence number of logical message within group
   15 MQMDE-MSGSEQNUMBER   PIC S9(9) BINARY.
**    Offset of data in physical message from start of logical message
   15 MQMDE-OFFSET         PIC S9(9) BINARY.
**    Message flags
   15 MQMDE-MSGFLAGS       PIC S9(9) BINARY.
**    Length of original message
   15 MQMDE-ORIGINALLENGTH PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQMDE

```
dcl
 1 MQMDE based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 StrucLength    fixed bin(31), /* Length of MQMDE structure */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
                                     follows MQMDE */
  3 CodedCharSetId fixed bin(31), /* Character-set identifier of data
                                     that follows MQMDE */
  3 Format         char(8),       /* Format name of data that follows
                                     MQMDE */
  3 Flags          fixed bin(31), /* General flags */
  3 GroupId        char(24),      /* Group identifier */
  3 MsgSeqNumber   fixed bin(31), /* Sequence number of logical message
                                     within group */
  3 Offset         fixed bin(31), /* Offset of data in physical message
                                     from start of logical message */
  3 MsgFlags       fixed bin(31), /* Message flags */
  3 OriginalLength fixed bin(31); /* Length of original message */
```

Dichiarazione High Level Assembler per MQMDE

```
MQMDE                 DSECT
MQMDE_STRUCID         DS   CL4   Structure identifier
MQMDE_VERSION         DS   F     Structure version number
MQMDE_STRUCLENGTH     DS   F     Length of MQMDE structure
MQMDE_ENCODING        DS   F     Numeric encoding of data that follows
*                                MQMDE
MQMDE_CODEDCHARSETID  DS   F     Character-set identifier of data that
*                                follows MQMDE
MQMDE_FORMAT          DS   CL8   Format name of data that follows MQMDE
MQMDE_FLAGS           DS   F     General flags
MQMDE_GROUPID         DS   XL24  Group identifier
MQMDE_MSGSEQNUMBER    DS   F     Sequence number of logical message
*                                within group
MQMDE_OFFSET          DS   F     Offset of data in physical message from
*                                start of logical message
MQMDE_MSGFLAGS        DS   F     Message flags
MQMDE_ORIGINALLENGTH  DS   F     Length of original message
*
MQMDE_LENGTH          EQU  *-MQMDE
                      ORG  MQMDE
MQMDE_AREA            DS   CL(MQMDE_LENGTH)
```

Dichiarazione di Visual Basic per MQMDE

```
Type MQMDE
  StrucId        As String*4 'Structure identifier'
  Version        As Long     'Structure version number'
  StrucLength    As Long     'Length of MQMDE structure'
  Encoding       As Long     'Numeric encoding of data that follows'
                             'MQMDE'
  CodedCharSetId As Long     'Character-set identifier of data that'
                             'follows MQMDE'
  Format         As String*8 'Format name of data that follows MQMDE'
  Flags          As Long     'General flags'
  GroupId        As MQBYTE24 'Group identifier'
  MsgSeqNumber   As Long     'Sequence number of logical message within'
                             'group'
  Offset         As Long     'Offset of data in physical message from'
                             'start of logical message'
  MsgFlags       As Long     'Message flags'
  OriginalLength As Long     'Length of original message'
End Type
```

StrucId (MQCHAR4) per MQMDE

Questo è l'identificativo della struttura dell'estensione del descrittore del messaggio. È sempre un campo di immissione. Il valore è MQMDE_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQMDE
L'identificativo per la struttura di estensione del descrittore del messaggio.

Per il linguaggio di programmazione C, viene definita anche la costante MQMDE_STRUC_ID_ARRAY. Questo ha lo stesso valore di MQMDE_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQMDE

Questo è il numero di versione della struttura; il valore deve essere:

MQMDE_VERSION_2
Struttura di estensione del descrittore messaggi Version-2 .

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQMDE_CURRENT_
La versione corrente della struttura di estensione del descrittore del messaggio.

Il valore iniziale di questo campo è MQMDE_VERSION_2.

StrucLength (MQLONG) per MQMDE

Questa è la lunghezza della struttura MQMDE; è stato definito il seguente valore:

MQMDE_LENGTH_2
Lunghezza della struttura di estensione del descrittore del messaggio version-2 .

Il valore iniziale di questo campo è MQMDE_LENGTH_2.

Codifica (MQLONG) per MQMDE

Specifica la codifica numerica dei dati che seguono la struttura MQMDE; non si applica ai dati numerici nella struttura MQMDE stessa.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Il gestore code non verifica la validità del campo. Consultare il campo *Encoding* descritto in “MQMD - Descrittore messaggi” a pagina 431 per ulteriori informazioni sulle codifiche dei dati.

Il valore iniziale di questo campo è MQENC_NATIVE.

CodedCharSetId (MQLONG) per MQMDE

Specifica l'identificativo della serie di caratteri dei dati che seguono la struttura MQMDE; non si applica ai dati carattere nella struttura MQMDE stessa.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Il gestore code non controlla che questo campo sia valido. È possibile utilizzare il seguente valore speciale:

MQCCSI_INHERIT
I dati carattere nei dati *che seguono* questa struttura si trovano nella stessa serie di caratteri di questa struttura.

Il gestore code modifica questo valore nella struttura inviata nel messaggio nell'effettivo identificativo della serie di caratteri della struttura. Se non si verifica alcun errore, il valore MQCCSI_INHERIT non viene restituito dalla chiamata MQGET.

MQCCSI_INHERIT non può essere utilizzato se il valore del campo *PutApplType* in MQMD è MQAT_BROKER.

Questo valore è supportato nei seguenti ambienti:

- AIX
- IBM i
- Linux

- Windows Windows

e per i client IBM MQ connessi a questi sistemi.

Il valore iniziale di questo campo è MQCCSI_UNDEFINED.

Formato (MQCHAR8) per MQMDE

Specifica il nome del formato dei dati che seguono la struttura MQMDE.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Il gestore code non controlla che questo campo sia valido. Consultare il campo *Format* descritto in"MQMD - Descrittore messaggi" a pagina 431per ulteriori informazioni sui nomi dei formati.

Il valore iniziale di questo campo è MQFMT_NONE.

Indicatori (MQLONG) per MQMDE

È possibile specificare il seguente indicatore:

MQMDEF_NONE
Nessun indicatore.

Il valore iniziale di questo campo è MQMDEF_NONE.

GroupId (MQBYTE24) per MQMDE

Consultare il campo *GroupId* descritto in"MQMD - Descrittore messaggi" a pagina 431. Il valore iniziale di questo campo è MQGI_NONE.

Numero MsgSeq(MQLONG) per MQMDE

Consultare il campo *MsgSeqNumber* descritto in"MQMD - Descrittore messaggi" a pagina 431. Il valore iniziale di questo campo è 1.

Offset (MQLONG) per MQMDE

Consultare il campo *Offset* descritto in"MQMD - Descrittore messaggi" a pagina 431. Il valore iniziale di questo campo è 0.

MsgFlags (MQLONG) per MQMDE

Consultare il campo *MsgFlags* descritto in "MQMD - Descrittore messaggi" a pagina 431. Il valore iniziale di questo campo è MQMF_NONE.

OriginalLength (MQLONG) per MQMDE

Consultare il campo *OriginalLength* descritto in"MQMD - Descrittore messaggi" a pagina 431. Il valore iniziale di questo campo è MQOL_UNDEFINED.

MQMHBO - Gestore messaggi per opzioni buffer

La struttura MQMHBO consente alle applicazioni di specificare le opzioni che controllano la modalità di produzione dei buffer dagli handle dei messaggi. La struttura è un parametro di input nella chiamata MQMHBUF.

Serie di caratteri e codifica

I dati in MQMHBO devono trovarsi nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (MQENC_NATIVE).

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 504. Campi in MQMHBO

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQMHBO_	'MHBO'
Versione (numero versione struttura)	MQMHBO_VERSION_1	1
Opzioni (opzioni che controllano l'azione di MQMHBUF)	MQMHBO_PROPERTIES_IN_MQRFH2	

Note:

1. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.
2. Nel linguaggio di programmazione C, la variabile macroMQMHBO_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQMHBO MyMHBO = {MQMHBO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQMHBO

```
typedef struct tagMQMHBO MQMHBO;
struct tagMQMHBO {
  MQCHAR4  StrucId;        /* Structure identifier */
  MQLONG   Version;        /* Structure version number */
  MQLONG   Options;        /* Options that control the action of
                              MQMHBUF */
};
```

Dichiarazione COBOL per MQMHBO

```
**  MQMHBO structure
  10 MQMHBO.
**    Structure identifier
   15 MQMHBO-STRUCID               PIC X(4).
**    Structure version number
   15 MQMHBO-VERSION               PIC S9(9) BINARY.
**    Options that control the action of MQMHBUF
   15 MQMHBO-OPTIONS               PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQMHBO

```
Dcl
 1 MQMHBO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action
                                     of MQMHBUF */
```

Dichiarazione High Level Assembler per MQMHBO

```
MQMHBO                    DSECT
MQMHBO_STRUCID            DS   CL4  Structure identifier
MQMHBO_VERSION            DS   F    Structure version number
```

```
MQMHBO_OPTIONS              DS   F    Options that control the
*                                     action of MQMHBUF
MQMHBO_LENGTH               EQU  *-MQMHBO
MQMHBO_AREA                 DS   CL(MQMHBO_LENGTH)
```

StrucId (MQCHAR4) per MQMHBO

Questo è l'identificativo della struttura dell'handle del messaggio per la struttura delle opzioni del buffer. È sempre un campo di immissione. Il valore è MQMHBO_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQMHBO_
Identificativo per l'handle del messaggio nella struttura delle opzioni del buffer.

Per il linguaggio di programmazione C, viene definita anche la costante MQMHBO_STRUC_ID_ARRAY. Ha lo stesso valore di MQMHBO_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQMHBO

Handle del messaggio per la struttura delle opzioni di buffer - campo Versione

Questo è il numero di versione della struttura. Il valore deve essere:

MQMHBO_VERSION_1
Numero di versione per l'handle del messaggio nella struttura delle opzioni del buffer.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQMHBO_CURRENT_
La versione corrente dell'handle del messaggio nella struttura delle opzioni del buffer.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQMHBO_VERSION_1.

Opzioni (MQLONG) per MQMHBO

Handle del messaggio per la struttura delle opzioni del buffer - Campo Opzioni

Queste opzioni controllano l'azione di MQMHBUF.

È possibile specificare la seguente opzione:

MQMHBO_PROPERTIES_IN_MQRFH2
Durante la conversione delle proprietà da un handle del messaggio in un buffer, convertirle nel formato MQRFH2 .

Facoltativamente, è anche possibile specificare la seguente opzione. Per specificare più di un'opzione, aggiungere i valori insieme (non aggiungere la stessa costante più di una volta) o combinare i valori utilizzando l'operazione OR bit per bit (se il linguaggio di programmazione supporta le operazioni bit).

MQMHBO_DELETE_PROPERTIES
Le proprietà che vengono aggiunte al buffer vengono eliminate dall'handle del messaggio. Se la chiamata ha esito negativo, non viene eliminata alcuna proprietà.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQMHBO_PROPERTIES_IN_MQRFH2.

MQOD - Descrittore oggetto

La struttura MQOD viene utilizzata per specificare un oggetto in base al nome. La struttura è un parametro di input / output sulle chiamate MQOPEN e MQPUT1 .
Sono validi i seguenti tipi di oggetto:

- Coda o elenco di distribuzione
- Elenco nomi
- Definizione di processo
- Gestore code
- Argomento

Disponibilità

Tutti i sistemi IBM MQ , più IBM MQ MQI clients connessi a questi sistemi.

Versione

La versione corrente di MQOD è MQOD_VERSION_4. Le applicazioni che si desidera trasferire tra diversi ambienti devono garantire che la versione richiesta di MQOD sia supportata in tutti gli ambienti interessati. I campi che esistono solo nelle versioni più recenti della struttura sono identificati come tali nelle descrizioni che seguono.

I file di intestazione, COPY e INCLUDE forniti per i linguaggi di programmazione supportati contengono la versione più recente di MQOD supportata dall'ambiente, ma con il valore iniziale del campo *Version* impostato su MQOD_VERSION_1. Per utilizzare i campi non presenti nella struttura version-1 , l'applicazione deve impostare il campo *Version* sul numero di versione della versione richiesta.

Per aprire un elenco di distribuzione, *Version* deve essere MQOD_VERSION_2 o superiore.

Serie di caratteri e codifica

I dati in MQOD devono essere nella serie di caratteri fornita dall'attributo gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da MQENC_NATIVE. Tuttavia, se l'applicazione è in esecuzione come client MQ MQI, la struttura deve essere nella serie di caratteri e nella codifica del client.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQOD	'OD¬¬'
Versione (numero versione struttura)	MQOD_VERSION_1	1
ObjectType (tipo oggetto)	MQOT_Q	1
ObjectName (nome oggetto)	Nessuna	Stringa null o spazi vuoti
ObjectQMgrNome (nome gestore code oggetto)	Nessuna	Stringa null o spazi vuoti
DynamicQName (nome coda dinamica)	Nessuna	'CSQ.*' su z/OS ; 'AMQ.*' altrimenti
AlternateUserId (identificativo utente alternativo)	Nessuna	Stringa null o spazi vuoti
Nota: I restanti campi vengono ignorati se *Version* è minore di MQOD_VERSION_2.		
RecsPresent (numero di record oggetto presenti)	Nessuna	0
KnownDestKnownDest (numero di code locali aperte correttamente)	Nessuna	0
UnknownDestUnknownDest (numero di code remote aperte correttamente)	Nessuna	0
InvalidDestInvalidDest (il numero di code che non è stato possibile aprire)	Nessuna	0
ObjectRecOffset (offset del primo record di oggetto dall'avvio di MQOD)	Nessuna	0

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
ResponseRecResponseRec (offset del primo record di risposta dall'inizio di MQOD)	Nessuna	0
ObjectRecPtr (indirizzo del primo record oggetto)	Nessuna	Puntatore null o byte null
ResponseRecPtr (indirizzo del primo record di risposta)	Nessuna	Puntatore null o byte null
Nota: I restanti campi vengono ignorati se *Version* è minore di MQOD_VERSION_3.		
AlternateSecurityId (identificativo di sicurezza alternativo)	MQSID_NONE	Valori null
ResolvedQName (nome coda risolto)	Nessuna	Stringa null o spazi vuoti
ResolvedQMgrNome (nome gestore code risolto)	Nessuna	Stringa null o spazi vuoti
Nota: I restanti campi vengono ignorati se *Version* è minore di MQOD_VERSION_4.		
ObjectString (nome oggetto lungo)	MQCHARV_PREDEFINITO	Come definito per MQCHARV
SelectionString (stringa di selezione)	MQCHARV_PREDEFINITO	Come definito per MQCHARV
ResObjectString (nome oggetto esteso risolto)	MQCHARV_PREDEFINITO	Come definito per MQCHARV
ResolvedType (tipo di oggetto risolto)	MQOT_NONE	0

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.
3. Nel linguaggio di programmazione C, la variabile macroMQOD_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura:

```
MQOD MyOD = {MQOD_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQDO

```
typedef struct tagMQOD MQOD;
struct tagMQOD {
  MQCHAR4    StrucId;              /* Structure identifier */
  MQLONG     Version;              /* Structure version number */
  MQLONG     ObjectType;           /* Object type */
  MQCHAR48   ObjectName;           /* Object name */
  MQCHAR48   ObjectQMgrName;       /* Object queue manager name */
  MQCHAR48   DynamicQName;         /* Dynamic queue name */
  MQCHAR12   AlternateUserId;      /* Alternate user identifier */
  /* Ver:1 */
  MQLONG     RecsPresent;          /* Number of object records present */
  MQLONG     KnownDestCount;       /* Number of local queues opened
                                      successfully */
```

```
  MQLONG     UnknownDestCount;     /* Number of remote queues opened
                                      successfully */
  MQLONG     InvalidDestCount;     /* Number of queues that failed to
                                      open */
  MQLONG     ObjectRecOffset;      /* Offset of first object record from
                                      start of MQOD */
  MQLONG     ResponseRecOffset;    /* Offset of first response record
                                      from start of MQOD */
  MQPTR      ObjectRecPtr;         /* Address of first object record */
  MQPTR      ResponseRecPtr;       /* Address of first response record */
  /* Ver:2 */
  MQBYTE40   AlternateSecurityId;  /* Alternate security identifier */
  MQCHAR48   ResolvedQName;        /* Resolved queue name */
  MQCHAR48   ResolvedQMgrName;     /* Resolved queue manager name */
  /* Ver:3 */
  MQCHARV    ObjectString;         /* Object Long name */
  MQCHARV    SelectionString;      /* Message Selector */
  MQCHARV    ResObjectString;      /* Resolved Long object name*/
  MQLONG     ResolvedType          /* Alias queue resolved
                                      oject type */
  /* Ver:4 */
};
```

Dichiarazione COBOL per MQOD

```
** MQOD structure
 10  MQOD.
** Structure identifier
 15  MQOD-STRUCID                    PIC X(4).
** Structure version number
 15  MQOD-VERSION                    PIC S9(9) BINARY.
** Object type
 15  MQOD-OBJECTTYPE                 PIC S9(9) BINARY.
** Object name
 15  MQOD-OBJECTNAME                 PIC X(48).
** Object queue manager name
 15  MQOD-OBJECTQMGRNAME             PIC X(48).
** Dynamic queue name
 15  MQOD-DYNAMICQNAME               PIC X(48).
** Alternate user identifier
 15  MQOD-ALTERNATEUSERID            PIC X(12).
** Number of object records present
 15  MQOD-RECSPRESENT                PIC S9(9) BINARY.
** Number of local queues opened successfully
 15  MQOD-KNOWNDESTCOUNT             PIC S9(9) BINARY.
** Number of remote queues opened successfully
 15  MQOD-UNKNOWNDESTCOUNT           PIC S9(9) BINARY.
** Number of queues that failed to open
 15  MQOD-INVALIDDESTCOUNT           PIC S9(9) BINARY.
** Offset of first object record from start of MQOD
 15  MQOD-OBJECTRECOFFSET            PIC S9(9) BINARY.
** Offset of first response record from start of MQOD
 15  MQOD-RESPONSERECOFFSET          PIC S9(9) BINARY.
** Address of first object record
 15  MQOD-OBJECTRECPTR               POINTER.
** Address of first response record
 15  MQOD-RESPONSERECPTR             POINTER.
** Alternate security identifier
 15  MQOD-ALTERNATESECURITYID        PIC X(40).
** Resolved queue name
 15  MQOD-RESOLVEDQNAME              PIC X(48).
** Resolved queue manager name
 15  MQOD-RESOLVEDQMGRNAME           PIC X(48).
** Object Long name
 15 MQOD-OBJECTSTRING.
** Address of variable length string
    20  MQOD-OBJECTSTRING-VSPTR      POINTER.
** Offset of variable length string
    20  MQOD-OBJECTSTRING-VSOFFSET   PIC S9(9) BINARY.
** size of buffer
    20  MQOD-OBJECTSTRING-VSBUFSIZE  PIC S9(9) BINARY.
** Length of variable length string
    20  MQOD-OBJECTSTRING-VSLENGTH   PIC S9(9) BINARY.
** CCSID of variable length string
    20  MQOD-OBJECTSTRING-VSCCSID    PIC S9(9) BINARY.
** Message Selector
 15 MQOD-SELECTIONSTRING.
** Address of variable length string
    20  MQOD-SELECTIONSTRING-VSPTR   POINTER.
```

```
** Offset of variable length string
   20  MQOD-SELECTIONSTRING-VSOFFSET  PIC S9(9) BINARY.
** size of buffer
   20  MQOD-SELECTIONSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
   20  MQOD-SELECTIONSTRING-VSLENGTH  PIC S9(9) BINARY.
** CCSID of variable length string
   20  MQOD-SELECTIONSTRING-VSCCSID   PIC S9(9) BINARY.
** Resolved Long object name
 15 MQOD-RESOBJECTSTRING.
** Address of variable length string
   20  MQOD-RESOBJECTSTRING-VSPTR     POINTER.
** Offset of variable length string
   20  MQOD-RESOBJECTSTRING-VSOFFSET  PIC S9(9) BINARY.
** size of buffer
   20  MQOD-RESOBJECTSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
   20  MQOD-RESOBJECTSTRING-VSLENGTH  PIC S9(9) BINARY.
** CCSID of variable length string
   20  MQOD-RESOBJECTSTRING-VSCCSID   PIC S9(9) BINARY.
** Alias queue resolved object type
   15  MQOD-RESOLVEDTYPE              PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQOD

```
dcl
 1 MQOD based,
  3 StrucId             char(4),       /* Structure identifier */
  3 Version             fixed bin(31), /* Structure version number */
  3 ObjectType          fixed bin(31), /* Object type */
  3 ObjectName          char(48),      /* Object name */
  3 ObjectQMgrName      char(48),      /* Object queue manager name */
  3 DynamicQName        char(48),      /* Dynamic queue name */
  3 AlternateUserId     char(12),      /* Alternate user identifier */
  3 RecsPresent         fixed bin(31), /* Number of object records
                                          present */
  3 KnownDestCount      fixed bin(31), /* Number of local queues opened
                                          successfully */
  3 UnknownDestCount    fixed bin(31), /* Number of remote queues opened
                                          successfully */
  3 InvalidDestCount    fixed bin(31), /* Number of queues that failed to
                                          open */
  3 ObjectRecOffset     fixed bin(31), /* Offset of first object record
                                          from start of MQOD */
  3 ResponseRecOffset   fixed bin(31), /* Offset of first response record
                                          from start of MQOD */
  3 ObjectRecPtr        pointer,       /* Address of first object record */
  3 ResponseRecPtr      pointer,       /* Address of first response
                                          record */
  3 AlternateSecurityId char(40),      /* Alternate security identifier */
  3 ResolvedQName       char(48),      /* Resolved queue name */
  3 ResolvedQMgrName    char(48),      /* Resolved queue manager name */
  3 ObjectString,                      /* Object Long name */
   5 VSPtr              pointer,       /* Address of variable length string */
   5 VSOffset           fixed bin(31), /* Offset of variable length string */
   5 VSBufSize          fixed bin(31), /* size of buffer */
   5 VSLength           fixed bin(31), /* Length of variable length string */
   5 VSCCSID            fixed bin(31), /* CCSID of variable length string */
  3 SelectionString,                   /* Message Selection */
   5 VSPtr              pointer,       /* Address of variable length string */
   5 VSOffset           fixed bin(31), /* Offset of variable length string */
   5 VSBufSize          fixed bin(31), /* size of buffer */
   5 VSLength           fixed bin(31), /* Length of variable length string */
   5 VSCCSID            fixed bin(31), /* CCSID of variable length string */
  3 ResObjectString,                   /* Resolved Long object name */
   5 VSPtr              pointer,       /* Address of variable length string */
   5 VSOffset           fixed bin(31), /* Offset of variable length string */
   5 VSBufSize          fixed bin(31), /* size of buffer */
   5 VSLength           fixed bin(31), /* Length of variable length string */
   5 VSCCSID            fixed bin(31), /* CCSID of variable length string */
  3 ResolvedType        fixed bin(31); /* Alias queue resolved object type */
```

Dichiarazione High Level Assembler per MQOD

```
MQOD                          DSECT
MQOD_STRUCID                  DS   CL4   Structure identifier
MQOD_VERSION                  DS   F     Structure version number
```

```
MQOD_OBJECTTYPE                DS   F     Object type
MQOD_OBJECTNAME                DS   CL48  Object name
MQOD_OBJECTQMGRNAME            DS   CL48  Object queue manager name
MQOD_DYNAMICQNAME              DS   CL48  Dynamic queue name
MQOD_ALTERNATEUSERID           DS   CL12  Alternate user identifier
MQOD_RECSPRESENT               DS   F     Number of object records present
MQOD_KNOWNDESTCOUNT            DS   F     Number of local queues opened
*                                         successfully
MQOD_UNKNOWNDESTCOUNT          DS   F     Number of remote queues opened
*                                         successfully
MQOD_INVALIDDESTCOUNT          DS   F     Number of queues that failed to
*                                         open
MQOD_OBJECTRECOFFSET           DS   F     Offset of first object record from
*                                         start of MQOD
MQOD_RESPONSERECOFFSET         DS   F     Offset of first response record
*                                         from start of MQOD
MQOD_OBJECTRECPTR              DS   F     Address of first object record
MQOD_RESPONSERECPTR            DS   F     Address of first response record
MQOD_ALTERNATESECURITYID       DS   XL40  Alternate security identifier
MQOD_RESOLVEDQNAME             DS   CL48  Resolved queue name
MQOD_RESOLVEDQMGRNAME          DS   CL48  Resolved queue manager name
MQOD_OBJECTSTRING              DS   F     Object Long name
MQOD_OBJECTSTRING_VSPTR        DS   F     Address of variable length string
MQOD_OBJECTSTRING_VSOFFSET     DS   F     Offset of variable length string
MQOD_OBJECTSTRING_VSBUFSIZE    DS   F     size of buffer
MQOD_OBJECTSTRING_VSLENGTH     DS   F     Length of variable length string
MQOD_OBJECTSTRING_VSCCSID      DS   F     CCSID of variable length string
MQOD_OBJECTSTRING_LENGTH       EQU  *- MQOD_OBJECTSTRING
                               ORG  MQOD_OBJECTSTRING
MQOD_OBJECTSTRING_AREA         DS   CL(MQOD_OBJECTSTRING_LENGTH)
*
MQOD_SELECTIONSTRING           DS   F     Message Selector
MQOD_SELECTIONSTRING_VSPTR     DS   F     Address of variable length string
MQOD_SELECTIONSTRING_VSOFFSET  DS   F     Offset of variable length string
MQOD_SELECTIONSTRING_VSBUFSIZE DS   F     size of buffer
MQOD_SELECTIONSTRING_VSLENGTH  DS   F     Length of variable length string
MQOD_SELECTIONSTRING_VSCCSID   DS   F     CCSID of variable length string
MQOD_SELECTIONSTRING_LENGTH    EQU  *- MQOD_SELECTIONSTRING
                               ORG  MQOD_SELECTIONSTRING
MQOD_SELECTIONSTRING_AREA      DS   CL(MQOD_SELECTIONSTRING_LENGTH)
*
MQOD_RESOBJECTSTRING           DS   F     Resolved Long object name
MQOD_RESOBJECTSTRING_VSPTR     DS   F     Address of variable length string
MQOD_RESOBJECTSTRING_VSOFFSET  DS   F     Offset of variable length string
MQOD_RESOBJECTSTRING_VSBUFSIZE DS   F     size of buffer
MQOD_RESOBJECTSTRING_VSLENGTH  DS   F     Length of variable length string
MQOD_RESOBJECTSTRING_VSCCSID   DS   F     CCSID of variable length string
MQOD_RESOBJECTSTRING_LENGTH    EQU  *- MQOD_RESOBJECTSTRING
                               ORG  MQOD_RESOBJECTSTRING
MQOD_RESOBJECTSTRING_AREA      DS   CL(MQOD_RESOBJECTSTRING_LENGTH)
MQOD_RESOLVEDTYPE              DS   F     Alias queue object resolved type
*
MQOD_LENGTH                    EQU  *-MQOD
                               ORG  MQOD
MQOD_AREA                      DS   CL(MQOD_LENGTH)
```

Dichiarazione Visual Basic per MQOD

```
Type MQOD
  StrucId             As String*4  'Structure identifier'
  Version             As Long      'Structure version number'
  ObjectType          As Long      'Object type'
  ObjectName          As String*48 'Object name'
  ObjectQMgrName      As String*48 'Object queue manager name'
  DynamicQName        As String*48 'Dynamic queue name'
  AlternateUserId     As String*12 'Alternate user identifier'
  RecsPresent         As Long      'Number of object records present'
  KnownDestCount      As Long      'Number of local queues opened'
                                   'successfully'
  UnknownDestCount    As Long      'Number of remote queues opened'
                                   'successfully'
  InvalidDestCount    As Long      'Number of queues that failed to'
                                   'open'
  ObjectRecOffset     As Long      'Offset of first object record from'
                                   'start of MQOD'
  ResponseRecOffset   As Long      'Offset of first response record'
                                   'from start of MQOD'
  ObjectRecPtr        As MQPTR     'Address of first object record'
  ResponseRecPtr      As MQPTR     'Address of first response record'
```

```
   AlternateSecurityId As MQBYTE40  'Alternate security identifier'
   ResolvedQName       As String*48 'Resolved queue name'
   ResolvedQMgrName    As String*48 'Resolved queue manager name'
 End Type
```

StrucId (MQCHAR4) per MQOD

Questo è l'identificativo della struttura del descrittore oggetto. È sempre un campo di immissione. Il valore è MQOD_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQOD
Identificativo per la struttura descrittore dell'oggetto.

Per il linguaggio di programmazione C, viene definita anche la costante MQOD_STRUC_ID_ARRAY. Ha lo stesso valore di MQOD_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQOD

Questo è il numero di versione della struttura; il valore deve essere uno dei seguenti:

MQOD_VERSION_1
Struttura del descrittore oggetto Version-1 .

MQOD_VERSION_2
Struttura descrittore oggetto Version-2 .

MQOD_VERSION_3
Struttura del descrittore oggetto Version-3 .

MQOD_VERSION_4
Struttura descrittore oggetto Version-4 .

Tutte le versioni sono supportate in tutti gli ambienti IBM MQ V7.0 .

I campi che esistono solo nelle versioni più recenti della struttura vengono identificati come tali nelle descrizioni dei campi. La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQOD_CURRENT_
Versione corrente della struttura descrittore dell'oggetto.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQOD_VERSION_1.

ObjectType (MQLONG) per MQOD

Il tipo di oggetto indicato nel descrittore dell'oggetto. I possibili valori sono:

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client. Il nome dell'oggetto si trova nel campo *ObjectName* .

MQOT_Q
Coda. Il nome dell'oggetto si trova nel campo *ObjectName* .

ELENCO NOMI MQOTT
Elenco nomi. Il nome dell'oggetto si trova nel campo *ObjectName*

PROCESSO MQOT_
Definizione processo. Il nome dell'oggetto si trova nel campo *ObjectName*

Gestore code MQOT_GR
Gestore code. Il nome dell'oggetto si trova nel campo *ObjectName*

TOPIC MQOT_T
. Il nome completo dell'argomento può essere creato da due campi differenti: *ObjectName* e *ObjectString*.

Per i dettagli su come vengono utilizzati questi due campi, consultare Combinazione di stringhe di argomenti.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQOT_Q.

ObjectName (MQCHAR48) per MQOD

Questo è il nome locale dell'oggetto come definito sul gestore code identificato da *ObjectQMgrName*. Il nome può contenere i seguenti caratteri:

- Caratteri alfabetici maiuscoli (da A a Z)
- Caratteri alfabetici minuscoli (da a a z)
- Cifre numeriche (da 0 a 9)
- Punto (.), barra (/), sottolineatura (_), percentuale (%)

Il nome non deve contenere spazi iniziali o intermedi, ma può contenere spazi finali. Utilizzare un carattere null per indicare la fine dei dati significativi nel nome; il valore null e i caratteri che lo seguono vengono trattati come spazi vuoti. Le seguenti limitazioni si applicano agli ambienti indicati:

- Sui sistemi che utilizzano EBCDIC Katakana, non è possibile utilizzare caratteri minuscoli.
- Su z/OS:
  - Evitare i nomi che iniziano o terminano con un carattere di sottolineatura; non possono essere elaborati dalle operazioni e dai pannelli di controllo.
  - Il carattere percentuale ha un significato speciale per RACF. Se RACF viene utilizzato come gestore della sicurezza esterno, i nomi non devono contenere la percentuale. In tal caso, tali nomi non vengono inclusi nei controlli di sicurezza quando vengono utilizzati i profili generici RACF .
- Su IBM i, i nomi contenenti caratteri minuscoli, barra o percentuale, devono essere racchiusi tra virgolette quando vengono specificati nei comandi. Questi apici non devono essere specificati per i nomi che si verificano come campi nelle strutture o come parametri nelle chiamate.

Il nome completo dell'argomento può essere creato da due campi differenti: *ObjectName* e *ObjectString*. Per i dettagli sul modo in cui questi due campi vengono utilizzati, consultare Combinazione di stringhe argomento.

I seguenti punti si applicano ai tipi di oggetto indicati:

- Se *ObjectName* è il nome di una coda modello, il gestore code crea una coda dinamica con gli attributi della coda modello e restituisce nel campo *ObjectName* il nome della coda creata. Una coda modello può essere specificata solo sulla chiamata MQOPEN; una coda modello non è valida sulla chiamata MQPUT1 .
- Se *ObjectName* è il nome di una coda alias con TARGTYPE (TOPIC), un controllo di sicurezza viene eseguito prima sulla coda alias denominata; ciò è normale quando vengono utilizzate le code alias. Quando il controllo di sicurezza viene completato correttamente, la chiamata MQOPEN continua e si comporta come una chiamata MQOPEN su un MQOT_TOPIC; ciò include l'esecuzione di un controllo di sicurezza rispetto all'oggetto argomento di gestione.
- Se *ObjectName* e *ObjectQMgrName* identificano una coda condivisa appartenente al gruppo di condivisione code a cui appartiene il gestore code locale, non deve essere presente anche una definizione di coda con lo stesso nome sul gestore code locale. Se esiste una definizione di questo tipo (una coda locale, una coda alias, una coda remota o una coda modello), la chiamata ha esito negativo con codice motivo MQRC_OBJECT_NOT_UNIQUE.
- Se l'oggetto che si sta aprendo è un elenco di distribuzione (ovvero, *RecsPresent* è presente e maggiore di zero), *ObjectName* deve essere vuoto o la stringa null. Se questa condizione non viene soddisfatta, la chiamata ha esito negativo con codice motivo MQRC_OBJECT_NAME_ERROR.
- Se *ObjectType* è MQOT_Q_MGR, si applicano le regole speciali; in questo caso, il nome deve essere completamente vuoto fino al primo carattere null o alla fine del campo.

Questo è un campo di input / output per la chiamata MQOPEN quando *ObjectName* è il nome di una coda modello e un campo di solo input in tutti gli altri casi. La lunghezza di questo campo è fornita da MQ_Q_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

Nome ObjectQMgr(MQCHAR48) per MQOD

Questo è il nome del gestore code su cui è definito l'oggetto *ObjectName* . I caratteri validi nel nome sono gli stessi di quelli per *ObjectName* (consultare “ObjectName (MQCHAR48) per MQOD” a pagina 499). Un nome che è completamente vuoto fino al primo carattere null o alla fine del campo indica il gestore code a cui è connessa l'applicazione (il gestore code locale).

I seguenti punti si applicano ai tipi di oggetto indicati:

- Se *ObjectType* è MQOT_TOPIC, MQOT_NAMELIST, MQOT_PROCESS o MQOT_Q_MGR, *ObjectQMgrName* deve essere vuoto o il nome del gestore code locale.
- Se *ObjectName* è il nome di una coda modello, il gestore code crea una coda dinamica con gli attributi della coda modello e restituisce nel campo *ObjectQMgrName* il nome del gestore code su cui è stata creata la coda; questo è il nome del gestore code locale. Una coda modello può essere specificata solo sulla chiamata MQOPEN; una coda modello non è valida sulla chiamata MQPUT1 .
- Se *ObjectName* è il nome di una coda cluster e *ObjectQMgrName* è vuoto, la destinazione dei messaggi inviati utilizzando l'handle della coda restituito dalla chiamata MQOPEN viene scelta dal gestore code (o dall'uscita del carico di lavoro del cluster, se installato) come segue:
  - Se viene specificato MQOO_BIND_ON_OPEN, il gestore code seleziona una particolare istanza della coda del cluster durante l'elaborazione della chiamata MQOPEN e tutti i messaggi immessi utilizzando questo handle della coda vengono inviati a tale istanza.
  - Se viene specificato MQOO_BIND_NOT_FIXED, il gestore code può scegliere un'istanza differente della coda di destinazione (che si trova su un gestore code diverso nel cluster) per ogni chiamata MQPUT successiva che utilizza questo handle di coda.

  Se l'applicazione deve inviare un messaggio a un'istanza *specifica* di una coda cluster (ovvero, un'istanza della coda che si trova su un determinato gestore code nel cluster), l'applicazione deve specificare il nome di tale gestore code nel campo *ObjectQMgrName* . Ciò forza il gestore code locale a inviare il messaggio al gestore code di destinazione specificato.
- Se *ObjectName* è il nome di una coda condivisa che è di proprietà del gruppo di condivisione code a cui appartiene il gestore code locale, *ObjectQMgrName* può essere il nome del gruppo di condivisione code, il nome del gestore code locale o vuoto; il messaggio viene inserito nella stessa coda, a seconda di quale di questi valori è specificato.

  I gruppi di condivisione code sono supportati solo su z/OS.
- Se *ObjectName* è il nome di una coda condivisa appartenente a un gruppo di condivisione code remoto (ovvero, un gruppo di condivisione code a cui non appartiene il gestore code locale), *ObjectQMgrName* deve essere il nome del gruppo di condivisione code. È possibile utilizzare il nome di un gestore code che appartiene a quel gruppo, ma ciò può ritardare il messaggio se quel particolare gestore code non è disponibile quando il messaggio arriva al gruppo di condivisione code.
- Se l'oggetto che si sta aprendo è un elenco di distribuzione (ovvero, *RecsPresent* è maggiore di zero), *ObjectQMgrName* deve essere vuoto o la stringa null. Se questa condizione non è soddisfatta, la chiamata ha esito negativo con codice motivo MQRC_OBJECT_Q_MGR_NAME_ERROR.

Questo è un campo di input / output per la chiamata MQOPEN quando *ObjectName* è il nome di una coda modello e un campo di solo input in tutti gli altri casi. La lunghezza di questo campo viene fornita da MQ_Q_MGR_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

DynamicQName (MQCHAR48) per MQOD

Questo è il nome di una coda dinamica che deve essere creata dalla chiamata MQOPEN. Ciò è rilevante solo quando *ObjectName* specifica il nome di una coda modello; in tutti gli altri casi *DynamicQName* viene ignorato.

I caratteri validi nel nome sono gli stessi di quelli per *ObjectName*, ma è valido anche un asterisco. Un nome che è vuoto (o uno in cui solo gli spazi sono presenti prima del primo carattere null) non è valido se *ObjectName* è il nome di una coda modello.

Se l'ultimo carattere non vuoto nel nome è un asterisco (*), il gestore code sostituisce l'asterisco con una stringa di caratteri che garantisce che il nome generato per la coda sia univoco nel gestore code locale.

Per consentire un numero di caratteri sufficiente, l'asterisco è valido solo nelle posizioni da 1 a 33. Non devono essere presenti caratteri diversi dagli spazi o un carattere null dopo l'asterisco.

È valido che l'asterisco si trovi nella posizione del primo carattere, nel qual caso il nome è costituito esclusivamente dai caratteri generati dal gestore code.

Su z/OS, non utilizzare un nome con l'asterisco nella prima posizione di carattere, poiché non possono essere eseguiti controlli di sicurezza su una coda con un nome completo generato automaticamente.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da MQ_Q_NAME_LENGTH. Il valore iniziale di questo campo è determinato dall'ambiente:

- Su z/OS, il valore è `'CSQ.*'`.
- Su altre piattaforme, il valore è `'AMQ.*'`.

Il valore è una stringa con terminazione null in C e una stringa con spazi vuoti in altri linguaggi di programmazione.

ID AlternateUser(MQCHAR12) per MQOD

Se si specifica MQOO_ALTERNATE_USER_AUTHORITY per la chiamata MQOPEN o MQPMO_ALTERNATE_USER_AUTHORITY per la chiamata MQPUT1 , questo campo contiene un identificativo utente alternativo utilizzato per controllare l'autorizzazione per l'apertura, al posto dell'identificativo utente con cui è attualmente in esecuzione l'applicazione. Alcuni controlli, tuttavia, vengono ancora eseguiti con l'identificativo utente corrente (ad esempio, controlli di contesto).

Se viene specificato MQOO_ALTERNATE_USER_AUTHORITY o MQPMO_ALTERNATE_USER_AUTHORITY e questo campo è completamente vuoto fino al primo carattere null o alla fine del campo, l'apertura può avere esito positivo solo se non è necessaria alcuna autorizzazione utente per aprire questo oggetto con le opzioni specificate.

Se non viene specificato né MQOO_ALTERNATE_USER_AUTHORITY né MQPMO_ALTERNATE_USER_AUTHORITY, questo campo viene ignorato.

Le seguenti differenze esistono negli ambienti indicati:

- Su z/OS, solo i primi 8 caratteri di *AlternateUserId* vengono utilizzati per controllare l'autorizzazione per l'apertura. Tuttavia, l'identificativo utente corrente deve essere autorizzato a specificare questo particolare identificativo utente alternativo; per questo controllo vengono utilizzati tutti i 12 caratteri dell'identificativo utente alternativo. L'identificativo utente deve contenere solo i caratteri consentiti dal gestore della sicurezza esterno.

  Se *AlternateUserId* viene specificato per una coda, il valore può essere utilizzato successivamente dal gestore code quando vengono inseriti i messaggi. Se le opzioni MQPMO_ * _CONTEXT specificate nella chiamata MQPUT o MQPUT1 fanno sì che il gestore code generi le informazioni sul contesto di identità, il gestore code inserisce il *AlternateUserId* nel campo *UserIdentifier* nell'MQMD del messaggio, al posto dell'identificativo utente corrente.
- In altri ambienti, *AlternateUserId* viene utilizzato solo per le verifiche del controllo accessi sull'oggetto che viene aperto. Se l'oggetto è una coda, *AlternateUserId* non influisce sul contenuto del campo *UserIdentifier* nell'MQMD dei messaggi inviati utilizzando tale handle di coda.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da MQ_USER_ID_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 12 caratteri vuoti in altri linguaggi di programmazione.

RecsPresent (MQLONG) per MQOD

Questo è il numero di record oggetto MQOR forniti dall'applicazione. Se questo numero è maggiore di zero, indica che si sta aprendo un elenco di distribuzione, con *RecsPresent* che rappresenta il numero di code di destinazione nell'elenco. Un elenco di distribuzione può contenere una sola destinazione.

Il valore di *RecsPresent* non deve essere inferiore a zero e se è maggiore di zero *ObjectType* deve essere MQOT_Q; la chiamata ha esito negativo con codice motivo MQRC_RECS_PRESENT_ERROR se queste condizioni non vengono soddisfatte.

Su z/OS, questo campo deve essere zero.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è minore di MQOD_VERSION_2.

Conteggio KnownDest(MQLONG) per MQOD

Questo è il numero di code nell'elenco di distribuzione che si risolvono in code locali e che sono state aperte correttamente. Il conteggio non include le code che si risolvono in code remote (anche se una coda di trasmissione locale viene utilizzata inizialmente per memorizzare il messaggio). Se presente, questo campo viene impostato anche quando si apre una singola coda che non si trova in un elenco di distribuzione.

Questo è un campo di output. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è minore di MQOD_VERSION_1.

Conteggio UnknownDest(MQLONG) per MQOD

Questo è il numero di code nell'elenco di distribuzione che si risolvono in code remote e che sono state aperte correttamente. Se presente, questo campo viene impostato anche quando si apre una singola coda che non si trova in un elenco di distribuzione.

Questo è un campo di output. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è minore di MQOD_VERSION_1.

Conteggio InvalidDest(MQLONG) per MQOD

Questo è il numero di code nell'elenco di distribuzione che non sono state aperte correttamente. Se presente, questo campo viene impostato anche quando si apre una singola coda che non si trova in un elenco di distribuzione.

Nota: Se presente, questo campo viene impostato solo se il parametro **CompCode** nella chiamata MQOPEN o MQPUT1 è MQCC_OK o MQCC_WARNING; non viene impostato se il parametro **CompCode** è MQCC_FAILED.

Questo è un campo di output. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è minore di MQOD_VERSION_1.

Offset ObjectRec(MQLONG) per MQOD

Questo è l'offset in byte del primo record di oggetto MQOR dall'inizio della struttura MQOD. L'offset può essere positivo o negativo. *ObjectRecOffset* viene utilizzato solo quando si sta aprendo un elenco di distribuzione. Il campo viene ignorato se *RecsPresent* è zero.

Quando viene aperto un elenco di distribuzione, è necessario fornire un array di uno o più record oggetto MQOR per specificare i nomi delle code di destinazione nell'elenco di distribuzione. Questa operazione può essere eseguita in due modi:

- Utilizzando il campo offset *ObjectRecOffset*.

  In questo caso, l'applicazione deve dichiarare la propria struttura contenente un MQOD seguito dall'array di record MQOR (con tutti gli elementi dell'array necessari) e impostare *ObjectRecOffset* sull'offset del primo elemento dell'array dall'inizio del MQOD. Assicurarsi che questo offset sia corretto e che abbia un valore che possa essere utilizzato all'interno di un MQLONG (il linguaggio di programmazione più restrittivo è COBOL, per cui l'intervallo valido è compreso tra -999 999 999 e +999 999 999).

  Utilizzare *ObjectRecOffset* per i linguaggi di programmazione che non supportano il tipo di dati puntatore o che implementano il tipo di dati puntatore in un modo non portabile in ambienti differenti (ad esempio, il linguaggio di programmazione COBOL).
- Utilizzando il campo puntatore *ObjectRecPtr*.

  In questo caso, l'applicazione può dichiarare l'array delle strutture MQOR separatamente dalla struttura MQOD e impostare *ObjectRecPtr* sull'indirizzo dell'array.

Utilizzare *ObjectRecPtr* per i linguaggi di programmazione che supportano il tipo di dati puntatore in un modo che sia portabile in ambienti differenti (ad esempio, il linguaggio di programmazione C).

Qualunque sia la tecnica scelta, utilizzare una delle opzioni *ObjectRecOffset* e *ObjectRecPtr* ; la chiamata ha esito negativo con codice motivo MQRC_OBJECT_RECORDS_ERROR se entrambi sono zero o entrambi sono diversi da zero.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è minore di MQOD_VERSION_2.

Offset ResponseRec(MQLONG) per MQOD

Questo è l'offset in byte del primo record di risposta MQRR dall'inizio della struttura MQOD. L'offset può essere positivo o negativo. *ResponseRecOffset* viene utilizzato solo quando si sta aprendo un elenco di distribuzione. Il campo viene ignorato se *RecsPresent* è zero.

Quando un elenco di distribuzione è in fase di apertura, è possibile fornire un array di uno o più record di risposta MQRR per identificare le code che non sono state aperte (campo*CompCode* in MQRR) e il motivo di ciascun errore (campo*Reason* in MQRR). I dati vengono restituiti nella schiera di record di risposta nello stesso ordine in cui i nomi coda si verificano nella schiera di record oggetto. Il gestore code imposta i record di risposta solo quando il risultato della chiamata è misto (ovvero, alcune code sono state aperte correttamente mentre altre non sono riuscite o tutte non sono riuscite, ma per motivi diversi); il codice motivo MQRC_MULTIPLE_REASON dalla chiamata indica questo caso. Se lo stesso codice motivo si applica a tutte le code, tale motivo viene restituito nel parametro **Reason** della chiamata MQOPEN o MQPUT1 e i record di risposta non vengono impostati. I record di risposta sono facoltativi, ma se vengono forniti devono essere *RecsPresent* .

I record di risposta possono essere forniti nello stesso modo dei record di oggetto, specificando un offset in *ResponseRecOffset*o specificando un indirizzo in *ResponseRecPtr* ; per i dettagli su come eseguire questa operazione, consultare “Offset ObjectRec(MQLONG) per MQOD” a pagina 502. Tuttavia, non è possibile utilizzare più di uno tra *ResponseRecOffset* e *ResponseRecPtr* ; la chiamata ha esito negativo con codice di errore MQRC_RESPONSE_RECORDS_ERROR se entrambi sono diversi da zero.

Per la chiamata MQPUT1 , questi record di risposta vengono utilizzati per restituire informazioni sugli errori che si verificano quando il messaggio viene inviato alle code nell'elenco di distribuzione, nonché gli errori che si verificano quando le code vengono aperte. Il codice di completamento e il codice motivo dell'operazione di inserimento per una coda sostituiscono quelli dell'operazione di apertura per tale coda solo se il codice di completamento da quest' ultima era MQCC_OK o MQCC_WARNING.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è minore di MQOD_VERSION_2.

ObjectRecPtr (MQPTR) per MQOD

Questo è l'indirizzo del primo record di oggetto MQOR. *ObjectRecPtr* viene utilizzato solo quando si sta aprendo un elenco di distribuzione. Il campo viene ignorato se *RecsPresent* è zero.

È possibile utilizzare *ObjectRecPtr* o *ObjectRecOffset* per specificare i record oggetto, ma non entrambi; per la descrizione del campo *ObjectRecOffset* , consultare “Offset ObjectRec(MQLONG) per MQOD” a pagina 502. Se non si utilizza *ObjectRecPtr*, impostarlo sul puntatore null o sui byte null.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e, in caso contrario, una stringa di byte completamente null. Questo campo viene ignorato se *Version* è minore di MQOD_VERSION_2.

Nota: Sulle piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata, con il valore iniziale che è la stringa di byte null.

ResponseRecPtr (MQPTR) per MQOD

Questo è l'indirizzo del primo record di risposta MQRR. *ResponseRecPtr* viene utilizzato solo quando si sta aprendo un elenco di distribuzione. Il campo viene ignorato se *RecsPresent* è zero.

Utilizzare *ResponseRecPtr* o *ResponseRecOffset* per specificare i record di risposta, ma non entrambi; per i dettagli, vedere "Offset ResponseRec(MQLONG) per MQOD" a pagina 503. Se non si utilizza *ResponseRecPtr*, impostarlo sul puntatore null o sui byte null.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e, in caso contrario, una stringa di byte completamente null. Questo campo viene ignorato se *Version* è minore di MQOD_VERSION_2.

Nota: Sulle piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata, con il valore iniziale che è la stringa di byte null.

ID AlternateSecurity(MQBYTE40) per MQOD

Si tratta di un identificativo di sicurezza che viene passato con *AlternateUserId* al servizio di autorizzazione per consentire l'esecuzione dei controlli di autorizzazione appropriati. *AlternateSecurityId* viene utilizzato solo se:

- MQOO_ALTERNATE_USER_AUTHORITY è specificato nella chiamata MQOPEN oppure
- MQPMO_ALTERNATE_USER_AUTHORITY è specificato sulla chiamata MQPUT1 ,

e il campo *AlternateUserId* non è completamente vuoto fino al primo carattere null o alla fine del campo.

Su Windows, *AlternateSecurityId* può essere utilizzato per fornire il SID (security identifier) Windows che identifica in maniera univoca *AlternateUserId*. Il SID per un utente può essere ottenuto dal sistema Windows utilizzando la chiamata API `LookupAccountName()` Windows .

Su z/OS, questo campo viene ignorato.

Il campo *AlternateSecurityId* ha la seguente struttura:

- Il primo byte è un numero intero binario contenente la lunghezza dei dati significativi che seguono; il valore esclude il byte di lunghezza stesso. Se non è presente alcun identificativo di sicurezza, la lunghezza è zero.
- Il secondo byte indica il tipo di identificativo di sicurezza presente; sono possibili i seguenti valori:

  ID SICUREZZA MQSIDT_NT_SECURITY_ID
  Identificativo di sicurezza Windows .

  MQSIDT_NONE
  Nessun identificativo di sicurezza.
- Il terzo byte e i byte successivi fino alla lunghezza definita dal primo byte contengono l'identificativo di sicurezza stesso.
- I byte rimanenti nel campo sono impostati a zero binario.

È possibile utilizzare il seguente valore speciale:

MQSID_NONE
Nessun identificativo di sicurezza specificato.

Il valore è zero binario per la lunghezza del campo.

Per il linguaggi di programmazione C, viene definita anche la costante MQSID_NONE_ARRAY, che ha lo stesso valore di MQSID_NONE, ma è un array di caratteri anziché una stringa.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da MQ_SECURITY_ID_LENGTH. Il valore iniziale di questo campo è MQSID_NONE. Questo campo viene ignorato se *Version* è minore di MQOD_VERSION_3.

ResolvedQName (MQCHAR48) per MQOD

Questo è il nome della coda di destinazione dopo che il gestore code locale ha risolto il nome. Il nome restituito è il nome di una coda che esiste sul gestore code identificato da *ResolvedQMgrName*.

Un valore non vuoto viene restituito solo se l'oggetto è una singola coda aperta per la ricerca, l'immissione o l'emissione (o qualsiasi combinazione). Se l'oggetto aperto è uno dei seguenti, *ResolvedQName* viene impostato su spazi vuoti:

- Non è una coda
- Una coda, ma non aperta per la ricerca, l'immissione o l'emissione
- Un elenco di distribuzione
- Una coda alias che fa riferimento a un oggetto argomento (fare riferimento a ResObjectString).
- Una coda alias che si risolve in un oggetto argomento.

Questo è un campo di output. La lunghezza di questo campo è fornita da MQ_Q_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione. Questo campo viene ignorato se *Version* è minore di MQOD_VERSION_3.

Nome ResolvedQMgr(MQCHAR48) per MQOD

È il nome del gestore code di destinazione dopo che il gestore code locale ha risolto il nome. Il nome restituito è il nome del gestore code proprietario della coda identificata da *ResolvedQName*. *ResolvedQMgrName* può essere il nome del gestore code locale.

Se *ResolvedQName* è una coda condivisa di proprietà del gruppo di condivisione code a cui appartiene il gestore code locale, *ResolvedQMgrName* è il nome del gruppo di condivisione code. Se la coda è di proprietà di un altro gruppo di condivisione code, *ResolvedQName* può essere il nome del gruppo di condivisione code o il nome di un gestore code che è un membro del gruppo di condivisione code (la natura del valore restituito è determinata dalle definizioni di coda che esistono nel gestore code locale).

Un valore non vuoto viene restituito solo se l'oggetto è una singola coda aperta per la ricerca, l'immissione o l'emissione (o qualsiasi combinazione). Se l'oggetto aperto è uno dei seguenti, *ResolvedQMgrName* viene impostato su spazi vuoti:

- Non è una coda
- Una coda, ma non aperta per la ricerca, l'immissione o l'emissione
- Una coda cluster con MQOO_BIND_NOT_FIXED specificato (o con MQOO_BIND_AS_Q_DEF attivo quando l'attributo della coda **DefBind** ha il valore MQBND_BIND_NOT_FIXED)
- Un elenco di distribuzione

Questo è un campo di output. La lunghezza di questo campo è fornita da MQ_Q_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione. Questo campo viene ignorato se *Version* è minore di MQOD_VERSION_3.

ObjectString (MQCHARV) per MQOD

Il campo ObjectString specifica il nome oggetto lungo.

Specifica il nome oggetto lungo da utilizzare. Questo campo è indicato solo per alcuni valori di *ObjectType*e viene ignorato per tutti gli altri valori. Vedere la descrizione di *ObjectType* per i dettagli su quali valori indicano che questo campo è utilizzato.

Se *ObjectString* viene specificato in modo non corretto, in base alla descrizione di come utilizzare la struttura MQCHARV o se supera la lunghezza massima, la chiamata ha esito negativo con codice motivo MQRC_OBJECT_STRING_ERROR.

Questo è un campo di immissione. I valori iniziali dei campi in questa struttura sono gli stessi della struttura MQCHARV.

Il nome completo dell'argomento può essere creato da due campi differenti: *ObjectName* e *ObjectString*. Per i dettagli sul modo in cui questi due campi vengono utilizzati, consultare Combinazione di stringhe argomento.

SelectionString (MQCHARV) per MQOD

Questa è la stringa utilizzata per fornire i criteri di selezione utilizzati durante il richiamo dei messaggi da una coda.

SelectionString non deve essere fornito nei casi seguenti:

- Se *ObjectType* non è MQOT_Q
- Se la coda aperta non viene aperta utilizzando una delle opzioni MQOO_BROWSE o MQOO_INPUT_ *

Se *SelectionString* viene fornito in questi casi, la chiamata ha esito negativo con codice motivo MQRC_SELECTOR_INVALID_FOR_TYPE.

Se *SelectionString* viene specificato in modo non corretto, in base alla descrizione di come utilizzare la struttura "MQCHARV - Stringa a lunghezza variabile" a pagina 299 o se supera la lunghezza massima, la chiamata ha esito negativo con codice motivo MQRC_SELECTION_STRING_ERROR. La lunghezza massima di *SelectionString* è MQ_SELECTOR_LENGTH.

L'utilizzo di *SelectionString* è descritto in Selettori.

ResObjectStringa (MQCHARV) per MQOD

Il campo ResObjectString è il nome dell'oggetto lungo dopo che il gestore code ha risolto il nome fornito nel campo *ObjectName* .

Questo campo viene restituito solo per gli argomenti e gli alias della coda che fanno riferimento a un oggetto argomento.

Se il nome oggetto lungo viene fornito in *ObjectString* e non viene fornito nulla in *ObjectName*, il valore restituito in questo campo è uguale a quello fornito in *ObjectString*.

Se questo campo viene omesso (ovvero ResObjectString.VSBufSize è zero), *ResObjectString* non verrà restituito, ma la lunghezza verrà restituita in ResObjectString.VSLength.

Se la lunghezza del buffer (fornita in ResObjectStrng.VSBufSize) è inferiore al *ResObjectString*completo, la stringa verrà troncata e restituirà il maggior numero di caratteri a destra possibile nel buffer fornito.

Se *ResObjectString* non viene specificato correttamente, in base alla descrizione di come utilizzare la struttura MQCHARV o se supera la lunghezza massima, la chiamata ha esito negativo con codice motivo MQRC_RES_OBJECT_STRING_ERROR.

ResolvedType (MQLONG) per MQOD

Il tipo di oggetto risolto (base) che si sta aprendo.

I valori possibili sono:

MQOT_Q
L'oggetto risolto è una coda. Questo valore si applica quando una coda viene aperta direttamente o quando viene aperta una coda alias che punta a una coda.

TOPIC MQOT_T
L'oggetto risolto è un argomento. Questo valore si applica quando un argomento viene aperto direttamente o quando viene aperta una coda alias che punta a un oggetto argomento.

MQOT_NONE
Il tipo risolto non è né una coda né un argomento.

MQOR - Record oggetto

Utilizzare la struttura MQOR per specificare il nome coda e il nome gestore code di una singola coda di destinazione. MQOR è una struttura di input per chiamate MQOPEN e MQPUT1 .

Disponibilità

La struttura MQOR è disponibile sulle piattaforme seguenti:

- AIX
- IBM i

- Linux
- Windows

e per IBM MQ MQI clients collegati a questi sistemi.

Serie di caratteri e codifica

I dati in MQOR devono trovarsi nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da MQENC_NATIVE. Tuttavia, se l'applicazione è in esecuzione come client MQ MQI, la struttura deve essere nella serie di caratteri e nella codifica del client.

Utilizzo

Fornendo un array di queste strutture sulla chiamata MQOPEN, è possibile aprire un elenco di code; questo elenco viene denominato elenco di distribuzione. Ogni messaggio inserito utilizzando l'handle della coda restituito dalla chiamata MQOPEN viene inserito su ciascuna delle code nell'elenco, purché la coda sia stata aperta correttamente.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 505. Campi in MQOR per MQOR

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
ObjectName (nome oggetto)	Nessuna	Stringa null o spazi vuoti
ObjectQMgrNome (nome gestore code oggetto)	Nessuna	Stringa null o spazi vuoti

Note:

1. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.
2. Nel linguaggio di programmazione C, la variabile macroMQOR_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura:

```
MQOR MyOR = {MQOR_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQOR

```
typedef struct tagMQOR MQOR;
struct tagMQOR {
  MQCHAR48  ObjectName;      /* Object name */
  MQCHAR48  ObjectQMgrName;  /* Object queue manager name */
};
```

Dichiarazione COBOL per MQOR

```
**   MQOR structure
  10 MQOR.
```

```
**    Object name
   15 MQOR-OBJECTNAME     PIC X(48).
**    Object queue manager name
   15 MQOR-OBJECTQMGRNAME PIC X(48).
```

Dichiarazione PL/I per MQOR

```
dcl
 1 MQOR based,
  3 ObjectName     char(48), /* Object name */
  3 ObjectQMgrName char(48); /* Object queue manager name */
```

Dichiarazione Visual Basic per MQOR

```
Type MQOR
  ObjectName     As String*48 'Object name'
  ObjectQMgrName As String*48 'Object queue manager name'
End Type
```

ObjectName (MQCHAR48) per MQOR

È uguale al campo *ObjectName* nella struttura MQOD (per i dettagli, consultare MQOD), tranne che:

- Deve essere il nome di una coda.
- Non deve essere il nome di una coda modello.

Questo è sempre un campo di input. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

Nome ObjectQMgr(MQCHAR48) per MQOR

È uguale al campo *ObjectQMgrName* nella struttura MQOD (per i dettagli, consultare MQOD).

Questo è sempre un campo di input. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

MQPD - Descrittore proprietà

La struttura **MQPD** viene utilizzata per definire gli attributi di una proprietà. La struttura è un parametro di input / output sulla chiamata MQSETMP e un parametro di input sulla chiamata MQINQMP.

Disponibilità

La struttura **MQPD** è disponibile sulle piattaforme seguenti:

- AIX
- IBM i
- Linux
- Windows
- z/OS

e per IBM MQ MQI clients.

Serie di caratteri e codifica

I dati in **MQPD** devono essere nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (**MQENC_NATIVE**).

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 506. Campi in MQPD

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQPD	'PD'
Versione (numero versione struttura)	MQPD_VERSION_1	1
Opzioni (opzioni)	MQPD_NONE	0
Supporto (supporto richiesto per la proprietà del messaggio)	MQPD_SUPPORT_OPZIONALE	0
Contesto (contesto del messaggio a cui appartiene la proprietà)	MQPD_NO_CONTEXT	0
CopyOptions (opzioni di copia a cui appartiene la proprietà)	MQCOPY_DEFAULT	0

Note:

1. Nel linguaggio di programmazione C, la variabile macro MQPD_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura:

```
MQPD MyPD = {MQPD_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQPD

```
typedef struct tagMQPD MQPD;
struct tagMQPD {
  MQCHAR4  StrucId;     /* Structure identifier */
  MQLONG   Version;     /* Structure version number */
  MQLONG   Options;     /* Options that control the action of
                           MQSETMP and MQINQMP */
  MQLONG   Support;     /* Property support option */
  MQLONG   Context;     /* Property context */
  MQLONG   CopyOptions; /* Property copy options */
};
```

Dichiarazione COBOL per MQPD

```
**   MQPD structure
  10 MQPD.
**    Structure identifier
   15 MQPD-STRUCID PIC X(4).
**    Structure version number
   15 MQPD-VERSION PIC S9(9) BINARY.
**    Options that control the action of MQSETMP and
**    MQINQMP
   15 MQPD-OPTIONS PIC S9(9) BINARY.
**    Property support option
   15 MQPD-SUPPORT PIC S9(9) BINARY.
**    Property context
   15 MQPD-CONTEXT PIC S9(9) BINARY.
**    Property copy options
   15 MQPD-COPYOPTIONS PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQPD

```
dcl
 1 MQPD based,
  3 StrucId     char(4),       /* Structure identifier */
  3 Version     fixed bin(31), /* Structure version number */
  3 Options     fixed bin(31), /* Options that control the action
                                  of MQSETMP and MQINQMP */
  3 Support     fixed bin(31), /* Property support option */
  3 Context     fixed bin(31), /* Property context */
  3 CopyOptions fixed bin(31); /* Property copy options */
```

Dichiarazione High Level Assembler per MQPD

```
MQPD             DSECT
MQPD_STRUCID     DS   CL4    Structure identifier
MQPD_VERSION     DS   F      Structure version number
MQPD_OPTIONS     DS   F      Options that control the
*                            action of MQSETMP and MQINQMP
MQPD_SUPPORT     DS   F      Property support option
MQPD_CONTEXT     DS   F      Property context
MQPD_COPYOPTIONS DS   F      Property copy options
MQPD_LENGTH      EQU  *-MQPD
MQPD_AREA        DS   CL(MQPD_LENGTH)
```

StrucId (MQCHAR4) per MQPD

Questo è l'identificativo della struttura del descrittore della proprietà. È sempre un campo di immissione. Il valore è MQPD_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQPD
Identificativo per la struttura del descrittore della proprietà.

Per il linguaggio di programmazione C, viene definita anche la costante **MQPD_STRUC_ID_ARRAY** . Ha lo stesso valore di **MQPD_STRUC_ID**, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQPD

Questo è il numero di versione della struttura; il valore deve essere:

MQPD_VERSION_1
Struttura descrittore della proprietà Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQPD_CURRENT_
La versione corrente della struttura descrittore della proprietà.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **MQPD_VERSION_1**.

Opzioni (MQLONG) per MQPD

Il valore deve essere:

MQPD_NONE
Nessuna opzione specificata

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQPD_NONE.

Supporto (MQLONG) per MQPD

Questo campo descrive il livello di supporto per la proprietà del messaggio richiesto dal gestore code, affinché il messaggio che contiene questa proprietà venga inserito in una coda. Ciò si applica solo alle proprietà definite da IBM MQ; il supporto per tutte le altre proprietà è facoltativo.

Il campo viene impostato automaticamente sul valore corretto quando la proprietà definita IBM MQè nota al gestore code. Se la proprietà non è riconosciuta, viene assegnato MQPD_SUPPORT_OPTIONAL. Quando un gestore code riceve un messaggio contenente una proprietà definita da IBM MQche il gestore code riconosce come non corretta, corregge il valore del campo *Support* .

Quando si imposta una proprietà definita da IBM MQutilizzando la chiamata MQSETMP su un handle del messaggio in cui è stata impostata l'opzione MQCMHO_NO_VALIDATION, *Support* diventa un campo di input. Ciò consente a un'applicazione di inserire una proprietà definita da IBM MQ, con il valore corretto, in cui la proprietà non è supportata dal gestore code connesso, ma in cui il messaggio è destinato ad essere elaborato su un altro gestore code.

Il valore MQPD_SUPPORT_OPTIONAL è sempre assegnato alle proprietà non definite da IBM MQ.

Se un gestore code IBM WebSphere MQ 7.0 , che supporta le proprietà del messaggio, riceve una proprietà che contiene un valore *Support* non riconosciuto, la proprietà viene considerata come se:

- MQPD_SUPPORT_REQUIRED è stato specificato se uno dei valori non riconosciuti è contenuto in MQPD_REJECT_UNSUP_MASK.
- MQPD_SUPPORT_REQUIRED_IF_LOCAL è stato specificato se uno qualsiasi dei valori non riconosciuti è contenuto in MQPD_ACCEPT_UNSUP_IF_XMIT_MASK
- MQPD_SUPPORT_OPTIONAL è stato specificato diversamente.

Uno dei seguenti valori viene restituito dalla chiamata MQINQMP, oppure è possibile specificare uno dei valori, quando si utilizza la chiamata MQSETMP su un handle del messaggio in cui è impostata l'opzione MQCMHO_NO_VALIDATION:

MQPD_SUPPORT_OPZIONALE
La proprietà viene accettata da un gestore code anche se non è supportata. La proprietà può essere eliminata in modo che il messaggio possa fluire in un gestore code che non supporta le proprietà del messaggio. Questo valore viene assegnato anche a proprietà non definite da IBM MQ.

MQPD_SUPPORT_XX_ENCODE_CASE_ONE obbligatorio
È richiesto il supporto per la proprietà. Il messaggio è stato rifiutato da un gestore code che non supporta la proprietà definita da IBM MQ. La chiamata MQPUT o MQPUT1 ha esito negativo con codice di completamento MQCC_FAILED e codice motivo MQRC_UNSUPPORTED_PROPERTY.

MQPD_SUPPORT_REQUIRED_IF_LOCAL
Il messaggio viene rifiutato da un gestore code che non supporta la proprietà definita da IBM MQse il messaggio è destinato a una coda locale. La chiamata MQPUT o MQPUT1 ha esito negativo con codice di completamento MQCC_FAILED e codice motivo MQRC_UNSUPPORTED_PROPERTY.

La chiamata MQPUT o MQPUT1 ha esito positivo se il messaggio è destinato a un gestore code remoto.

Questo è un campo di output nella chiamata MQINQMP e un campo di immissione nella chiamata MQSETMP se l'handle del messaggio è stato creato con l'opzione MQCMHO_NO_VALIDATION impostata. Il valore iniziale di questo campo è MQPD_SUPPORT_OPTIONAL.

Contesto (MQLONG) per MQPD

Descrive il contesto del messaggio a cui appartiene la proprietà.

Quando un gestore code riceve un messaggio contenente una proprietà definita da IBM MQche il gestore code riconosce come non corretta, corregge il valore del campo *Context* .

È possibile specificare la seguente opzione:

CONTEXT MQPD_USER_
La proprietà è associata al contesto utente.

Non è richiesta alcuna autorizzazione speciale per poter impostare una proprietà associata al contesto utente utilizzando la chiamata MQSETMP.

Su un gestore code IBM WebSphere MQ 7.0 , una proprietà associata al contesto utente viene salvata come descritto per MQOO_SAVE_ALL_CONTEXT. Una chiamata MQPUT con

MQPMO_PASS_ALL_CONTEXT specificato, fa sì che la proprietà venga copiata dal contesto salvato nel nuovo messaggio.

Se l'opzione precedentemente descritta non è richiesta, è possibile utilizzare la seguente opzione:

MQPD_NO_CONTEXT
La proprietà non è associata a un contesto di messaggio.

Un valore non riconosciuto viene rifiutato con un codice *Reason* di MQRC_PD_ERROR

Si tratta di un campo di input / output per la chiamata MQSETMP e di un campo di output dalla chiamata MQINQMP. Il valore iniziale di questo campo è MQPD_NO_CONTEXT.

CopyOptions (MQLONG) per MQPD

Descrive in quale tipo di messaggi deve essere copiata la proprietà. Questo è un campo di sola emissione per le IBM MQ proprietà definite IBM MQ riconosciute; imposta il valore appropriato.

Quando un gestore code riceve un messaggio contenente una proprietà definita IBM MQ che il gestore code riconosce come non corretta, corregge il valore del campo *CopyOptions* .

È possibile specificare una o più di queste opzioni. Per specificare più di un'opzione, aggiungere i valori insieme (non aggiungere la stessa costante più di una volta) o combinare i valori utilizzando l'operazione OR bit per bit (se il linguaggio di programmazione supporta le operazioni bit).

MQCOPY_FORWARD
Questa proprietà viene copiata in un messaggio inoltrato.

MQCOPY_PUBLISH
Questa proprietà viene copiata nel messaggio ricevuto da un sottoscrittore quando viene pubblicato un messaggio.

MQCOPY_REPLY
Questa proprietà viene copiata in un messaggio di risposta.

PROSPETTO MQCOPY_REPORT
Questa proprietà viene copiata in un messaggio di report.

MQCOPY_ALL
Questa proprietà viene copiata in tutti i messaggi successivi.

Opzione predefinita: è possibile specificare la seguente opzione per fornire la serie di opzioni di copia predefinita:

MQCOPY_DEFAULT
Questa proprietà viene copiata in un messaggio inoltrato, in un messaggio di report o in un messaggio ricevuto da un sottoscrittore quando viene pubblicato un messaggio.

Ciò equivale a specificare la combinazione delle opzioni MQCOPY_FORWARD, MQCOPY_REPORT e MQCOPY_PUBLISH.

Se nessuna delle opzioni descritte in precedenza è richiesta, utilizzare la seguente opzione:

MQCOPY_NONE
Utilizzare questo valore per indicare che non sono specificate altre opzioni di copia; programmaticamente non esiste alcuna relazione tra questa proprietà e i messaggi successivi. Viene sempre restituito per le proprietà del descrittore del messaggio.

Si tratta di un campo di input / output per la chiamata MQSETMP e di un campo di output dalla chiamata MQINQMP. Il valore iniziale di questo campo è MQCOPY_DEFAULT.

MQPMO - Opzioni inserimento messaggio

La struttura MQPMO consente all'applicazione di specificare le opzioni che controllano il modo in cui i messaggi vengono inseriti nelle code o pubblicati negli argomenti. La struttura è un parametro di input / output nelle chiamate MQPUT e MQPUT1 .

Versione

La versione corrente di MQPMO è MQPMO_VERSION_3. Alcuni campi sono disponibili solo in determinate versioni di MQPM. Se è necessario eseguire il port delle applicazioni tra diversi ambienti, è necessario assicurarsi che la versione di MQPMO sia congruente in tutti gli ambienti. I campi che esistono solo in particolari versioni della struttura vengono identificati come tali in questo argomento e nelle descrizioni dei campi.

I file di intestazione, COPY e INCLUDE forniti per i linguaggi di programmazione supportati contengono la versione più recente di MQPMO supportata dall'ambiente, ma con il valore iniziale del campo *Version* impostato su MQPMO_VERSION_1. Per utilizzare i campi non presenti nella struttura version-1 , l'applicazione deve impostare il campo *Version* sul numero di versione della versione richiesta.

Serie di caratteri e codifica

I dati in MQPMO devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da MQENC_NATIVE. Tuttavia, se l'applicazione è in esecuzione come client MQ MQI, la struttura deve essere nella serie di caratteri e nella codifica del client.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 507. Campi in MQPMO

Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
StrucId (identificativo della struttura)	ID_STRUC_MQPMO	'PMO¬'
Versione (numero versione struttura)	MQPMO_VERSION_1	1
Opzioni (opzioni che controllano l'azione di MQPUT e MQPUT1)	MQPMO_NONE	0
Timeout (riservato)	Nessuna	-1
Contesto (handle oggetto della coda di immissione)	Nessuna	0
KnownDestKnownDest (numero di messaggi inviati correttamente alle code locali)	Nessuna	0
UnknownDestsconosciuto (numero di messaggi inviati correttamente alle code remote)	Nessuna	0
InvalidDestConteggio (numero di messaggi che non possono essere inviati)	Nessuna	0
ResolvedQName (nome risolto della coda di destinazione)	Nessuna	Stringa null o spazi vuoti
ResolvedQMgrNome (nome risolto del gestore code di destinazione)	Nessuna	Stringa null o spazi vuoti
Nota: I restanti campi vengono ignorati se *Version* è minore di MQPMO_VERSION_2.		
RecsPresent (numero di record di messaggi di inserimento o di risposta presenti)	Nessuna	0
PutMsgRecFields (indicatori che indicano quali campi MQPMR sono presenti)	MQPMRF_NONE	0

Tabella 507. Campi in MQPMO (Continua)

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
PutMsgRecOffset (offset del record del primo messaggio di inserimento dall'inizio di MQPM)	Nessuna	0
ResponseRec (offset del primo record di risposta dall'avvio di MQPMO)	Nessuna	0
PutMsgRecPtr (indirizzo del record del primo messaggio inserito)	Nessuna	Puntatore null o byte null
ResponseRecPtr (indirizzo del primo record di risposta)	Nessuna	Puntatore null o byte null
Nota: I restanti campi vengono ignorati se *Version* è minore di MQPMO_VERSION_3.		
OriginalMsgHandle (handle del messaggio originale)	MQHM_NONE	0
NewMsgHandle (nuovo handle del messaggio)	MQHM_NONE	0
Azione (tipo di inserimento eseguito e relazione tra il messaggio originale specificato nel campo *OriginalMsgHandle* e il nuovo messaggio specificato nel campo *NewMsgHandle*)	NUOVO	0
PubLevel (livello di sottoscrizione di destinazione della pubblicazione)	Nessuna	9

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.
3. Nel linguaggio di programmazione C, la variabile macroMQPMO_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQPMO MyPMO = {MQPMO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQPMO

```
typedef struct tagMQPMO MQPMO;
struct tagMQPMO {
  MQCHAR4   StrucId;            /* Structure identifier */
  MQLONG    Version;            /* Structure version number */
  MQLONG    Options;            /* Options that control the action of
                                   MQPUT and MQPUT1 */
  MQLONG    Timeout;            /* Reserved */
  MQHOBJ    Context;            /* Object handle of input queue */
  MQLONG    KnownDestCount;     /* Number of messages sent
                                   successfully to local queues */
  MQLONG    UnknownDestCount;   /* Number of messages sent
                                   successfully to remote queues */
  MQLONG    InvalidDestCount;   /* Number of messages that could not
                                   be sent */
  MQCHAR48  ResolvedQName;      /* Resolved name of destination
                                   queue */
```

```
  MQCHAR48  ResolvedQMgrName;   /* Resolved name of destination queue
                                   manager */
  /* Ver:1 */
  MQLONG    RecsPresent;        /* Number of put message records or
                                   response records present */
  MQLONG    PutMsgRecFields;    /* Flags indicating which MQPMR fields
                                   are present */
  MQLONG    PutMsgRecOffset;    /* Offset of first put message record
                                   from start of MQPMO */
  MQLONG    ResponseRecOffset;  /* Offset of first response record
                                   from start of MQPMO */
  MQPTR     PutMsgRecPtr;       /* Address of first put message
                                   record */
  MQPTR     ResponseRecPtr;     /* Address of first response record */
  /* Ver:2 */
  MQHMSG    OriginalMsgHandle;  /* Original message handle */
  MQHMSG    NewMsgHandle;       /* New message handle */
  MQLONG    Action;             /* The action being performed */
  MQLONG    PubLevel;           /* Subscription level */
  /* Ver:3 */
};
```

Dichiarazione COBOL per MQPMO

```
**   MQPMO structure
  10 MQPMO.
**    Structure identifier
   15 MQPMO-STRUCID           PIC X(4).
**    Structure version number
   15 MQPMO-VERSION           PIC S9(9) BINARY.
**    Options that control the action of MQPUT and MQPUT1
   15 MQPMO-OPTIONS           PIC S9(9) BINARY.
**    Reserved
   15 MQPMO-TIMEOUT           PIC S9(9) BINARY.
**    Object handle of input queue
   15 MQPMO-CONTEXT           PIC S9(9) BINARY.
**    Number of messages sent successfully to local queues
   15 MQPMO-KNOWNDESTCOUNT    PIC S9(9) BINARY.
**    Number of messages sent successfully to remote queues
   15 MQPMO-UNKNOWNDESTCOUNT  PIC S9(9) BINARY.
**    Number of messages that could not be sent
   15 MQPMO-INVALIDDESTCOUNT  PIC S9(9) BINARY.
**    Resolved name of destination queue
   15 MQPMO-RESOLVEDQNAME     PIC X(48).
**    Resolved name of destination queue manager
   15 MQPMO-RESOLVEDQMGRNAME  PIC X(48).
**    Number of put message records or response records present
   15 MQPMO-RECSPRESENT       PIC S9(9) BINARY.
**    Flags indicating which MQPMR fields are present
   15 MQPMO-PUTMSGRECFIELDS   PIC S9(9) BINARY.
**    Offset of first put message record from start of MQPMO
   15 MQPMO-PUTMSGRECOFFSET   PIC S9(9) BINARY.
**    Offset of first response record from start of MQPMO
   15 MQPMO-RESPONSERECOFFSET PIC S9(9) BINARY.
**    Address of first put message record
   15 MQPMO-PUTMSGRECPTR      POINTER.
**    Address of first response record
   15 MQPMO-RESPONSERECPTR    POINTER.
**    Original message handle
   15 MQPMO-ORIGINALMSGHANDLE PIC S9(18) BINARY.
**    New message handle
   15 MQPMO-NEWMSGHANDLE      PIC S9(18) BINARY.
**    The action being performed
   15 MQPMO-ACTION            PIC S9(9) BINARY.
**    Publish level
   15 MQPMO-PUBLEVEL          PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQPMO

```
dcl
 1 MQPMO based,
  3 StrucId           char(4),       /* Structure identifier */
  3 Version           fixed bin(31), /* Structure version number */
  3 Options           fixed bin(31), /* Options that control the action
                                        of MQPUT and MQPUT1 */
  3 Timeout           fixed bin(31), /* Reserved */
  3 Context           fixed bin(31), /* Object handle of input queue */
```

```
  3 KnownDestCount    fixed bin(31), /* Number of messages sent
                                        successfully to local queues */
  3 UnknownDestCount  fixed bin(31), /* Number of messages sent
                                        successfully to remote queues */
  3 InvalidDestCount  fixed bin(31), /* Number of messages that could
                                        not be sent */
  3 ResolvedQName     char(48),      /* Resolved name of destination
                                        queue */
  3 ResolvedQMgrName  char(48),      /* Resolved name of destination
                                        queue manager */
  3 RecsPresent       fixed bin(31), /* Number of put message records or
                                        response records present */
  3 PutMsgRecFields   fixed bin(31), /* Flags indicating which MQPMR
                                        fields are present */
  3 PutMsgRecOffset   fixed bin(31), /* Offset of first put message
                                        record from start of MQPMO */
  3 ResponseRecOffset fixed bin(31), /* Offset of first response record
                                        from start of MQPMO */
  3 PutMsgRecPtr      pointer,       /* Address of first put message
                                        record */
  3 ResponseRecPtr    pointer,       /* Address of first response
                                        record */
  3 OriginalMsgHandle fixed bin(63), /* Original message handle */
  3 NewMsgHandle      fixed bin(63); /* New message handle */
  3 Action            fixed bin(31); /* The action being performed */
  3 PubLevel          fixed bin(31); /* Publish level */
```

Dichiarazione High Level Assembler per MQPMO

```
MQPMO                   DSECT
MQPMO_STRUCID           DS   CL4   Structure identifier
MQPMO_VERSION           DS   F     Structure version number
MQPMO_OPTIONS           DS   F     Options that control the action of
*                                  MQPUT and MQPUT1
MQPMO_TIMEOUT           DS   F     Reserved
MQPMO_CONTEXT           DS   F     Object handle of input queue
MQPMO_KNOWNDESTCOUNT    DS   F     Number of messages sent successfully
*                                  to local queues
MQPMO_UNKNOWNDESTCOUNT  DS   F     Number of messages sent successfully
*                                  to remote queues
MQPMO_INVALIDDESTCOUNT  DS   F     Number of messages that could not be
*                                  sent
MQPMO_RESOLVEDQNAME     DS   CL48  Resolved name of destination queue
MQPMO_RESOLVEDQMGRNAME  DS   CL48  Resolved name of destination queue
*                                  manager
MQPMO_RECSPRESENT       DS   F     Number of put message records or
*                                  response records present
MQPMO_PUTMSGRECFIELDS   DS   F     Flags indicating which MQPMR
*                                  fields are present
MQPMO_PUTMSGRECOFFSET   DS   F     Offset of first put message record
*                                  from start of MQPMO
MQPMO_RESPONSERECOFFSET DS   F     Offset of first response record
*                                  from start of MQPMO
MQPMO_PUTMSGRECPTR      DS   F     Address of first put message
*                                  record
MQPMO_RESPONSERECPTR    DS   F     Address of first response record
MQPMO_ORIGINALMSGHANDLE DS   D     Original message handle
MQPMO_NEWMSGHANDLE      DS   D     New message handle
MQPMO_ACTION            DS   F     The action being performed
MQPMO_PUBLEVEL          DS   F     Publish level
*
MQPMO_LENGTH            EQU  *-MQPMO
                        ORG  MQPMO
MQPMO_AREA              DS   CL(MQPMO_LENGTH)
```

Dichiarazione Visual Basic per MQPMO

```
Type MQPMO
  StrucId           As String*4  'Structure identifier'
  Version           As Long      'Structure version number'
  Options           As Long      'Options that control the action of'
                                 'MQPUT and MQPUT1'
  Timeout           As Long      'Reserved'
  Context           As Long      'Object handle of input queue'
  KnownDestCount    As Long      'Number of messages sent successfully'
                                 'to local queues'
  UnknownDestCount  As Long      'Number of messages sent successfully'
```

```
                                'to remote queues'
  InvalidDestCount  As Long     'Number of messages that could not be'
                                'sent'
  ResolvedQName     As String*48 'Resolved name of destination queue'
  ResolvedQMgrName  As String*48 'Resolved name of destination queue'
                                'manager'
  RecsPresent       As Long     'Number of put message records or'
                                'response records present'
  PutMsgRecFields   As Long     'Flags indicating which MQPMR fields'
                                'are present'
  PutMsgRecOffset   As Long     'Offset of first put message record'
                                'from start of MQPMO'
  ResponseRecOffset As Long     'Offset of first response record from'
                                'start of MQPMO'
  PutMsgRecPtr      As MQPTR    'Address of first put message record'
  ResponseRecPtr    As MQPTR    'Address of first response record'
End Type
```

StrucId (MQCHAR4) per MQPM

Questo è l'identificativo della struttura delle opzioni del messaggio di inserimento. È sempre un campo di immissione. Il suo valore è MQPMO_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQPMO
Identificativo per la struttura delle opzioni del messaggio di inserimento.

Per il linguaggio di programmazione C, è definita anche la costante MQPMO_STRUC_ID_ARRAY. Ha lo stesso valore di MQPMO_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQPMO

Numero di versione della struttura.

Il valore deve essere uno dei seguenti.

MQPMO_VERSION_1
Version-1 struttura di opzioni put - message.

Questa versione è supportata in tutti gli ambienti.

MQPMO_VERSION_2
Version-2 struttura delle opzioni put - message.

Questa versione è supportata nei seguenti ambienti:

- AIX
- IBM i
- Linux
- Windows

e per IBM MQ MQI clients collegati a questi sistemi.

MQPMO_VERSION_3
Struttura delle opzioni Version-3 put - message.

Questa versione è supportata in tutti gli ambienti.

I campi esistenti solo nella versione più recente della struttura vengono identificati come tali nelle descrizioni dei campi. La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQPMO_CURRENT_
Versione corrente della struttura delle opzioni put - message.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQPMO_VERSION_1.

Opzioni (MQLONG) per MQPMO

Il campo Opzioni controlla l'operazione delle chiamate **MQPUT** e **MQPUT1** .

Opzione ambito È possibile specificare una o nessuna delle opzioni MQPMO. Per specificare più di un'opzione, aggiungere i valori insieme (non aggiungere la stessa costante più di una volta) o combinare i valori utilizzando l'operazione OR bit per bit (se il linguaggio di programmazione supporta le operazioni bit). Le combinazioni non valide sono annotate; tutte le altre combinazioni sono valide.

La seguente opzione controlla l'ambito delle pubblicazioni inviate:

MQPMO_SCOPE_QMGR

La pubblicazione viene inviata solo ai sottoscrittori che hanno sottoscritto questo gestore code. La pubblicazione non viene inoltrata ai gestori code di pubblicazione / sottoscrizione remoti che hanno effettuato una sottoscrizione a questo gestore code, che sovrascrive qualsiasi comportamento impostato utilizzando l'attributo dell'argomento PUBSCOPE.

Nota: Se non è impostato, l'ambito di pubblicazione è determinato dall'attributo dell'argomento PUBSCOPE.

Opzioni di pubblicazione. Le seguenti opzioni controllano il modo in cui i messaggi vengono pubblicati in un argomento:

MQPMO_SUPPRESS_REPLYTO

Tutte le informazioni specificate nei campi *ReplyToQ* e *ReplyToQMgr* di MQMD di questa pubblicazione non vengono trasmesse ai sottoscrittori. Se questa opzione viene utilizzata con un'opzione di report che richiede un *ReplyToQ*, la chiamata non riesce con MQRC_MISSING_REPLY_TO_Q.

MQPMO_RETAIN

La pubblicazione inviata deve essere conservata dal gestore code. Questa conservazione consente a un sottoscrittore di richiedere una copia di questa pubblicazione dopo l'ora in cui è stata pubblicata, utilizzando la chiamata MQSUBRQ. Consente inoltre l'invio di una pubblicazione alle applicazioni che effettuano la loro sottoscrizione dopo l'ora in cui è stata effettuata la pubblicazione (a meno che non scelgano di non inviarla utilizzando l'opzione MQSO_NEW_PUBLICATIONS_ONLY). Se un'applicazione riceve una pubblicazione che è stata conservata, viene indicata dalla proprietà del messaggio MQIsRetained di tale pubblicazione.

È possibile conservare solo una pubblicazione su ciascun nodo della struttura ad albero degli argomenti. Pertanto, se esiste già una pubblicazione conservata per questo argomento, pubblicata da qualsiasi altra applicazione, viene sostituita con questa pubblicazione. È quindi preferibile evitare che più di un editore conservi i messaggi sullo stesso argomento.

Quando le pubblicazioni conservate vengono richieste da un sottoscrittore, la sottoscrizione utilizzata potrebbe contenere un carattere jolly nell'argomento, nel qual caso un numero di pubblicazioni conservate potrebbe corrispondere (su vari nodi nella struttura ad albero dell'argomento) e diverse pubblicazioni potrebbero essere inviate all'applicazione richiedente. Consultare la descrizione della chiamata “MQSUBRQ - Richiesta di sottoscrizione” a pagina 815 per ulteriori dettagli.

Per informazioni su come le pubblicazioni conservate interagiscono con i livelli di sottoscrizione, consultare Intercettazione delle pubblicazioni.

Se viene utilizzata questa opzione e la pubblicazione non può essere conservata, il messaggio non viene pubblicato e la chiamata ha esito negativo con MQRC_PUT_NOT_USED.

MQPMO_NOT_OWN_SUBS
Indica al gestore code che l'applicazione non desidera inviare alcuna delle sue pubblicazioni alle sottoscrizioni di cui è proprietaria. Le sottoscrizioni sono considerate di proprietà della stessa applicazione se gli handle di connessione sono gli stessi.

MQPMO_WARN_IF_NO_SUBS_MATCHED
Se nessuna sottoscrizione corrisponde alla pubblicazione, restituire un codice di completamento (*CompCode*) di MQCC_WARNING e il codice motivo MQRC_NO_SUBS_MATCHED.

Se MQRC_NO_SUBS_MATCHED viene restituito dall'operazione di inserimento, la pubblicazione non è stata consegnata ad alcuna sottoscrizione. Tuttavia, se l'opzione MQPMO_RETAIN viene specificata nell'operazione di inserimento, il messaggio viene conservato e consegnato a qualsiasi sottoscrizione corrispondente definita successivamente.

Una sottoscrizione sull'argomento corrisponde alla pubblicazione se si verifica una delle condizioni seguenti:

- Il messaggio viene consegnato alla coda di sottoscrizione
- Il messaggio sarebbe stato recapitato alla coda di sottoscrizione, ma un problema con la coda significa che il messaggio non può essere inserito nella coda e di conseguenza è stato inserito nella coda di messaggi non recapitati o eliminato.
- È definita un'uscita di instradamento che sopprime la consegna del messaggio alla sottoscrizione

Una sottoscrizione sull'argomento non corrisponde alla pubblicazione se si verifica una delle condizioni riportate di seguito:

- La sottoscrizione ha una stringa di selezione che non corrisponde alla pubblicazione
- La sottoscrizione ha specificato l'opzione MQSO_PUBLICATION_ON_REQUEST
- La pubblicazione non è stata consegnata perché l'opzione MQPMO_NOT_OWN_SUBS è stata specificata sull'operazione di inserimento e la sottoscrizione corrisponde all'identità del publisher

Opzioni punto di sincronizzazione. Le seguenti opzioni si riferiscono alla partecipazione della chiamata MQPUT o MQPUT1 all'interno di un'unità di lavoro:

MQPMO_SYNCPOINT
La richiesta è di operare all'interno dei normali protocolli di unità di lavoro. Il messaggio non è visibile all'esterno dell'unità di lavoro fino a quando non viene eseguito il commit dell'unità di lavoro. Se viene eseguito il backout dell'unità di lavoro, il messaggio viene eliminato.

Se MQPMO_SYNCPOINT e MQPMO_NO_SYNCPOINT non vengono specificati, l'inclusione della richiesta di inserimento nei protocolli dell'unità di lavoro viene determinata dall'ambiente che esegue il gestore code e non dall'ambiente che esegue l'applicazione. Su z/OS, la richiesta di inserimento è all'interno di un'unità di lavoro. In tutti gli altri ambienti, la richiesta di inserimento non si trova all'interno di un'unità di lavoro.

A causa di queste differenze, un'applicazione che si desidera trasferire non deve consentire l'impostazione predefinita di questa opzione; specificare esplicitamente MQPMO_SYNCPOINT o MQPMO_NO_SYNCPOINT.

Non specificare MQPMO_SYNCPOINT con MQPMO_NO_SYNCPOINT.

MQPMO_NO_SYNCPOINT
La richiesta è di operare al di fuori dei normali protocolli di unità di lavoro. Il messaggio è disponibile immediatamente e non può essere eliminato ripristinando un'unità di lavoro.

Se MQPMO_NO_SYNCPOINT e MQPMO_SYNCPOINT non vengono specificati, l'inclusione della richiesta di inserimento nei protocolli dell'unità di lavoro viene determinata dall'ambiente che esegue il gestore code e non dall'ambiente che esegue l'applicazione. Su z/OS, la richiesta di inserimento è all'interno di un'unità di lavoro. In tutti gli altri ambienti, la richiesta di inserimento non si trova all'interno di un'unità di lavoro.

A causa di queste differenze, un'applicazione che si desidera trasferire non deve consentire l'impostazione predefinita di questa opzione; specificare esplicitamente MQPMO_SYNCPOINT o MQPMO_NO_SYNCPOINT.

Non specificare MQPMO_NO_SYNCPOINT con MQPMO_SYNCPOINT.

Opzioni identificativo messaggio e identificativo di correlazione. Le seguenti opzioni richiedono al gestore code di generare un nuovo identificativo del messaggio o un nuovo identificativo di correlazione:

ID_MQPMO_NEW_MSG_
Il gestore code sostituisce il contenuto del campo *MsgId* in MQMD con un nuovo ID messaggio. Questo identificativo del messaggio viene inviato con il messaggio e restituito all'applicazione all'output dalla chiamata MQPUT o MQPUT1 .

L'opzione MQPMO_NEW_MSG_ID può essere specificata anche quando il messaggio viene inserito in un elenco di distribuzione; per i dettagli, consultare la descrizione del campo *MsgId* nella struttura MQPMR.

L'utilizzo di questa opzione allevia l'applicazione della necessità di reimpostare il campo *MsgId* su MQMI_NONE prima di ogni richiamo MQPUT o MQPUT1 .

ID_CORREL_NEW_MQPMO_
Il gestore code sostituisce il contenuto del campo *CorrelId* in MQMD con un nuovo identificativo di correlazione. Questo identificativo di correlazione viene inviato con il messaggio e restituito all'applicazione all'output dalla chiamata MQPUT o MQPUT1 .

L'opzione MQPMO_NEW_CORREL_ID può essere specificata anche quando il messaggio viene inserito in un elenco di distribuzione; per i dettagli, consultare la descrizione del campo *CorrelId* nella struttura MQPMR.

MQPMO_NEW_CORREL_ID è utile nelle situazioni in cui l'applicazione richiede un identificativo di correlazione univoco.

Opzioni di gruppo e segmento. Le seguenti opzioni sono relative all'elaborazione dei messaggi in gruppi e segmenti di messaggi logici. Leggere le definizioni che seguono per comprendere l'opzione.

Attenzione: Non è possibile utilizzare messaggi segmentati o raggruppati con Pubblicazione / Sottoscrizione.

Messaggio fisico
È l'unità di informazioni più piccola che è possibile inserire o rimuovere da una coda; spesso corrisponde alle informazioni specificate o richiamate in una singola chiamata MQPUT, MQPUT1o MQGET. Ogni messaggio fisico ha il proprio descrittore di messaggio (MQMD). Generalmente, i messaggi fisici sono distinti da valori differenti per l'identificativo del messaggio (campo*MsgId* in MQMD), sebbene ciò non venga applicato dal gestore code.

Messaggio logico
Un messaggio logico è una singola unità di informazioni dell'applicazione solo per piattaforme non z/OS . In assenza di vincoli di sistema, un messaggio logico è uguale a un messaggio fisico. Ma quando i messaggi logici sono estremamente grandi, i vincoli di sistema potrebbero rendere consigliabile o necessario suddividere un messaggio logico in due o più messaggi fisici, denominati *segmenti*.

Un messaggio logico che è stato segmentato è costituito da due o più messaggi fisici che hanno lo stesso identificativo di gruppo non null (campo*GroupId* in MQMD) e lo stesso numero di sequenza del messaggio (campo*MsgSeqNumber* in MQMD). I segmenti sono distinti da valori differenti per l'offset del segmento (campo*Offset* in MQMD), che fornisce l'offset dei dati nel messaggio fisico dall'inizio dei dati nel messaggio logico. Poiché ogni segmento è un messaggio fisico, i segmenti in un messaggio logico di solito hanno identificativi di messaggio differenti.

Un messaggio logico che non è stato segmentato, ma per il quale la segmentazione è stata consentita dall'applicazione mittente, ha anche un identificativo di gruppo non null, sebbene in questo caso esista solo un messaggio fisico con tale identificativo di gruppo se il messaggio logico non appartiene a un gruppo di messaggi. I messaggi logici per i quali la segmentazione è stata inibita dall'applicazione di invio hanno un identificativo di gruppo null (MQGI_NONE), a meno che il messaggio logico non appartenga a un gruppo di messaggi.

Gruppo di messaggi
Un gruppo di messaggi è una serie di uno o più messaggi logici che hanno lo stesso identificativo di gruppo non null. I messaggi logici nel gruppo si distinguono per valori differenti per il numero di sequenza del messaggio, che è un numero intero compreso tra 1 e *n*, dove *n* è il numero di messaggi logici nel gruppo. Se uno o più messaggi logici sono segmentati, nel gruppo sono presenti più di *n* messaggi fisici.

ORDER MQPMO_LOGICAL_
Questa opzione indica al gestore code il modo in cui l'applicazione inserisce i messaggi in gruppi e segmenti di messaggi logici. Può essere specificato solo nella chiamata MQPUT; non è valido nella chiamata MQPUT1 .

Se MQPMO_LOGICAL_ORDER è specificato, indica che l'applicazione utilizza le chiamate MQPUT successive per:

1. Inserire i segmenti in ciascun segmento logico nell'ordine di offset crescente dei segmenti, a partire da 0, senza intervalli.
2. Inserire tutti i segmenti in un unico messaggio logico prima di inserirli nel successivo messaggio logico.
3. Inserire i messaggi logici in ciascun gruppo di messaggi nell'ordine crescente del numero di sequenza dei messaggi, a partire da 1, senza intervalli. IBM MQ aumenta il numero di sequenza dei messaggi automaticamente.
4. Inserire tutti i messaggi logici in un unico gruppo di messaggi prima di inserirli nel successivo gruppo di messaggi.

Per informazioni dettagliate su MQPMO_LOGICAL_ORDER, consultare Ordine logico e fisico

Opzioni di contesto Le opzioni seguenti controllano l'elaborazione del contesto del messaggio:

MQPMO_NO_CONTEXT
L'identità e il contesto di origine sono impostati in modo da non indicare alcun contesto. Ciò significa che i campi di contesto in MQMD sono impostati su:

- Spazi vuoti per i campi di caratteri
- Valori null per i campi byte
- Zeri per campi numerici

MQPMO_DEFAULT_CONTEXT
Il messaggio deve avere informazioni di contesto predefinite associate ad esso, sia per l'identità che per l'origine. Il gestore code imposta i campi di contesto nel descrittore del messaggio nel modo seguente:

Tabella 508. Valori predefiniti delle informazioni di contesto per i campi MQMD

Campo in MQMD	Valore utilizzato
UserIdentifier	Determinato dall'ambiente, se possibile; altrimenti, impostare su spazi vuoti.
AccountingToken	Determinato dall'ambiente, se possibile; altrimenti, impostare su MQACT_NONE.
ApplIdentityData	Impostare su spazi vuoti.
PutApplType	Determinato dall'ambiente.
PutApplName	Determinato dall'ambiente, se possibile; altrimenti, impostare su spazi vuoti.
PutDate	Impostare la data di inserimento del messaggio.
PutTime	Impostare l'ora di inserimento del messaggio.
ApplOriginData	Impostare su spazi vuoti.

Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio.

Questi sono i valori predefiniti e le azioni se non viene specificata alcuna opzione di contesto.

MQPMO_PASS_IDENTITY_CONTEXT
Il messaggio deve essere associato alle informazioni di contesto. Il contesto di identità viene preso dall'handle di coda specificato nel campo *Context* . Le informazioni sul contesto di origine vengono

create dal gestore code nello stesso modo in cui vengono generate per MQPMO_DEFAULT_CONTEXT (consultare la precedente tabella per i valori). Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio.

Per la chiamata MQPUT, la coda deve essere stata aperta con l'opzione MQOO_PASS_IDENTITY_CONTEXT (o un'opzione che la implica). Per la chiamata MQPUT1 , viene eseguito lo stesso controllo di autorizzazione della chiamata MQOPEN con l'opzione MQOO_PASS_IDENTITY_CONTEXT.

MQPMO_PASS_ALL_CONTEXT
Il messaggio deve essere associato alle informazioni di contesto. Il contesto viene preso dall'handle di coda specificato nel campo *Context* . Per ulteriori informazioni sul contesto del messaggio, consultare Controllo delle informazioni sul contesto.

Per la chiamata MQPUT, la coda deve essere stata aperta con l'opzione MQOO_PASS_ALL_CONTEXT (o un'opzione che la implica). Per la chiamata MQPUT1 , viene eseguito lo stesso controllo di autorizzazione della chiamata MQOPEN con l'opzione MQOO_PASS_ALL_CONTEXT.

MQPMO_SET_IDENTITY_CONTEXT
Il messaggio deve essere associato alle informazioni di contesto. L'applicazione specifica il contesto di identità nella struttura MQMD. Le informazioni sul contesto di origine vengono create dal gestore code nello stesso modo in cui vengono generate per MQPMO_DEFAULT_CONTEXT (consultare la precedente tabella per i valori). Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio.

Per la chiamata MQPUT, la coda deve essere stata aperta con l'opzione MQOO_SET_IDENTITY_CONTEXT (o un'opzione che la implica). Per la chiamata MQPUT1 , viene eseguito lo stesso controllo di autorizzazione della chiamata MQOPEN con l'opzione MQOO_SET_IDENTITY_CONTEXT.

MQPMO_SET_ALL_CONTEXT
Il messaggio deve essere associato alle informazioni di contesto. L'applicazione specifica l'identità, l'origine e il contesto utente nella struttura MQMD. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio.

Per la chiamata MQPUT, la coda deve essere stata aperta con l'opzione MQOO_SET_ALL_CONTEXT. Per la chiamata MQPUT1 , viene eseguito lo stesso controllo di autorizzazione della chiamata MQOPEN con l'opzione MQOO_SET_ALL_CONTEXT.

È possibile specificare solo una delle opzioni MQPMO_ * _CONTEXT. Se non si specifica alcun valore, viene utilizzato MQPMO_DEFAULT_CONTEXT.

Opzioni proprietà. La seguente opzione è correlata alle proprietà del messaggio:

MQPMO_MD_FOR_OUTPUT_ONLY
Il parametro del descrittore del messaggio deve essere utilizzato solo per l'output per restituire il descrittore del messaggio inserito. I campi descrittore del messaggio associati ai campi *NewMsgHandle*, *OriginalMsgHandle*o entrambi della struttura **MQPMO** devono essere utilizzati per l'input.

Se non viene fornito un handle del messaggio valido, la chiamata ha esito negativo con il codice di errore **MQRC_MD_ERROR**.

Inserisci opzioni di risposta. Le seguenti opzioni controllano la risposta restituita a una chiamata MQPUT o MQPUT1 . È possibile specificare solo una di queste opzioni. Se MQPMO_ASYNC_RESPONSE e MQPMO_SYNC_RESPONSE non sono specificati, viene assunto MQPMO_RESPONSE_AS_Q_DEF o MQPMO_RESPONSE_AS_TOPIC_DEF.

MQPMO_ASYNC_RESPONSE
L'opzione MQPMO_ASYNC_RESPONSE richiede il completamento di un'operazione MQPUT o MQPUT1 senza che l'applicazione attenda il completamento della chiamata da parte del gestore code. L'utilizzo di questa opzione può migliorare le prestazioni della messaggistica, in particolare per le applicazioni che utilizzano i collegamenti client. Un'applicazione può controllare periodicamente, utilizzando il comando MQSTAT, se si è verificato un errore durante le chiamate asincrone precedenti.

Con questa opzione, solo i seguenti campi sono garantiti per essere completati in MQMD;

- ApplIdentityData
- PutApplType
- PutApplName
- ApplOriginData

Inoltre, se una o entrambe le opzioni MQPMO_NEW_MSG_ID o MQPMO_NEW_CORREL_ID vengono specificate come opzioni, vengono restituiti anche MsgId e CorrelId . (MQPMO_NEW_MSG_ID può essere specificato implicitamente specificando un campo MsgId vuoto).

Vengono completati solo i precedenti campi specificati. Altre informazioni che normalmente vengono restituite nella struttura MQMD o MQPMO non sono definite.

Quando si richiede una risposta di inserimento asincrona per MQPUT1, i nomi ResolvedQName e ResolvedQMgrrestituiti nella struttura MQOD non sono definiti.

Quando si richiede una risposta di inserimento asincrono per MQPUT o MQPUT1, un CompCode e motivo di MQCC_OK e MQRC_NONE non significa necessariamente che il messaggio è stato inserito correttamente in una coda. Quando si sviluppa un'applicazione MQI che utilizza una risposta di inserimento asincrono e richiede la conferma che i messaggi sono stati inseriti in una coda, è necessario controllare sia i codici di errore CompCode che i codici di errore dalle operazioni di inserimento e utilizzare MQSTAT per eseguire la query delle informazioni di errore asincrone.

Anche se l'esito positivo o negativo di ogni singola chiamata MQPUT o MQPUT1 non può essere restituito immediatamente, il primo errore che si è verificato durante una chiamata asincrona può essere determinato in un secondo momento tramite una chiamata a MQSTAT.

Se un messaggio persistente nel punto di sincronizzazione non riesce ad essere recapitato utilizzando la risposta di inserimento asincrona e si tenta di eseguire il commit della transazione, il commit non riesce e la transazione viene ripristinata con un codice di completamento MQCC_FAILED e un motivo di MQRC_BACKED_OUT. L'applicazione può effettuare una chiamata a MQSTAT per determinare la causa di un errore precedente di MQPUT o MQPUT1 .

MQPMO_SYNC_RESPONSE
La specifica di questo tipo di risposta di inserimento assicura che l'operazione MQPUT o MQPUT1 venga sempre emessa in modo sincrono. Se l'operazione di inserimento ha esito positivo, vengono completati tutti i campi in MQMD e MQPMO.

Questa opzione garantisce una risposta sincrona indipendentemente dal valore di risposta di inserimento predefinito definito sulla coda o sull'oggetto argomento.

MQPMO_RESPONSE_AS_Q_DEF
Se questo valore viene specificato per una chiamata MQPUT, il tipo di risposta di inserimento utilizzato viene preso dal valore DEFPRESP specificato sulla coda quando è stato aperto per la prima volta dall'applicazione.

- Se la coda è una coda cluster e questo valore viene specificato per una chiamata MQPUT, il tipo di risposta di inserimento utilizzato viene ricavato dall'attributo **DEFPRESP** definito nel gestore code *di destinazione* che possiede la particolare istanza della coda in cui si trova il messaggio.

  Quando esistono più istanze della coda cluster e differiscono in questo attributo, viene selezionato il valore di uno di essi e non è possibile prevedere quale sarà utilizzato. È quindi necessario impostare questo attributo sullo stesso valore per tutte le istanze. In caso contrario, viene emesso il messaggio di errore AMQ9407 nei log del gestore code. Consultare anche How are destination object attributes resolved for aliases, remote and cluster queues?
- Se la coda non è una coda cluster e questo valore è specificato per una chiamata MQPUT, il tipo di risposta di inserimento utilizzato viene ricavato dall'attributo **DEFPRESP** definito nel gestore code *locale* , anche se il gestore code di destinazione è remoto.

Se un'applicazione client è connessa a un gestore code a un livello precedente a IBM WebSphere MQ 7.0, si comporta come se fosse stato specificato MQPMO_SYNC_RESPONSE.

Se questa opzione viene specificata per una chiamata MQPUT1 , il valore dell'attributo DEFPRESP non è noto prima che la richiesta venga inviata al server. Per impostazione predefinita, se la chiamata

MQPUT1 utilizza MQPMO_SYNCPOINT si comporta come per MQPMO_ASYNC_RESPONSE e se utilizza MQPMO_NO_SYNCPOINT si comporta come per MQPMO_SYNC_RESPONSE. Tuttavia, è possibile sovrascrivere questo comportamento predefinito impostando la proprietà Put1DefaultAlwaysSync nel file di configurazione del client; consultare Stanza CHANNELS del file di configurazione client.

MQPMO_RESPONSE_AS_TOPIC_DEF
MQPMO_RESPONSE_AS_TOPIC_DEF è un sinonimo di MQPMO_RESPONSE_AS_Q_DEF da utilizzare con gli oggetti argomento.

Altre opzioni. Le seguenti opzioni controllano il controllo dell'autorizzazione, cosa accade quando il gestore code è in quiesce e risolvono i nomi di code e gestori code:

MQPMO_ALTERNATE_USER_AUTHORITY
MQPMO_ALTERNATE_USER_AUTHORITY indica che il campo *AlternateUserId* del parametro **ObjDesc** della chiamata MQPUT1 contiene un identificativo utente che deve essere utilizzato per convalidare l'autorità di inserire i messaggi nella coda. La chiamata può avere esito positivo solo se *AlternateUserId* è autorizzato ad aprire la coda con le opzioni specificate, a prescindere dal fatto che l'identificativo utente con cui l'applicazione è in esecuzione sia autorizzato a farlo. (Ciò non si applica alle opzioni di contesto specificate, tuttavia, che vengono sempre controllate rispetto all'identificativo utente con cui è in esecuzione l'applicazione.)

Questa opzione è valida solo con la chiamata MQPUT1 .

MQPMO_FAIL_IF_QUIESCING
Questa opzione forza la chiamata MQPUT o MQPUT1 ad avere esito negativo se il gestore code è in stato di sospensione.

Su z/OS, questa opzione forza anche l'esito negativo della chiamata MQPUT o MQPUT1 se la connessione (per un'applicazione CICS o IMS) è in stato di sospensione.

La chiamata restituisce il codice di completamento MQCC_FAILED con codice motivo MQRC_Q_MGR_QUIESCING o MQRC_CONNECTION_QUIESCING.

MQPMO_RESOLVE_LOCAL_Q
Utilizzare questa opzione per riempire *ResolvedQName* nella struttura MQPMO con il nome della coda locale in cui viene inserito il messaggio e *ResolvedQMgrName* con il nome del gestore code locale che ospita la coda locale. Per ulteriori informazioni relative a MQPMO_RESOLVE_LOCAL_Q, consultare l'argomento MQOO_RESOLVE_LOCAL_Q.

Se si è autorizzati a inserire in una coda, si dispone dell'autorità richiesta per specificare questo indicatore nella chiamata MQPUT; non è necessaria alcuna autorizzazione speciale.

Opzione predefinita. Se non è necessaria alcuna delle opzioni descritte, utilizzare la seguente opzione:

MQPMO_NONE
Utilizzare questo valore per indicare che non sono state specificate altre opzioni; tutte le opzioni assumono i propri valori predefiniti. MQPMO_NONE è definito per aiutare la documentazione del programma; non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

MQPMO_NONE è un campo di input. Il valore iniziale del campo *Options* è MQPMO_NONE.

Timeout (MQLONG) per MQPMO

Questo è un campo riservato; il suo valore non è significativo. Il valore iniziale di questo campo è -1.

Contesto (MQHOBJ) per MQPMO

Se viene specificato MQPMO_PASS_IDENTITY_CONTEXT o MQPMO_PASS_ALL_CONTEXT, questo campo deve contenere l'handle della coda di input da cui vengono prese le informazioni di contesto da associare al messaggio inserito.

Se non viene specificato né MQPMO_PASS_IDENTITY_CONTEXT né MQPMO_PASS_ALL_CONTEXT, questo campo viene ignorato.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

Conteggio KnownDest(MQLONG) per MQPMO

Questo è il numero di messaggi che la chiamata MQPUT o MQPUT1 corrente ha inviato correttamente alle code nell'elenco di distribuzione che sono code locali. Il conteggio non include i messaggi inviati alle code che si risolvono in code remote (anche se una coda di trasmissione locale viene utilizzata inizialmente per memorizzare il messaggio). Questo campo viene impostato anche quando si immette un messaggio in una singola coda che non è in un elenco di distribuzione.

Questo è un campo di output. Il valore iniziale di questo campo è 0. Questo campo non è impostato se *Version* è minore di MQPMO_VERSION_1.

Questo campo non è definito in z/OS perché gli elenchi di distribuzione non sono supportati.

Conteggio UnknownDest(MQLONG) per MQPMO

Questo è il numero di messaggi che la chiamata MQPUT o MQPUT1 corrente ha inviato correttamente alle code nell'elenco di distribuzione che si risolvono in code remote. I messaggi che il gestore code conserva temporaneamente nel modulo dell'elenco di distribuzione vengono conteggiati come numero di singole destinazioni contenute in tali elenchi di distribuzione. Questo campo viene impostato anche quando si immette un messaggio in una singola coda che non è in un elenco di distribuzione.

Questo è un campo di output. Il valore iniziale di questo campo è 0. Questo campo non è impostato se *Version* è minore di MQPMO_VERSION_1.

Questo campo non è definito in z/OS perché gli elenchi di distribuzione non sono supportati.

Conteggio InvalidDest(MQLONG) per MQPMO

Questo è il numero di messaggi che non possono essere inviati alle code nell'elenco di distribuzione. Il conteggio include le code che non è stato possibile aprire, così come le code che sono state aperte correttamente ma per cui l'operazione di inserimento non è riuscita. Questo campo viene impostato anche quando si immette un messaggio in una singola coda che non è in un elenco di distribuzione.

Nota: Questo campo è impostato se il valore del parametro **CompCode** nella chiamata MQPUT o MQPUT1 è MQCC_OK o MQCC_WARNING; potrebbe essere impostato se il parametro **CompCode** è MQCC_FAILED, ma non si basa su questo nel codice dell'applicazione.

Questo è un campo di output. Il valore iniziale di questo campo è 0. Questo campo non è impostato se *Version* è minore di MQPMO_VERSION_1.

Questo campo non è definito in z/OS perché gli elenchi di distribuzione non sono supportati.

ResolvedQName (MQCHAR48) per MQPM

Questo è il nome della coda di destinazione dopo che la risoluzione del nome è stata eseguita dal gestore code locale. Il nome restituito è il nome di una coda che esiste sul gestore code identificato da *ResolvedQMgrName*.

Un valore non vuoto viene restituito solo se l'oggetto è una singola coda; se l'oggetto è un elenco di distribuzioni o un argomento, il valore restituito non è definito.

Questo è un campo di output. La lunghezza di questo campo è fornita da MQ_Q_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

ResolvedQMgr(MQCHAR48) per MQPMO

Questo è il nome del gestore code di destinazione dopo che la risoluzione dei nomi è stata eseguita dal gestore code locale. Il nome restituito è il nome del gestore code proprietario della coda identificata da *ResolvedQName*e può essere il nome del gestore code locale.

Se *ResolvedQName* è una coda condivisa di proprietà del gruppo di condivisione code a cui appartiene il gestore code locale, *ResolvedQMgrName* è il nome del gruppo di condivisione code. Se la coda è di proprietà di un altro gruppo di condivisione code, *ResolvedQName* può essere il nome del gruppo di condivisione code o il nome di un gestore code che è un membro del gruppo di condivisione code (la natura del valore restituito è determinata dalle definizioni di coda che esistono nel gestore code locale).

Un valore non vuoto viene restituito solo se l'oggetto è una singola coda; se l'oggetto è un elenco di distribuzioni o un argomento, il valore restituito non è definito.

Questo è un campo di output. La lunghezza di questo campo viene fornita da MQ_Q_MGR_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

RecsPresent (MQLONG) per MQPMO

Questo è il numero di record di messaggi di inserimento MQPMR o di risposta MQRR forniti dall'applicazione. Questo numero può essere maggiore di zero solo se il messaggio viene inserito in un elenco di distribuzione. I record di messaggi di inserimento e i record di risposta sono facoltativi; l'applicazione non deve fornire alcun record oppure può scegliere di fornire record di un solo tipo. Tuttavia, se l'applicazione fornisce record di entrambi i tipi, deve fornire record *RecsPresent* di ciascun tipo.

Il valore di *RecsPresent* non deve necessariamente essere uguale al numero di destinazioni nell'elenco di distribuzione. Se vengono forniti troppi record, l'eccesso non viene utilizzato; se viene fornito un numero troppo basso di record, vengono utilizzati i valori predefiniti per le proprietà del messaggio per le destinazioni che non hanno inserito i record del messaggio (consultare *PutMsgRecOffset*).

Se *RecsPresent* è minore di zero o è maggiore di zero ma il messaggio non viene inserito in un elenco di distribuzione, la chiamata ha esito negativo con codice di errore MQRC_RECS_PRESENT_ERROR.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è inferiore a MQPMO_VERSION_2.

PutMsgRecFields (MQLONG) per MQPMO

Questo campo contiene indicatori che indicano quali campi MQPMR sono presenti nei record del messaggio di inserimento forniti dall'applicazione. Utilizzare *PutMsgRecFields* solo quando il messaggio viene inserito in un elenco di distribuzione. Il campo viene ignorato se *RecsPresent* è zero o se entrambi *PutMsgRecOffset* e *PutMsgRecPtr* sono zero.

Per i campi presenti, il gestore code utilizza per ogni destinazione i valori dei campi nel corrispondente record del messaggio di inserimento. Per i campi assenti, il gestore code utilizza i valori della struttura MQMD.

Utilizzare uno o più dei seguenti indicatori per indicare quali campi sono presenti nei record dei messaggi di inserimento:

ID_MSG_MQPMRF
Il campo identificativo messaggio è presente.

ID CORREL_MQPMRF
Il campo Identificativo correlazione è presente.

ID_GROUP_MQPMRF
Il campo identificativo gruppo è presente.

MQPMRF_FEEDBACK
Il campo Feedback è presente.

MQPMRF_ACCOUNTING_TOKEN
Il campo Accounting - token è presente.

Se si specifica questo indicatore, specificare MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT nel campo *Options* ; se questa condizione non viene soddisfatta, la chiamata ha esito negativo con codice motivo MQRC_PMO_RECORD_FLAGS_ERROR.

Se non è presente alcun campo MQPMR, è possibile specificare quanto segue:

MQPMRF_NONE
Non sono presenti campi record di messaggi di immissione.

Se questo valore viene specificato, *RecsPresent* deve essere zero o entrambi *PutMsgRecOffset* e *PutMsgRecPtr* devono essere zero.

MQPMRF_NONE è definito per aiutare la documentazione del programma. Non è previsto che questa costante venga utilizzata con altre, ma poiché il valore è zero, tale utilizzo non può essere rilevato.

Se *PutMsgRecFields* contiene indicatori non validi o se vengono forniti record di messaggi di inserimento, ma *PutMsgRecFields* ha il valore MQPMRF_NONE, la chiamata ha esito negativo con codice motivo MQRC_PMO_RECORD_FLAGS_ERROR.

Questo è un campo di immissione. Il valore iniziale di questo campo è MQPMRF_NONE. Questo campo viene ignorato se *Version* è inferiore a MQPMO_VERSION_2.

PutMsgRecOffset (MQLONG) per MQPM

Questo è l'offset in byte del primo record del messaggio di inserimento MQPMR dall'inizio della struttura MQPMO. L'offset può essere positivo o negativo. *PutMsgRecOffset* viene utilizzato solo quando il messaggio viene inserito in un elenco di distribuzione. Il campo viene ignorato se *RecsPresent* è zero.

Quando il messaggio viene inserito in un elenco di distribuzione, è possibile fornire un array di uno o più record di messaggi di inserimento MQPMR per specificare determinate proprietà del messaggio per ciascuna destinazione singolarmente; queste proprietà sono:

- ID messaggio
- Identificativo di correlazione
- ID gruppo
- Valore di feedback
- Token account

Non è necessario specificare tutte queste proprietà, ma qualsiasi sottoinsieme si scelga, specificare i campi nell'ordine corretto. Per ulteriori dettagli, consultare la descrizione della struttura MQPMR.

Di solito, ci devono essere tanti record di messaggi put quanti sono i record di oggetti specificati da MQOD quando viene aperto l'elenco di distribuzione; ogni record di messaggi put fornisce le proprietà del messaggio per la coda identificata dal record di oggetto corrispondente. Le code nell'elenco di distribuzione che non riescono ad aprirsi devono ancora avere i record dei messaggi assegnati nelle posizioni appropriate nella schiera, anche se in questo caso le proprietà del messaggio vengono ignorate.

Il numero di record di messaggi di inserimento può essere diverso dal numero di record di oggetti. Se il numero di record dei messaggi di inserimento è inferiore a quello dei record degli oggetti, le proprietà dei messaggi per le destinazioni che non dispongono di record dei messaggi di inserimento vengono prese dai campi corrispondenti nel descrittore dei messaggi MQMD. Se ci sono più record di messaggi di inserimento che record di oggetti, gli eccessi non vengono utilizzati (anche se deve essere ancora possibile accedervi). I record dei messaggi di inserimento sono facoltativi, ma se vengono forniti devono essere *RecsPresent* .

Fornire i record del messaggio di inserimento in modo simile ai record oggetto in MQOD, specificando un offset in *PutMsgRecOffset*o specificando un indirizzo in *PutMsgRecPtr* ; per i dettagli su come eseguire questa operazione, consultare il campo *ObjectRecOffset* descritto in"MQOD - Descrittore oggetto" a pagina 492.

Non è possibile utilizzare più di uno tra *PutMsgRecOffset* e *PutMsgRecPtr* ; la chiamata ha esito negativo con codice motivo MQRC_PUT_MSG_RECORDS_ERROR se entrambi sono diversi da zero.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è inferiore a MQPMO_VERSION_2.

Offset ResponseRec(MQLONG) per MQPMO

Questo è l'offset in byte del primo record di risposta MQRR dall'inizio della struttura MQPMO. L'offset può essere positivo o negativo. *ResponseRecOffset* viene utilizzato solo quando il messaggio viene inserito in un elenco di distribuzione. Il campo viene ignorato se *RecsPresent* è zero.

Quando si inserisce il messaggio in un elenco di distribuzione, è possibile fornire un array di uno o più record di risposta MQRR per identificare le code a cui il messaggio non è stato inviato correttamente (campo*CompCode* in MQRR) e il motivo di ciascun errore (campo*Reason* in MQRR). Il messaggio potrebbe

non essere stato inviato perché non è stato possibile aprire la coda o perché l'operazione di inserimento non è riuscita. Il gestore code imposta i record di risposta solo quando il risultato della chiamata è misto (ovvero, alcuni messaggi sono stati inviati correttamente mentre altri hanno avuto esito negativo o tutti hanno avuto esito negativo ma per motivi diversi); il codice motivo MQRC_MULTIPLE_REASON dalla chiamata indica questo caso. Se lo stesso codice di errore si applica a tutte le code, tale motivo viene restituito nel parametro **Reason** della chiamata MQPUT o MQPUT1 e i record di risposta non vengono impostati.

Di solito, ci sono tanti record di risposta quanti sono i record di oggetto specificati da MQOD quando viene aperto l'elenco di distribuzione; quando necessario, ogni record di risposta viene impostato sul codice di completamento e sul codice motivo per l'inserimento nella coda identificato dal record di oggetto corrispondente. Le code nell'elenco di distribuzione che non riescono ad aprirsi devono avere ancora i record di risposta assegnati nelle posizioni appropriate nella schiera, anche se sono impostati sul codice di completamento e sul codice di errore risultanti dall'operazione di apertura, piuttosto che sull'operazione di inserimento.

Il numero di record di risposta può essere diverso dal numero di record oggetto. Se ci sono meno record di risposta rispetto ai record oggetto, l'applicazione potrebbe non essere in grado di identificare tutte le destinazioni per cui l'operazione di inserimento non è riuscita o le cause degli errori. Se ci sono più record di risposta che record di oggetto, gli eccessi non vengono utilizzati (anche se deve essere ancora possibile accedervi). I record di risposta sono facoltativi, ma se vengono forniti devono essere *RecsPresent* .

Fornire i record di risposta in modo simile ai record di oggetto in MQOD, specificando un offset in *ResponseRecOffset*o specificando un indirizzo in *ResponseRecPtr* ; per i dettagli su come eseguire questa operazione, consultare il campo *ObjectRecOffset* descritto in "MQOD - Descrittore oggetto" a pagina 492. Tuttavia, non utilizzare più di uno tra *ResponseRecOffset* e *ResponseRecPtr* ; la chiamata ha esito negativo con codice motivo MQRC_RESPONSE_RECORDS_ERROR se entrambi sono diversi da zero.

Per la chiamata MQPUT1 , questo campo deve essere zero. Ciò è dovuto al fatto che le informazioni di risposta (se richieste) vengono restituite nei record di risposta specificati dal descrittore dell'oggetto MQOD.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è inferiore a MQPMO_VERSION_2.

PutMsgRecPtr (MQPTR) per MQPM

Questo è l'indirizzo del primo record del messaggio di inserimento MQPMR. Utilizzare *PutMsgRecPtr* solo quando il messaggio viene inserito in un elenco di distribuzione. Il campo viene ignorato se *RecsPresent* è zero.

È possibile utilizzare *PutMsgRecPtr* o *PutMsgRecOffset* per specificare i record dei messaggi di inserimento, ma non entrambi; per i dettagli, consultare "PutMsgRecOffset (MQLONG) per MQPM" a pagina 527. Se non si utilizza *PutMsgRecPtr*, impostarlo sul puntatore null o sui byte null.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e, in caso contrario, una stringa di byte completamente null. Questo campo viene ignorato se *Version* è inferiore a MQPMO_VERSION_2.

Nota: Sulle piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata, con il valore iniziale che è la stringa di byte null.

ResponseRecPtr (MQPTR) per MQPMO

Questo è l'indirizzo del primo record di risposta MQRR. *ResponseRecPtr* viene utilizzato solo quando il messaggio viene inserito in un elenco di distribuzione. Il campo viene ignorato se *RecsPresent* è zero.

Utilizzare *ResponseRecPtr* o *ResponseRecOffset* per specificare i record di risposta, ma non entrambi; per i dettagli, vedere "Offset ResponseRec(MQLONG) per MQPMO" a pagina 527. Se non si utilizza *ResponseRecPtr* , impostarlo sul puntatore null o sui byte null.

Per la chiamata MQPUT1 , questo campo deve essere il puntatore null o byte null. Ciò è dovuto al fatto che le informazioni di risposta (se richieste) vengono restituite nei record di risposta specificati dal descrittore dell'oggetto MQOD.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e, in caso contrario, una stringa di byte completamente null. Questo campo viene ignorato se *Version* è inferiore a MQPMO_VERSION_2.

Nota: Sulle piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata, con il valore iniziale che è la stringa di byte null.

Handle OriginalMsg(MQHMSG) per MQPMO

Questo è un handle facoltativo per un messaggio. Potrebbe essere stato precedentemente richiamato da una coda. L'utilizzo di questo handle è soggetto al valore del campo *Action* ; consultare anche NewMsgHandle.

Il contenuto dell'handle del messaggio originale non verrà modificato dalla chiamata **MQPUT** o **MQPUT1** .

Questo è un campo di immissione. Il valore iniziale di questo campo è **MQHM_NONE**. Questo campo viene ignorato se la versione è inferiore a **MQPMO_VERSION_3**.

NewMsgHandle (MQHMSG) per MQPMO

Si tratta di un handle facoltativo per il messaggio inserito in base al valore del campo Azione. Definisce le proprietà del messaggio e sovrascrive i valori di *OriginalMsgHandle*, se specificati.

Al ritorno dalla chiamata **MQPUT** o **MQPUT1** , il contenuto dell'handle riflette il messaggio effettivamente inserito.

Questo è un campo di immissione. Il valore iniziale di questo campo è **MQHM_NONE**. Questo campo viene ignorato se la versione è inferiore a **MQPMO_VERSION_3**.

Azione (MQLONG) per MQPMO

Specifica il tipo di inserimento eseguito e la relazione tra il messaggio originale specificato dal campo Handle OriginalMsge il nuovo messaggio specificato dal campo Handle NewMsg. Le proprietà del messaggio vengono scelte dal gestore code in base al valore dell'azione specificata.

È possibile scegliere di fornire il contenuto del descrittore del messaggio utilizzando il parametro MsgDesc nelle chiamate MQPUT o MQPUT1 . In alternativa, è possibile non fornire il parametro MsgDesc o specificare che è di sola emissione includendo MQPMO_MD_FOR_OUTPUT_ONLY nel campo Opzioni della struttura MQPMO.

Se il parametro MsgDesc non viene fornito o se è specificato per essere di sola emissione, il descrittore del messaggio per il nuovo messaggio viene popolato dai campi degli handle del messaggio di MQPMO, in base alle regole descritte in questo argomento.

L'impostazione del contesto e le attività di passaggio descritte in Controllo delle informazioni di contesto diventano effettive dopo la composizione del descrittore del messaggio.

Se viene specificato un valore di azione non corretto, la chiamata ha esito negativo con il codice motivo MQRC_ACTION_ERROR.

È possibile specificare una delle azioni riportate di seguito:

NUOVO
Viene inserito un nuovo messaggio e non viene specificata alcuna relazione con un messaggio precedente dal programma. Il descrittore del messaggio è composto come segue:

- Se viene fornito un MsgDesc sulla chiamata MQPUT o MQPUT1 e MQPMO_MD_FOR_OUTPUT_ONLY non si trova in MQPMO.Options, viene utilizzato come descrittore del messaggio non modificato.

- Se non viene fornito un MsgDesc o MQPMO_MD_FOR_OUTPUT_ONLY si trova in MQPMO.Options quindi il gestore code genera il descrittore del messaggio utilizzando una combinazione di proprietà dall'handle OriginalMsge dall'handle NewMsg. Tutti i campi del descrittore del messaggio esplicitamente impostati sul nuovo handle del messaggio hanno la precedenza su quelli del gestore del messaggio originale.

I dati del messaggio vengono presi dal parametro del buffer MQPUT o MQPUT1 .

FORWARD MQACTP

Un messaggio precedentemente richiamato è stato inoltrato. L'handle del messaggio originale specifica il messaggio che è stato precedentemente richiamato.

Il nuovo handle del messaggio specifica le modifiche alle proprietà (incluse quelle nel descrittore del messaggio) nell'handle del messaggio originale.

Il descrittore del messaggio è composto come segue:

- Se viene fornito un MsgDesc sulla chiamata MQPUT o MQPUT1 e MQPMO_MD_FOR_OUTPUT_ONLY non si trova in MQPMO.Options, viene utilizzato come descrittore del messaggio non modificato.
- Se non viene fornito un MsgDesc o MQPMO_MD_FOR_OUTPUT_ONLY si trova in MQPMO.Options quindi il gestore code genera il descrittore del messaggio utilizzando una combinazione di proprietà dall'handle OriginalMsge dall'handle NewMsg. Tutti i campi del descrittore del messaggio esplicitamente impostati sul nuovo handle del messaggio hanno la precedenza su quelli del gestore del messaggio originale.
- Se MQPMO_NEW_MSG_ID o MQPMO_NEW_CORREL_ID sono specificati in MQPMO.Options, quindi vengono onorate.

Le proprietà del messaggio sono composte come segue:

- Tutte le proprietà dell'handle del messaggio originale che hanno MQCOPY_FORWARD in MQPD.CopyOptions
- Tutte le proprietà dal nuovo handle del messaggio. Per ogni proprietà nel nuovo handle del messaggio che ha lo stesso nome di una proprietà nell'handle del messaggio originale, il valore viene preso dal nuovo handle del messaggio. L'unica eccezione a questa regola è il caso speciale quando la proprietà nel nuovo handle del messaggio ha lo stesso nome di una proprietà nell'handle del messaggio originale, ma il valore della proprietà è null. In questo caso la proprietà viene eliminata dal messaggio.

I dati del messaggio da inoltrare vengono presi dal parametro del buffer MQPUT o MQPUT1 .

MQACTP_REPLY

Si sta effettuando una risposta ad un messaggio precedentemente richiamato. L'handle del messaggio originale specifica il messaggio che è stato precedentemente richiamato.

Il nuovo handle del messaggio specifica le modifiche alle proprietà (incluse quelle nel descrittore del messaggio) nell'handle del messaggio originale.

Il descrittore del messaggio è composto come segue:

- Se viene fornito un MsgDesc sulla chiamata MQPUT o MQPUT1 e MQPMO_MD_FOR_OUTPUT_ONLY non si trova in MQPMO.Options, viene utilizzato come descrittore del messaggio non modificato.
- Se non viene fornito un MsgDesc o MQPMO_MD_FOR_OUTPUT_ONLY si trova in MQPMO.Options, quindi i campi del descrittore del messaggio iniziale vengono scelti come segue:

Tabella 509. Trasformazione dell'handle del messaggio di risposta

Campo in MQMD	**Valore utilizzato**
Prospetto	Se MQRO_PASS_DISCARD_AND_SCADENZA e MQRO_DISCARD_MSG sono impostate: MQRO_DISCARD_MSG altrimenti MQRO_NONE
MsgType	MQMT_REPLY
Scadenza	Se MQRO_PASS_DISCARD_AND_SCADENZA è impostato: Copiato dal messaggio di input altrimenti MQEI_UNLIMITED
Feedback	MQFB_NONE
MsgId	Se MQPMO_NEW_MSG_ID è impostato: Viene creato un nuovo identificativo di messaggio oppure se MQRO_PASS_MSG_ID è impostato: Copiato dal messaggio di input altrimenti MQMI_NONE
CorrelId	Se MQPMO_NEW_CORREL_ID è impostato: Viene generato un nuovo identificatore di correlazione else se MQRO_COPY_MSG_ID_TO_CORREL_ID è impostato: Copiato dal campo MsgId del messaggio di input oppure se MQRO_PASS_CORREL_ID è impostato: Copiato dal campo CorrelId del messaggio di input altrimenti MQCI_NONE
BackoutCount	0
ReplyToQ	Spazi
ReplyToQMgr	Spazi
GroupId	MQGI_NONE
MsgSeqNumber	1
Offset	0
MsgFlags	MQMF_NONE
OriginalLength	MQOL_NON DEFINITO

- Il descrittore del messaggio viene quindi modificato dal nuovo handle del messaggio - qualsiasi campo descrittore del messaggio esplicitamente impostato come proprietà nel nuovo handle del messaggio ha la precedenza sui campi descrittore del messaggio come descritto in precedenza.

Le proprietà del messaggio sono composte come segue:

- Tutte le proprietà dall'handle del messaggio originale che hanno MQCOPY_REPLY in MQPD.CopyOptions
- Tutte le proprietà dal nuovo handle del messaggio. Per ogni proprietà nel nuovo handle del messaggio che ha lo stesso nome di una proprietà nell'handle del messaggio originale, il valore viene preso dal nuovo handle del messaggio. L'unica eccezione a questa regola è il caso speciale quando la proprietà nel nuovo handle del messaggio ha lo stesso nome di una proprietà nell'handle del messaggio originale, ma il valore della proprietà è null. In questo caso la proprietà viene eliminata dal messaggio.

I dati del messaggio da inoltrare vengono presi dal parametro del buffer MQPUT/MQPUT1 .

PROSPETTO MQACTP_REPORT

È in corso la creazione di un report come risultato di un messaggio precedentemente richiamato. L'handle del messaggio originale specifica il messaggio che causa la creazione del prospetto.

Il nuovo handle del messaggio specifica le modifiche alle proprietà (incluse quelle nel descrittore del messaggio) nell'handle del messaggio originale.

Il descrittore del messaggio è composto come segue:

- Se viene fornito un MsgDesc sulla chiamata MQPUT o MQPUT1 e MQPMO_MD_FOR_OUTPUT_ONLY non si trova in MQPMO.Options, viene utilizzato come descrittore del messaggio non modificato.
- Se non viene fornito un MsgDesc o MQPMO_MD_FOR_OUTPUT_ONLY si trova in MQPMO.Options e i campi del descrittore del messaggio iniziale vengono scelti come segue:

Tabella 510. Trasformazione dell'handle del messaggio di report

Campo in MQMD	**Valore utilizzato**
Prospetto	Se MQRO_PASS_DISCARD_AND_SCADENZA e MQRO_DISCARD_MSG sono impostati: MQRO_DISCARD_MSG altrimenti MQRO_NONE
MsgType	REPORT MQMT
Scadenza	Se MQRO_PASS_DISCARD_AND_SCADENZA è impostato: Copiato dal messaggio di input altrimenti MQEI_UNLIMITED
MsgId	Se MQPMO_NEW_MSG_ID è impostato: Viene creato un nuovo identificativo di messaggio oppure se MQRO_PASS_MSG_ID è impostato: Copiato dal messaggio di input altrimenti MQMI_NONE

Tabella 510. Trasformazione dell'handle del messaggio di report (Continua)	
Campo in MQMD	**Valore utilizzato**
CorrelId	Se MQPMO_NEW_CORREL_ID è impostato: Viene generato un nuovo identificatore di correlazione else se MQRO_COPY_MSG_ID_TO_CORREL_ID è impostato: Copiato dal campo MsgId del messaggio di input oppure se MQRO_PASS_CORREL_ID è impostato: Copiato dal campo CorrelId del messaggio di input altrimenti MQCI_NONE
BackoutCount	0
ReplyToQ	Spazi
ReplyToQMgr	Spazi
OriginalLength	Impostare su *BufferLength*

- Il descrittore del messaggio viene quindi modificato dal nuovo handle del messaggio - qualsiasi campo descrittore del messaggio esplicitamente impostato come proprietà nel nuovo handle del messaggio ha la precedenza sui campi descrittore del messaggio come descritto in precedenza.

Le proprietà del messaggio sono composte come segue:

- Tutte le proprietà dall'handle del messaggio originale che hanno MQCOPY_REPORT in MQPD.CopyOptions
- Tutte le proprietà dal nuovo handle del messaggio. Per ogni proprietà nel nuovo handle del messaggio che ha lo stesso nome di una proprietà nell'handle del messaggio originale, il valore viene preso dal nuovo handle del messaggio. L'unica eccezione a questa regola è il caso speciale quando la proprietà nel nuovo handle del messaggio ha lo stesso nome di una proprietà nell'handle del messaggio originale, ma il valore della proprietà è null. In questo caso la proprietà viene eliminata dal messaggio.

Il campo Feedback nell'MQMD risultante rappresenta il report che deve essere generato. Un valore di Feedback di MQFB_NONE causa l'esito negativo della chiamata MQPUT o MQPUT1 con codice motivo MQRC_FEEDBACK_ERROR.

Per scegliere i dati utente del messaggio di report, IBM MQ consulta i campi Report e Feedback nell'MQMD risultante e i parametri Buffer e BufferLength della chiamata MQPUT o MQPUT1 .

- Se il feedback è MQFB_COA, MQFB_COD o MQFB_EXPIRATION, il valore del report viene esaminato.
- Se uno dei seguenti casi è true, vengono utilizzati i dati completi del messaggio dal buffer per una lunghezza di BufferLength .
  - Il feedback è MQFB_EXPIRATION e il report contiene MQRO_EXPIRATION_WITH_FULL_DATA
  - Il feedback è MQFB_COD e il report contiene MQRO_COD_WITH_FULL_DATA
  - Il feedback è MQFB_COA e il report contiene MQRO_COA_WITH_FULL_DATA
- Se uno dei seguenti casi è true, vengono utilizzati i primi 100 byte del messaggio (o BufferLength se è inferiore a 100) dal buffer
  - Il feedback è MQFB_EXPIRATION e il report contiene MQRO_EXPIRATION_WITH_DATA
  - Il feedback è MQFB_COD e il report contiene MQRO_COD_WITH_DATA
  - Il feedback è MQFB_COA e il report contiene MQRO_COA_WITH_DATA

- Se il Feedback è MQFB_EXPIRATION, MQFB_COD o MQFB_COA e il report non contiene le opzioni * _WITH_FULL_DATA o * _WITH_DATA relative a tale valore di Feeback, nessun dato utente viene incluso nel messaggio.
- Se Feedback assume un valore diverso da quelli elencati in precedenza, Buffer e BufferLength vengono utilizzati normalmente.

La derivazione dei dati utente descritta nell'elenco precedente è riportata anche nella tabella seguente:

Tabella 511. Origine dei dati utente

	COA MQFB	**COD MQFB**	**SCADENZA_MQFB**
MQRO_EXPIRATION _WITH_FULL_DATA	Nessuna	Nessuna	Buffer (lunghezza buffer)
MQRO_COD_CON _FULL_DATA	Nessuna	Buffer (lunghezza buffer)	Nessuna
MQRO_COA_WITH _FULL_DATA	Buffer (lunghezza buffer)	Nessuna	Nessuna
MQRO_EXPIRATION _WITH_DATA	Nessuna	Nessuna	Buffer (primi 100 byte)
DATI MQRO_COD_WITH_	Nessuna	Buffer (primi 100 byte)	Nessuna
DATA_COA_WITH_MQ RO	Buffer (primi 100 byte)	Nessuna	Nessuna

PubLevel (MQLONG) per MQPMO

Il valore iniziale di questo campo è 9. Il livello di sottoscrizione indicato da questa pubblicazione. Solo le sottoscrizioni con il SubLevel più alto minore o uguale a questo valore ricevono questa pubblicazione. Questo valore deve essere compreso tra zero e 9; zero è il livello più basso. Tuttavia, se una pubblicazione è stata conservata, non è più disponibile per i sottoscrittori a livelli superiori perché viene ripubblicata in PubLevel 1.

Per informazioni, consultare Intercettazione delle pubblicazioni.

MQPMR - Record messaggio di inserimento

Utilizzare la struttura MQPMR per specificare varie proprietà del messaggio per una singola destinazione durante l'inserimento di un messaggio in un elenco di distribuzione. MQPMR è una struttura di input / output per le chiamate MQPUT e MQPUT1 .

Disponibilità

La struttura MQPMR è disponibile sulle seguenti piattaforme:

- AIX
- IBM i
- Linux
- Windows

e per i client IBM MQ connessi a questi sistemi.

Serie di caratteri e codifica

I dati in MQPMR devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da MQENC_NATIVE. Tuttavia, se l'applicazione è in esecuzione come client MQ , la struttura deve essere nella serie di caratteri e nella codifica del client.

Utilizzo

Fornendo un array di queste strutture nella chiamata MQPUT o MQPUT1 , è possibile specificare valori differenti per ogni coda di destinazione in un elenco di distribuzione. Alcuni dei campi sono solo di immissione, altri sono di immissione / emissione.

Nota: Questa struttura è insolita in quanto non ha un layout fisso. I campi in questa struttura sono facoltativi e la presenza o l'assenza di ciascun campo è indicata dagli indicatori nel campo *PutMsgRecFields* in MQPMO. I campi presenti in ***devono essere nel seguente ordine*** :

- *MsgId*
- *CorrelId*
- *GroupId*
- *Feedback*
- *AccountingToken*

I campi assenti non occupano spazio nel record.

Poiché MQPMR non dispone di un layout fisso, non ne viene fornita alcuna definizione nei file di intestazione, COPY e INCLUDE per i linguaggi di programmazione supportati. Il programmatore dell'applicazione deve creare una dichiarazione contenente i campi richiesti dall'applicazione e impostare gli indicatori in *PutMsgRecFields* per indicare i campi presenti.

Campi

Non vi sono valori iniziali definiti per questa struttura, poiché non viene fornita alcuna dichiarazione di struttura nei file di intestazione COPYe INCLUDE per i linguaggi di programmazione supportati. Le dichiarazioni di esempio mostrano come dichiarare la struttura se tutti i campi sono obbligatori.

Tabella 512. Campi in MQPMR

Nome campo	Descrizione campo
MsgId	ID messaggio
CorrelId	Identificativo di correlazione
GroupId	ID gruppo
Feedback	Feedback o codice motivo
AccountingToken	Token account

Dichiarazioni di lingua

Dichiarazione C per MQPMR

```
typedef struct tagMQPMR MQPMR;
struct tagMQPMR {
  MQBYTE24  MsgId;            /* Message identifier */
  MQBYTE24  CorrelId;         /* Correlation identifier */
  MQBYTE24  GroupId;          /* Group identifier */
  MQLONG    Feedback;         /* Feedback or reason code */
  MQBYTE32  AccountingToken;  /* Accounting token */
};
```

Dichiarazione COBOL per MQPMR

```
**   MQPMR structure
  10 MQPMR.
**    Message identifier
   15 MQPMR-MSGID           PIC X(24).
**    Correlation identifier
   15 MQPMR-CORRELID        PIC X(24).
**    Group identifier
   15 MQPMR-GROUPID         PIC X(24).
**    Feedback or reason code
   15 MQPMR-FEEDBACK        PIC S9(9) BINARY.
**    Accounting token
   15 MQPMR-ACCOUNTINGTOKEN PIC X(32).
```

Dichiarazione PL/I per MQPMR

```
dcl
 1 MQPMR based,
   3 MsgId           char(24),      /* Message identifier */
   3 CorrelId        char(24),      /* Correlation identifier */
   3 GroupId         char(24),      /* Group identifier */
   3 Feedback        fixed bin(31), /* Feedback or reason code */
   3 AccountingToken char(32);      /* Accounting token */
```

Dichiarazione Visual Basic per MQPMR

```
Type MQPMR
  MsgId           As MQBYTE24 'Message identifier'
  CorrelId        As MQBYTE24 'Correlation identifier'
  GroupId         As MQBYTE24 'Group identifier'
  Feedback        As Long     'Feedback or reason code'
  AccountingToken As MQBYTE32 'Accounting token'
End Type
```

MsgId (MQBYTE24) per MQPMR

Questo è l'identificativo del messaggio da utilizzare per il messaggio inviato alla coda con un nome specificato dall'elemento corrispondente nell'array di strutture MQOR fornito nella chiamata MQOPEN o MQPUT1 . Viene elaborato nello stesso modo del campo *MsgId* in MQMD per un inserimento in una singola coda.

Se questo campo non è presente in un record MQPMR o se il numero di record MQPMR è inferiore rispetto alle destinazioni, il valore in MQMD viene utilizzato per quelle destinazioni che non hanno un record MQPMR contenente un campo *MsgId* . Se tale valore è MQMI_NONE, viene generato un nuovo identificativo del messaggio per *ogni* di tali destinazioni (ossia, non due di tali destinazioni hanno lo stesso identificativo del messaggio).

Se viene specificato MQPMO_NEW_MSG_ID, vengono generati nuovi identificatori di messaggio per tutte le destinazioni nell'elenco di distribuzione, indipendentemente dal fatto che abbiano o meno record MQPMR. È diverso dal modo in cui viene elaborato MQPMO_NEW_CORREL_ID (consultare il campo *CorrelId*).

Questo è un campo di immissione / emissione.

CorrelId (MQBYTE24) per MQPMR

Questo è l'identificativo di correlazione da utilizzare per il messaggio inviato alla coda con un nome specificato dall'elemento corrispondente nell'array di strutture MQOR fornito nella chiamata MQOPEN o MQPUT1 . Viene elaborato nello stesso modo del campo *CorrelId* in MQMD per un inserimento in una singola coda.

Se questo campo non è presente in un record MQPMR o se il numero di record MQPMR è inferiore rispetto alle destinazioni, il valore in MQMD viene utilizzato per quelle destinazioni che non hanno un record MQPMR contenente un campo *CorrelId* .

Se viene specificato MQPMO_NEW_CORREL_ID, viene generato e utilizzato un nuovo identificativo di correlazione *singolo* per tutte le destinazioni nell'elenco di distribuzione, indipendentemente dal fatto che dispongano o meno di record MQPMR. Ciò è diverso dal modo in cui MQPMO_NEW_MSG_ID viene elaborato (consultare il campo *MsgId*).

Questo è un campo di immissione / emissione.

GroupId (MQBYTE24) per MQPM

GroupId è l'identificativo del gruppo da utilizzare per il messaggio inviato alla coda con il nome specificato dall'elemento corrispondente nell'array di strutture MQOR fornito nella chiamata MQOPEN o MQPUT1 . Viene elaborato nello stesso modo del campo *GroupId* in MQMD per un inserimento in una singola coda.

Se questo campo non è presente in un record MQPMR o se il numero di record MQPMR è inferiore rispetto alle destinazioni, il valore in MQMD viene utilizzato per quelle destinazioni che non hanno un record MQPMR contenente un campo *GroupId* . Il valore viene elaborato come documentato in Ordine fisico su una coda, ma con le seguenti differenze:

- GroupId viene creato dal QMName e da una data/ora. Pertanto, per mantenere univoci anche i nomi dei gestori code GroupId . Inoltre, non reimpostare gli orologi sulla macchina dei gestori code.
- Nei casi in cui viene utilizzato un nuovo identificativo di gruppo, il gestore code genera un identificativo di gruppo differente per ciascuna destinazione (ossia, non vi sono due destinazioni con lo stesso identificativo di gruppo).
- Nei casi in cui viene utilizzato il valore nel campo, la chiamata ha esito negativo con codice motivo MQRC_GROUP_ID_ERROR

Questo è un campo di immissione / emissione.

Feedback (MQLONG) per MQPMR

Questo è il codice di feedback da utilizzare per il messaggio inviato alla coda con il nome specificato dall'elemento corrispondente nell'array delle strutture MQOR fornito sulla chiamata MQOPEN o MQPUT1 . Viene elaborato nello stesso modo del campo *Feedback* in MQMD per un inserimento in una singola coda.

Se questo campo non è presente, viene utilizzato il valore in MQMD.

Questo è un campo di immissione.

AccountingToken (MQBYTE32) per MQPM

Questo è il token di account da utilizzare per il messaggio inviato alla coda con il nome specificato dall'elemento corrispondente nell'array di strutture MQOR fornito nella chiamata MQOPEN o MQPUT1 . Viene elaborato nello stesso modo del campo *AccountingToken* in MQMD per un inserimento in una singola coda. Consultare la descrizione di *AccountingToken* in “MQMD - Descrittore messaggi” a pagina 431 per informazioni sul contenuto di questo campo.

Se questo campo non è presente, viene utilizzato il valore in MQMD.

Questo è un campo di immissione.

MQRFH - Regole e intestazione di formattazione

La struttura MQRFH definisce il layout delle regole e l'intestazione di formattazione. Utilizzare questa intestazione per inviare i dati stringa sotto forma di coppie nome - valore.

Disponibilità

Tutti i sistemi IBM MQ , più IBM MQ MQI clients connessi a tali sistemi.

Nome formato

MQFMT_RF_HEADER

Serie di caratteri e codifica

I campi nella struttura MQRFH (incluso *NameValueString*) si trovano nella serie di caratteri e nella codifica fornita dai campi *CodedCharSetId* e *Encoding* nella struttura di intestazione che precede MQRFH o da quei campi nella struttura MQMD se MQRFH si trova all'inizio dei dati del messaggio dell'applicazione.

La serie di caratteri deve essere una serie di caratteri a byte singolo per i caratteri validi nei nomi coda.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 513. Campi in MQRFH per MQRFH

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQRFH	'RFH¬'
Versione (numero versione struttura)	MQRFH_VERSION_1	1
StrucLength (lunghezza in byte della struttura MQRFH)	MQRFH_STRUC_LENGTH_FIXED	32
Codifica (codifica numerica dei dati che seguono *NameValueString*)	MQEN_NATIVE	Dipende dall'ambiente
CodedCharSetId (specifica l'identificativo della serie di caratteri dei dati che seguono *NameValueString*)	MQCCSI_UNDEFINED	0
Formato (nome formato dei dati che seguono *NameValueString*)	MQFMT_NONE	Spazi
Indicatori (indicatori)	MQRFH_NONE	0
NameValueString (stringa di caratteri a lunghezza variabile contenente coppie nome - valore)	Nessuna	Nessuna

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macroMQRFH_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura:

```
MQRFH MyRFH = {MQRFH_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQRFH

```
typedef struct tagMQRFH MQRFH;
struct tagMQRFH {
  MQCHAR4  StrucId;         /* Structure identifier */
  MQLONG   Version;         /* Structure version number */
  MQLONG   StrucLength;     /* Total length of MQRFH including
                               NameValueString */
  MQLONG   Encoding;        /* Numeric encoding of data that follows
                               NameValueString */
```

```
  MQLONG   CodedCharSetId;  /* Character set identifier of data that
                               follows NameValueString */
  MQCHAR8  Format;          /* Format name of data that follows
                               NameValueString */
  MQLONG   Flags;           /* Flags */
};
```

Dichiarazione COBOL per MQRFH

```
**   MQRFH structure
  10 MQRFH.
**    Structure identifier
   15 MQRFH-STRUCID        PIC X(4).
**    Structure version number
   15 MQRFH-VERSION        PIC S9(9) BINARY.
**    Total length of MQRFH including NAMEVALUESTRING
   15 MQRFH-STRUCLENGTH    PIC S9(9) BINARY.
**    Numeric encoding of data that follows NAMEVALUESTRING
   15 MQRFH-ENCODING       PIC S9(9) BINARY.
**    Character set identifier of data that follows NAMEVALUESTRING
   15 MQRFH-CODEDCHARSETID PIC S9(9) BINARY.
**    Format name of data that follows NAMEVALUESTRING
   15 MQRFH-FORMAT         PIC X(8).
**    Flags
   15 MQRFH-FLAGS          PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQRFH

```
dcl
 1 MQRFH based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 StrucLength    fixed bin(31), /* Total length of MQRFH including
                                     NameValueString */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
                                     follows NameValueString */
  3 CodedCharSetId fixed bin(31), /* Character set identifier of data
                                     that follows NameValueString */
  3 Format         char(8),       /* Format name of data that follows
                                     NameValueString */
  3 Flags          fixed bin(31); /* Flags */
```

Dichiarazione High Level Assembler per MQRFH

```
MQRFH                DSECT
MQRFH_STRUCID        DS   CL4  Structure identifier
MQRFH_VERSION        DS   F    Structure version number
MQRFH_STRUCLENGTH    DS   F    Total length of MQRFH including
*                              NAMEVALUESTRING
MQRFH_ENCODING       DS   F    Numeric encoding of data that follows
*                              NAMEVALUESTRING
MQRFH_CODEDCHARSETID DS   F    Character set identifier of data that
*                              follows NAMEVALUESTRING
MQRFH_FORMAT         DS   CL8  Format name of data that follows
*                              NAMEVALUESTRING
MQRFH_FLAGS          DS   F    Flags
*
MQRFH_LENGTH         EQU  *-MQRFH
                     ORG  MQRFH
MQRFH_AREA           DS   CL(MQRFH_LENGTH)
```

Dichiarazione Visual Basic per MQRFH

```
Type MQRFH
  StrucId        As String*4 'Structure identifier'
  Version        As Long     'Structure version number'
  StrucLength    As Long     'Total length of MQRFH including'
                             'NameValueString'
  Encoding       As Long     'Numeric encoding of data that follows'
                             'NameValueString'
  CodedCharSetId As Long     'Character set identifier of data that'
                             'follows NameValueString'
  Format         As String*8 'Format name of data that follows'
```

```
                              'NameValueString'
  Flags          As Long      'Flags'
End Type
```

StrucId (MQCHAR4) per MQRFH

È l'identificativo della struttura delle regole e della struttura dell'intestazione di formattazione. È sempre un campo di immissione. Il valore è MQRFH_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQRFH
Identificativo per la struttura di intestazione di formattazione e regole.

Per il linguaggio di programmazione C, viene definita anche la costante MQRFH_STRUC_ID_ARRAY. Ha lo stesso valore di MQRFH_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQRFH

Questo è il numero di versione della struttura; il valore deve essere:

MQRFH_VERSION_1
Version-1 regole e formattazione della struttura dell'intestazione.

Il valore iniziale di questo campo è MQRFH_VERSION_1.

StrucLength (MQLONG) per MQRFH

Questa è la lunghezza, in byte, della struttura MQRFH, incluso il campo *NameValueString* alla fine della struttura. La lunghezza non include i dati utente che seguono il campo *NameValueString* .

Per evitare problemi durante la conversione dei dati utente in alcuni ambienti, *StrucLength* deve essere un multiplo di quattro.

La seguente costante fornisce la lunghezza della parte *fissa* della struttura, ovvero la lunghezza escluso il campo *NameValueString* :

MQRFH_STRUC_LENGTH_FIXED
Lunghezza della parte fissa della struttura MQRFH.

Il valore iniziale di questo campo è MQRFH_STRUC_LENGTH_FIXED.

Codifica (MQLONG) per MQRFH

Specifica la codifica numerica dei dati che seguono *NameValueString* ; non si applica ai dati numerici nella stessa struttura MQRFH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati.

Il valore iniziale di questo campo è MQENC_NATIVE.

CodedCharSetId (MQLONG) per MQRFH

Specifica l'identificativo della serie di caratteri dei dati che seguono *NameValueString* ; non si applica ai dati carattere nella stessa struttura MQRFH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. È possibile utilizzare il seguente valore speciale:

MQCCSI_INHERIT
I dati carattere nei dati *che seguono* questa struttura si trovano nella stessa serie di caratteri di questa struttura.

Il gestore code modifica questo valore nella struttura inviata nel messaggio nell'effettivo identificativo della serie di caratteri della struttura. Se non si verifica alcun errore, il valore MQCCSI_INHERIT non viene restituito dalla chiamata MQGET.

MQCCSI_INHERIT non può essere utilizzato se il valore del campo *PutApplType* in MQMD è MQAT_BROKER.

Il valore iniziale di questo campo è MQCCSI_UNDEFINED.

Formato (MQCHAR8) per MQRFH

Specifica il nome del formato dei dati che seguono *NameValueString*.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono le stesse del campo *Format* in MQMD.

Il valore iniziale di questo campo è MQFMT_NONE.

Indicatori (MQLONG) per MQRFH

È possibile specificare quanto segue:

MQRFH_NONE
Nessun indicatore.

Il valore iniziale di questo campo è MQRFH_NONE.

Stringa NameValue(MQCHARn) per MQRFH

Questa è una stringa di caratteri a lunghezza variabile contenente coppie nome - valore nel formato:

```
name1 value1 name2 value2 name3 value3 ...
```

Ogni nome o valore deve essere separato dal nome o valore adiacente da uno o più caratteri vuoti; questi spazi vuoti non sono significativi. Un nome o un valore può contenere spazi significativi prefissando e aggiungendo un suffisso al nome o al valore con doppi apici; tutti i caratteri tra i doppi apici aperti e i doppi apici di chiusura corrispondenti vengono considerati significativi. Nel seguente esempio, il nome è FAMOUS_WORDSe il valore è `Hello World`:

```
FAMOUS_WORDS "Hello World"
```

Un nome o un valore può contenere qualsiasi carattere diverso dal carattere null (che funge da delimitatore per *NameValueString*). Tuttavia, per facilitare l'interoperabilità, un'applicazione può limitare i nomi ai caratteri seguenti:

- Primo carattere: alfabetico maiuscolo o minuscolo (da A a Z o da a a z) o carattere di sottolineatura.
- Caratteri successivi: alfabetico maiuscolo o minuscolo, cifra decimale (da 0 a 9), carattere di sottolineatura, trattino o punto.

Se un nome o un valore contiene uno o più doppi apici, il nome o il valore deve essere racchiuso tra doppi apici e ogni doppio apice all'interno della stringa deve essere raddoppiato:

```
Famous_Words "The program displayed ""Hello World"""
```

I nomi e i valori sono sensibili al maiuscolo / minuscolo, ovvero, le lettere minuscole non sono considerate uguali alle lettere maiuscole. Ad esempio, FAMOUS_WORDS e `Famous_Words` sono due nomi differenti.

La lunghezza in byte di *NameValueString* è uguale a *StrucLength* meno MQRFH_STRUC_LENGTH_FIXED. Per evitare problemi di conversione dei dati utente in alcuni ambienti, rendere questa lunghezza un multiplo di quattro. Riempire *NameValueString* con spazi vuoti fino a questa lunghezza oppure terminarlo prima posizionando un carattere null dopo l'ultimo carattere significativo nella stringa. Il carattere null e i byte successivi, fino alla lunghezza specificata di *NameValueString*, vengono ignorati.

Nota: Poiché la lunghezza di questo campo non è fissa, il campo viene omesso dalle dichiarazioni della struttura fornite per i linguaggi di programmazione supportati.

MQRFH2 - Regole e intestazione di formattazione 2

L'intestazione MQRFH2 si basa sull'intestazione MQRFH , ma consente alle stringhe Unicode di essere trasportate senza conversione e può contenere tipi di dati numerici.La struttura MQRFH2 definisce il formato delle regole version-2 e l'intestazione di formattazione. Utilizzare questa intestazione per inviare i dati che sono stati codificati utilizzando una sintassi simile a XML. Un messaggio può contenere due o più strutture MQRFH2 in serie, con i dati utente facoltativamente che seguono l'ultima struttura MQRFH2 nella serie.

Disponibilità

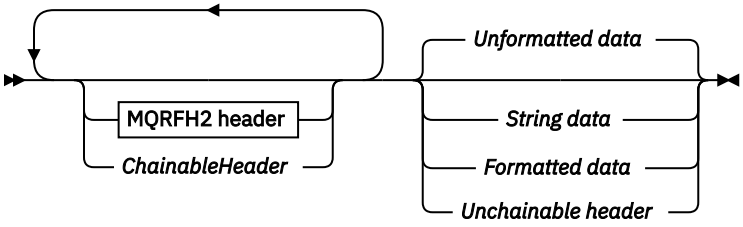
Tutti i sistemi IBM MQ , più IBM MQ MQI clients connessi a tali sistemi.

Nome formato

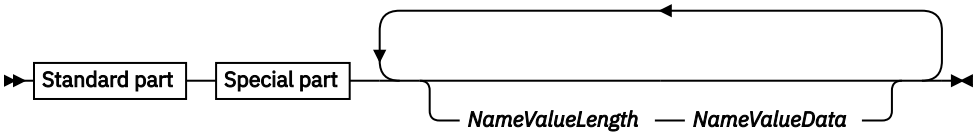
MQFMT_RF_HEADER_2

Syntax

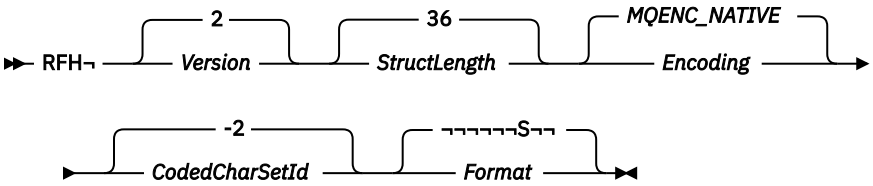
IBM MQ Message



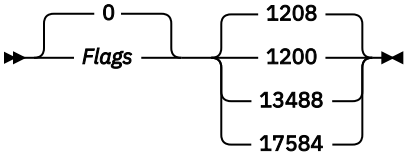
MQRFH2 header



Standard part



Special part



Serie di caratteri e codifica

Regole speciali si applicano alla serie di caratteri e alla codifica utilizzati per la struttura MQRFH2 :

- I campi diversi da *NameValueData* si trovano nella serie di caratteri e nella codifica forniti dai campi *CodedCharSetId* e *Encoding* nella struttura dell'intestazione che precede MQRFH2o da tali campi nella struttura MQMD se MQRFH2 si trova all'inizio dei dati del messaggio dell'applicazione.

  La serie di caratteri deve essere una serie di caratteri a byte singolo per i caratteri validi nei nomi coda.

Quando MQGMO_CONVERT viene specificato nella chiamata MQGET , il gestore code converte i campi MQRFH2 , diversi da *NameValueData*, nella serie di caratteri e nella codifica richiesti.

- *NameValueData* è nella serie di caratteri fornita dal campo *NameValueCCSID* . Solo le serie di caratteri Unicode elencate sono valide per *NameValueCCSID* ; Consultare la descrizione di *NameValueCCSID* per i dettagli.

  Alcune serie di caratteri hanno una rappresentazione che dipende dalla codifica. Se *NameValueCCSID* è una di queste serie di caratteri, *NameValueData* deve essere nella stessa codifica degli altri campi in MQRFH2.

  Quando viene specificato MQGMO_CONVERT nella chiamata MQGET , il gestore code converte *NameValueData* nella codifica richiesta, ma non ne modifica la serie di caratteri.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 514. Campi in MQRFH2 per MQRFH2

Nome campo	Nome della costante	Valore della costante
StrucId (identificativo della struttura)	ID_STRUC_MQRFH	'RFH¬'
Versione (numero versione struttura)	MQRFH_VERSION_2	2
StrucLength (lunghezza in byte della struttura MQRFH2)	MQRFH_STRUC_LENGTH_FIXED_2	36
Codifica (codifica numerica dei dati che seguono l'ultimo campo *NameValueData*)	MQEN_NATIVE	Dipende dall'ambiente
CodedCharSetId (identificativo della serie di caratteri dei dati che seguono l'ultimo campo *NameValueData*)	MQCCSI_INHERIT	-2
Formato (nome formato dei dati che seguono l'ultimo campo *NameValueData*)	MQFMT_NONE	Spazi
Indicatori (indicatori)	MQRFH_NONE	0
NameValueCCSID (coded character set identifier dei dati nel campo *NameValueData*)	Nessuna	1208
NameValueLunghezza (lunghezza in byte dei dati nel campo *NameValueData*)	Nessuna	None

Tabella 514. Campi in MQRFH2 per MQRFH2 (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
NameValueData (coppie nome - valore delle proprietà del messaggio)	Nessuna	Nessuna

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macroMQRFH2_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQRFH2 MyRFH2 = {MQRFH2_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQRFH2

```
typedef struct tagMQRFH2 MQRFH2;
struct tagMQRFH2 {
  MQCHAR4  StrucId;         /* Structure identifier */
  MQLONG   Version;         /* Structure version number */
  MQLONG   StrucLength;     /* Total length of MQRFH2 including all
                               NameValueLength and NameValueData
                               fields */
  MQLONG   Encoding;        /* Numeric encoding of data that follows
                               last NameValueData field */
  MQLONG   CodedCharSetId;  /* Character set identifier of data that
                               follows last NameValueData field */
  MQCHAR8  Format;          /* Format name of data that follows last
                               NameValueData field */
  MQLONG   Flags;           /* Flags */
  MQLONG   NameValueCCSID;  /* Character set identifier of
                               NameValueData */
};
```

Dichiarazione COBOL per MQRFH2

```
**   MQRFH2 structure
  10 MQRFH2.
**    Structure identifier
   15 MQRFH2-STRUCID        PIC X(4).
**    Structure version number
   15 MQRFH2-VERSION        PIC S9(9) BINARY.
**    Total length of MQRFH2 including all NAMEVALUELENGTH and
**    NAMEVALUEDATA fields
   15 MQRFH2-STRUCLENGTH    PIC S9(9) BINARY.
**    Numeric encoding of data that follows last NAMEVALUEDATA field
   15 MQRFH2-ENCODING       PIC S9(9) BINARY.
**    Character set identifier of data that follows last NAMEVALUEDATA
**    field
   15 MQRFH2-CODEDCHARSETID PIC S9(9) BINARY.
**    Format name of data that follows last NAMEVALUEDATA field
   15 MQRFH2-FORMAT         PIC X(8).
**    Flags
   15 MQRFH2-FLAGS          PIC S9(9) BINARY.
**    Character set identifier of NAMEVALUEDATA
   15 MQRFH2-NAMEVALUECCSID PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQRFH2

```
dcl
 1 MQRFH2 based,
  3 StrucId        char(4),       /* Structure identifier */
```

```
  3 Version        fixed bin(31), /* Structure version number */
  3 StrucLength    fixed bin(31), /* Total length of MQRFH2 including
                                     all NameValueLength and
                                     NameValueData fields */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
                                     follows last NameValueData field */
  3 CodedCharSetId fixed bin(31), /* Character set identifier of data
                                     that follows last NameValueData
                                     field */
  3 Format         char(8),       /* Format name of data that follows
                                     last NameValueData field */
  3 Flags          fixed bin(31), /* Flags */
  3 NameValueCCSID fixed bin(31); /* Character set identifier of
                                     NameValueData */
```

La dichiarazione High Level Assembler per MQRFH2

```
MQRFH                 DSECT
MQRFH_STRUCID         DS   CL4  Structure identifier
MQRFH_VERSION         DS   F    Structure version number
MQRFH_STRUCLENGTH     DS   F    Total length of MQRFH2 including all
*                               NAMEVALUELENGTH and NAMEVALUEDATA fields
MQRFH_ENCODING        DS   F    Numeric encoding of data that follows
*                               last NAMEVALUEDATA field
MQRFH_CODEDCHARSETID  DS   F    Character set identifier of data that
*                               follows last NAMEVALUEDATA field
MQRFH_FORMAT          DS   CL8  Format name of data that follows last
*                               NAMEVALUEDATA field
MQRFH_FLAGS           DS   F    Flags
MQRFH_NAMEVALUECCSID  DS   F    Character set identifier of
*                               NAMEVALUEDATA
*
MQRFH_LENGTH          EQU  *-MQRFH
                      ORG  MQRFH
MQRFH_AREA            DS   CL(MQRFH_LENGTH)
```

Dichiarazione Visual Basic per MQRFH2

```
Type MQRFH2
  StrucId        As String*4 'Structure identifier'
  Version        As Long     'Structure version number'
  StrucLength    As Long     'Total length of MQRFH2 including all'
                             'NameValueLength and NameValueData fields'
  Encoding       As Long     'Numeric encoding of data that follows'
                             'last NameValueData field'
  CodedCharSetId As Long     'Character set identifier of data that'
                             'follows last NameValueData field'
  Format         As String*8 'Format name of data that follows last'
                             'NameValueData field'
  Flags          As Long     'Flags'
  NameValueCCSID As Long     'Character set identifier of NameValueData'
End Type
```

StrucId (MQCHAR4) per MQRFH2

Questo è l'identificativo della struttura delle regole e la struttura dell'intestazione di formattazione due. È sempre un campo di immissione. Il valore è MQRFH2_STRUC_ID.

Il valore deve essere:

MQRFH2_STRUC_ID
Identificativo per la struttura delle regole e dell'intestazione di formattazione due.

Per il linguaggio di programmazione C, viene definita anche la costante MQRFH2_STRUC_ID_ARRAY . Ha lo stesso valore di MQRFH2_STRUC_ID, ma è un array di caratteri anziché una stringa.

Versione (MQLONG) per MQRFH2

Questo è il numero di versione della struttura; il valore deve essere:

MQRFH_VERSION_2
Regole Version-2 e struttura dell'intestazione di formattazione.

Il valore iniziale di questo campo è MQRFH_VERSION_2.

StrucLength (MQLONG) per MQRFH2

È la lunghezza in byte della struttura MQRFH2 , inclusi i campi *NameValueLength* e *NameValueData* alla fine della struttura. È valido che vi siano più coppie di campi *NameValueLength* e *NameValueData* alla fine della struttura, nella sequenza:

```
length1, data1, length2, data2, ...
```

StrucLength non include alcun dato utente che potrebbe seguire l'ultimo campo *NameValueData* alla fine della struttura.

Per evitare problemi con la conversione dei dati utente in alcuni ambienti, *StrucLength* deve essere un multiplo di quattro.

La seguente costante fornisce la lunghezza della parte *fissa* della struttura, ovvero la lunghezza escludendo i campi *NameValueLength* e *NameValueData* :

MQRFH_STRUC_LENGTH_FIXED_2
Lunghezza della parte fissa della struttura MQRFH2 .

Il valore iniziale di questo campo è MQRFH_STRUC_LENGTH_FIXED_2.

Codifica (MQLONG) per MQRFH2

Specifica la codifica numerica dei dati che seguono l'ultimo campo *NameValueData* ; non si applica ai dati numerici nella struttura MQRFH2 .

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati.

Il valore iniziale di questo campo è MQENC_NATIVE.

CodedCharSetId (MQLONG) per MQRFH2

Specifica l'identificativo della serie di caratteri dei dati che seguono l'ultimo campo *NameValueData* ; non si applica ai dati carattere nella stessa struttura MQRFH2 .

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. È possibile utilizzare il seguente valore speciale:

MQCCSI_INHERIT
I dati carattere nei dati *che seguono* questa struttura si trovano nella stessa serie di caratteri di questa struttura.

Il gestore code modifica questo valore nella struttura inviata nel messaggio nell'effettivo identificativo della serie di caratteri della struttura. Se non si verifica alcun errore, il valore MQCCSI_INHERIT non viene restituito dalla chiamata MQGET.

MQCCSI_INHERIT non può essere utilizzato se il valore del campo *PutApplType* in MQMD è MQAT_BROKER.

Il valore iniziale di questo campo è MQCCSI_INHERIT.

Formato (MQCHAR8) per MQRFH2

Specifica il nome formato dei dati che seguono l'ultimo campo *NameValueData* .

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono le stesse del campo *Format* in MQMD.

Il valore iniziale di questo campo è MQFMT_NONE.

Indicatori (MQLONG) per MQRFH2

Il valore iniziale di questo campo è MQRFH_NONE. MQRFH_NONE deve essere specificato.

MQRFH_NONE
Nessun indicatore.

MQRFH_INTERNAL
L'intestazione MQRFH2 contiene proprietà impostate internamente.

MQRFH_INTERNAL è per l'utilizzo del gestore code.

I primi 16 bit, MQRFH_FLAGS_RESTRICTED_MASK, sono riservati per gli indicatori impostati dal gestore code. Gli indicatori che un utente potrebbe impostare sono definiti negli ultimi 16 bit.

NameValueCCSID (MQLONG) per MQRFH2

Specifica il CCSID (coded character set identifier) dei dati nel campo *NameValueData* . Questo è diverso dalla serie di caratteri delle altre stringhe nella struttura MQRFH2 e può essere diverso dalla serie di caratteri dei dati (se presenti) che seguono l'ultimo campo *NameValueData* alla fine della struttura.

NameValueCCSID deve avere uno dei seguenti valori:

CCSID	**Significato**
1200	UTF-16, versione Unicode più recente supportata
13488	UTF-16, sottoinsieme Unicode versione 2.0
17584	UTF-16, sottoinsieme Unicode versione 3.0 (include il simbolo Euro)
1208	UTF-8, versione Unicode più recente supportata

Per le serie di caratteri UTF-16 , la codifica (ordine byte) di *NameValueData* deve corrispondere alla codifica degli altri campi nella struttura MQRFH2 .

I caratteri oltre il piano Unicode Basic Multilingual Plane (quelli sopra U + FFFF), rappresentati in UTF-16 da punti di codice surrogati (da X'D800'a X'DFFF') o quattro byte in UTF-8, non sono supportati.

Nota: Se *NameValueCCSID* non dispone di uno dei valori elencati in precedenza e la struttura MQRFH2 richiede la conversione nella chiamata MQGET, la chiamata viene completata con il codice di errore MQRC_SOURCE_CCSID_ERROR e il messaggio non viene convertito.

Il valore iniziale di questo campo è 1208.

NameValue(MQLONG) per MQRFH2

La lunghezza del campo NameValueData corrispondente

Specifica la lunghezza in byte dei dati nel campo *NameValueData* . *NameValueLength* deve essere un multiplo di quattro.

Nota: I campi *NameValueLength* e *NameValueData* sono facoltativi, ma se presenti devono essere una coppia e adiacenti. La coppia di campi può essere ripetuta tutte le volte necessarie, ad esempio:

```
length1 data1 length2 data2 length3 data3
```

Poiché questi campi sono facoltativi, vengono omessi dalle dichiarazioni della struttura fornite per i vari linguaggi di programmazione supportati.

NameValueDati (MQCHARn) per MQRFH2

NameValueData è un campo a lunghezza variabile che contiene una cartella contenente coppie nome - valore delle proprietà del messaggio. Una cartella è una stringa di caratteri a lunghezza variabile che contiene dati codificati utilizzando una sintassi simile a XML. La lunghezza in byte della stringa di caratteri viene fornita dal campo *NameValueLength* che precede il campo *NameValueData* . La lunghezza deve essere un multiplo di quattro.

I campi *NameValueLength* e *NameValueData* sono facoltativi, ma se presenti devono essere una coppia e adiacenti. La coppia di campi può essere ripetuta tutte le volte necessarie, ad esempio:

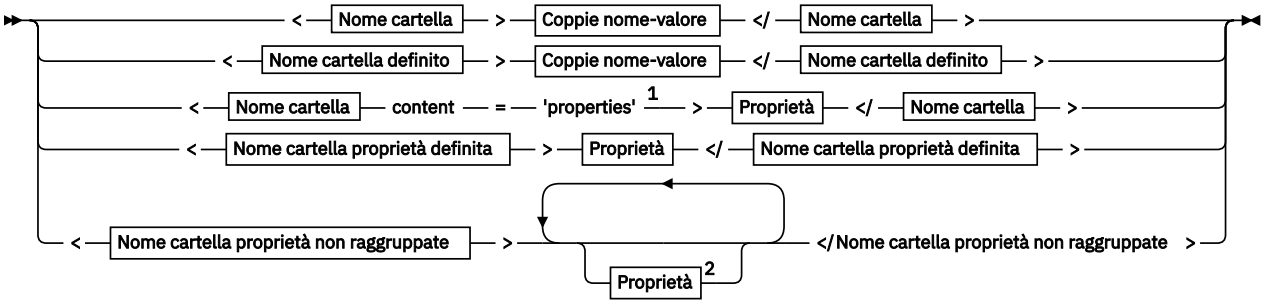
```
length1 data1 length2 data2 length3 data3
```

NameValueData non viene convertito nella serie di caratteri specificata nella chiamata MQGET . Anche se il messaggio viene richiamato con l'opzione MQGMO_CONVERT in vigore *NameValueData* rimane nella relativa serie di caratteri originale. Tuttavia, *NameValueData* viene convertito nella codifica specificata sulla chiamata MQGET .

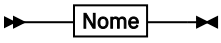
Note:

- Poiché questi campi sono facoltativi, vengono omessi dalle dichiarazioni della struttura fornite per i vari linguaggi di programmazione supportati.
- I termini "definiti" e "riservati" sono utilizzati nel diagramma di sintassi. "Definito" significa che IBM MQutilizza il nome. "Riservato" significa che il nome è riservato per un utilizzo futuro da parte di IBM MQ.

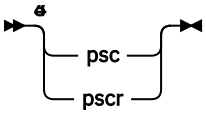
NameValueData sintassi



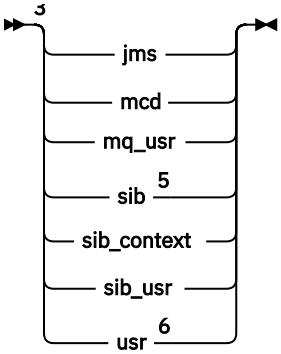
Nome cartella



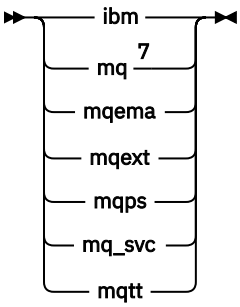
Nome cartella definito



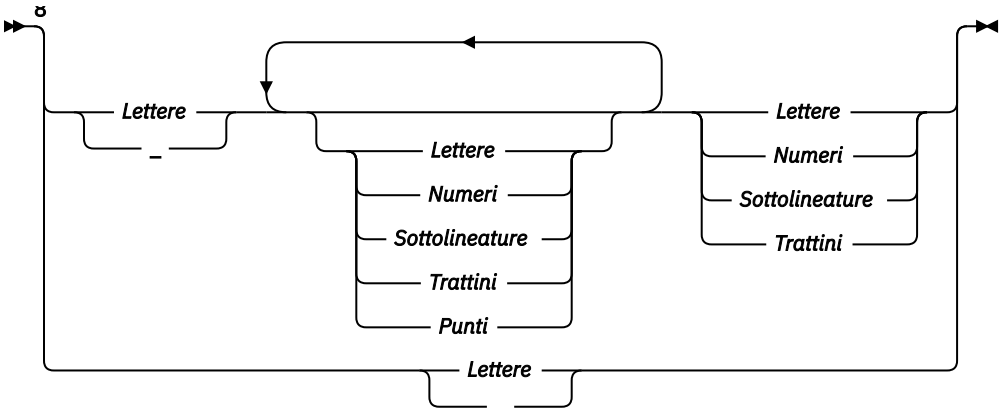
Nome cartella proprietà definita



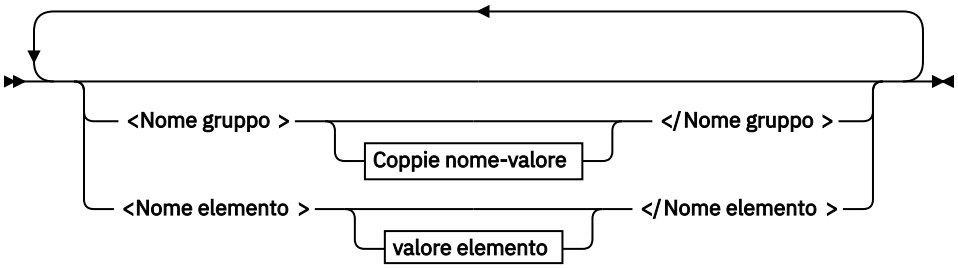
Nome cartella proprietà non raggruppate



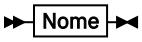
Nome



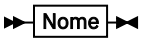
Coppie nome-valore



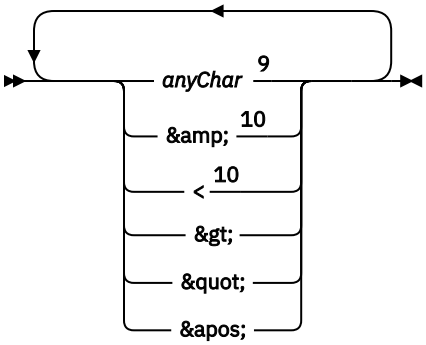
Nome gruppo



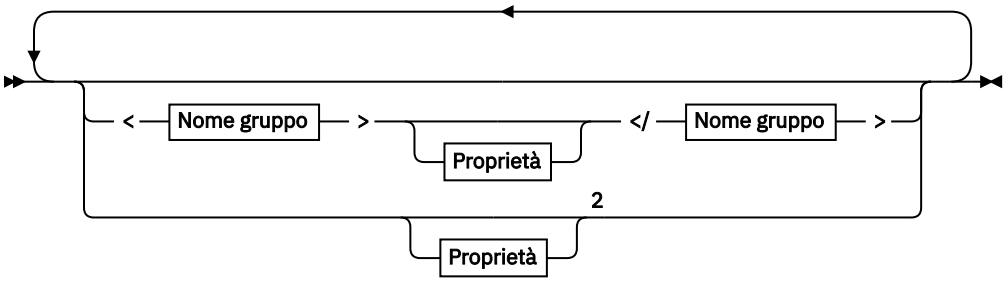
Nome elemento



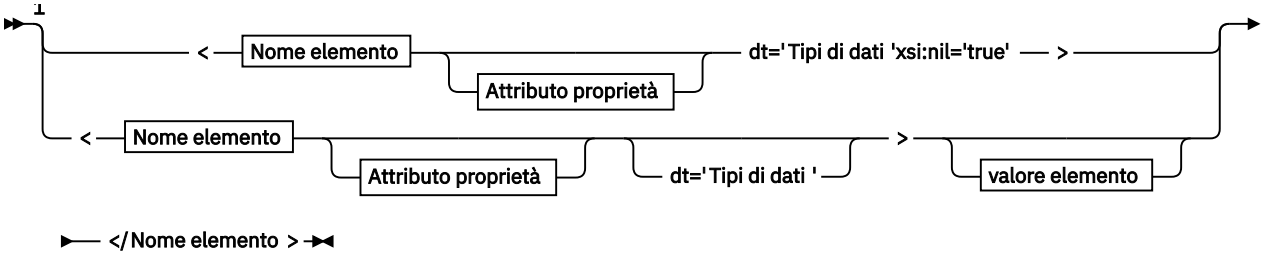
valore elemento



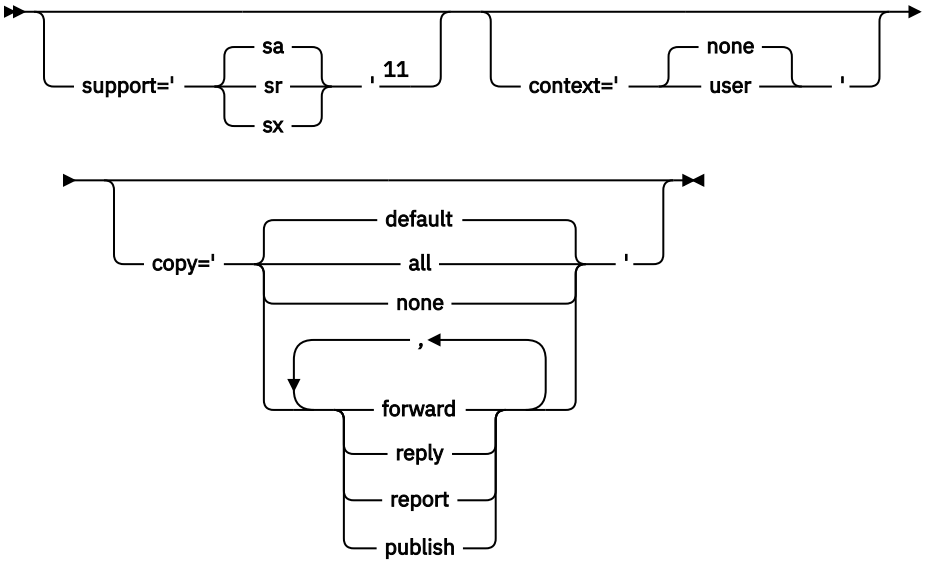
Proprietà



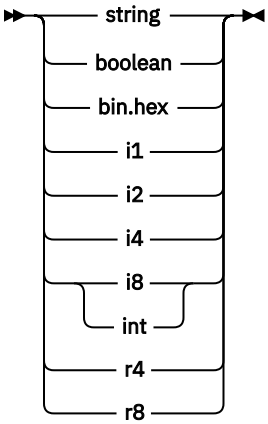
Proprietà



Attributo proprietà



Tipi di dati



Note:

[1] Le virgolette doppie o singole sono valide.

[2] Non utilizzare un nome proprietà non valido; consultare “Nome proprietà non valido” a pagina 562. Utilizzare un nome proprietà riservato solo per lo scopo definito; consultare “Nomi di proprietà definiti” a pagina 562.

[3] Il nome deve essere in minuscolo.

[4] È supportata solo una cartella `psc` e `pscr` .

[5] WebSphere Application Server Il servizio Integration Bus ignora le cartelle sib, sib_contexte sib_usr nelle successive intestazioni MQRFH2 e solo le proprietà nella prima intestazione MQRFH2 sono significative.

[6] Non più di una cartella `usr` deve essere presente in un MQRFH2. Le proprietà nella cartella `usr` non devono essere presenti più di una volta.

[7] Solo le proprietà nella prima cartella `mq` sono significative. Se la cartella è `UTF-8`, sono supportati solo caratteri `UTF-8` a byte singolo. L'unico carattere spazio vuoto è Unicode `U+0020`.

[8] I caratteri validi sono definiti nella specifica XML W3C e consistono essenzialmente in categorie Unicode `Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm,` e `Nd` ; vedere Categorie di caratteri Unicode

[9] Tutti i caratteri sono significativi. Gli spazi iniziali e finali fanno parte del valore dell'elemento.

[10] Non utilizzare caratteri non validi; consultare “Caratteri non validi” a pagina 561. Utilizzare una sequenza di escape invece di questi caratteri non validi.

[11] L'attributo della proprietà di supporto è valido solo sulla cartella `mq`

Nome cartella

NameValueData contiene una singola cartella. Per creare più cartelle, creare più campi *NameValueData* . È possibile creare più campi *NameValueData* in una singola intestazione MQRFH2 all'interno di un messaggio. In alternativa, è possibile creare più intestazioni concatenate MQRFH2 , ciascuna contenente più campi *NameValueData* .

L'ordine delle intestazioni MQRFH2 e l'ordine dei campi *NameValueData* non fa alcuna differenza per il contenuto logico di una cartella. Se la stessa cartella è presente più di una volta in un messaggio, la cartella viene analizzata nel suo insieme. Se la stessa proprietà si verifica in più istanze della stessa cartella, viene analizzata come un elenco.

Un'analisi corretta di un MQRFH2 non è influenzata dai modi alternativi in cui una cartella può essere fisicamente memorizzata in un messaggio.

Quattro cartelle non seguono questa regola. Vengono analizzate solo la prima istanza della cartella `mq`, `sib`, `sib_contexte` `sib_usr` .

Se la stessa proprietà si verifica più di una volta nel contenuto combinato delle intestazioni MQRFH2 concatenate, viene analizzata solo la prima istanza della proprietà. Se una proprietà viene impostata utilizzando una chiamata API, come MQSETMP, e aggiunta a un MQRFH2 direttamente da un'applicazione, la chiamata API ha la precedenza.

Un nome cartella è il nome di una cartella contenente coppie nome - valore o gruppi. I gruppi e le coppie nome - valore possono essere combinati allo stesso livello nella struttura di cartelle; consultare Figura 1 a pagina 551. Non combinare un nome gruppo e un nome elemento; consultare Figura 2 a pagina 551

```
<group1><nvp1>value</nvp1></group1><group2><nvp2>value</nvp2></group2>
<group3><nvp1>value</nvp1></group3><nvp3>value</nvp3>
```

Figura 1. Utilizzo corretto di gruppi e coppie nome - valore

```
<group1><nvp1> value </nvp1> value </group1>
```

Figura 2. Utilizzo non corretto di coppie nome - valore e gruppi

Non utilizzare un nome cartella non valido o riservato; consultare “Nome percorso non valido” a pagina 561 e “Nome cartella o proprietà riservata” a pagina 561. Utilizzare un nome cartella definito solo per lo scopo definito; consultare “Nome cartella definito” a pagina 552.

Se si aggiunge l'attributo `'content=properties'` alla tag del nome cartella, la cartella diventa una cartella delle proprietà; consultare Figura 3 a pagina 552.

```
<myFolder></myfolder>
<myPropertyFolder contents='properties'></myPropertyFolder>
```

Figura 3. Esempio di una cartella e di una cartella di proprietà

I nomi delle cartelle sono sensibili al maiuscolo / minuscolo. I nomi delle cartelle e i nomi delle cartelle delle proprietà condividono lo stesso spazio dei nomi. Devono avere nomi diversi. `Folder1` in Figura 4 a pagina 552 deve essere un nome diverso rispetto a `Folder2` in Figura 5 a pagina 552.

```
< Folder1 ><NVP1> value </NVP1></ Folder1 >
```

Figura 4. `Folder1` spazio dei nomi

```
< Folder2 content='properties'>< Property1 > value </ Property1 ></ Folder2 >
```

Figura 5. `Folder2` spazio dei nomi

Gruppi, proprietà e coppie nome - valore in cartelle differenti hanno spazi dei nomi differenti. `Property1` in Figura 5 a pagina 552 è una proprietà diversa da `Property1` in Figura 6 a pagina 552.

```
<Folder3 content='properties'>< Property1 > value </ Property1 ></Folder3>
```

Figura 6. `Folder3` spazio dei nomi

Le cartelle di proprietà sono diverse dalle cartelle non di proprietà in due aspetti importanti:

1. Le cartelle delle proprietà contengono proprietà e le cartelle non di proprietà contengono coppie nome - valore. Le cartelle differiscono leggermente, sintatticamente.
2. Utilizzare le interfacce definite, ad esempio le proprietà MQI o le proprietà del messaggio JMS , per accedere alle proprietà del messaggio. Le interfacce garantiscono che le cartelle delle proprietà in MQRFH2 abbiano un formato corretto. Una cartella di proprietà con formato corretto è interoperabile tra gestori code su piattaforme differenti e release differenti.

La proprietà del messaggio MQI è un modo efficace per leggere e scrivere un MQRFH2ed evita le difficoltà di analizzare correttamente un MQRFH2 .

Nome cartella definito

Un nome cartella definito è il nome di una cartella riservata per l'utilizzo da parte di IBM MQo di un altro prodotto. non creare una cartella con lo stesso nome e non aggiungere le proprie coppie nome - valore alle cartelle. Le cartelle definite sono `psc` e `pscr`.

`psc` e `pscr` vengono utilizzati dalla pubblicazione / sottoscrizione accodata.

Un messaggio segmentato inserito con MQMF_SEGMENT o MQMF_SEGMENTATION_ALLOWED non può contenere un MQRFH2 con un nome cartella definito. MQPUT ha esito negativo con codice di errore 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Nome cartella proprietà definita

Un nome cartella di proprietà definito è il nome di una cartella di proprietà utilizzata da IBM MQo da un altro prodotto. Per i nomi delle cartelle e il loro contenuto, vedere Cartelle delle proprietà. I nomi delle cartelle delle proprietà definiti sono un sottoinsieme di tutti i nomi delle cartelle riservati da IBM MQ; vedere "Nome cartella o proprietà riservata" a pagina 561.

Qualsiasi elemento memorizzato in una cartella di proprietà definita è una proprietà. Un elemento memorizzato in una cartella di proprietà definita non deve avere un attributo `content='properties'` .

È possibile aggiungere proprietà solo alle cartelle di proprietà definite `usr`, `mq_usre` `sib_usr`. In altre cartelle delle proprietà, come `mq` e `sib`, IBM MQ ignora o scarta le proprietà che non riconosce.

La descrizione di ogni cartella di proprietà definita elenca le proprietà definite da IBM MQ che possono essere utilizzate dai programmi applicativi. Ad alcune delle proprietà si accede indirettamente impostando o richiamando una proprietà JMS , mentre ad altre si accede direttamente utilizzando le chiamate MQSETMP e MQINQMP MQI.

Le cartelle delle proprietà definite contengono anche altre proprietà che IBM MQ ha riservato, ma a cui le applicazioni non hanno accesso. I nomi delle proprietà riservate non sono elencati. Nessuna proprietà riservata è presente nelle cartelle delle proprietà `usr`, `mq_usre` `sib_usr` . Ma non creare proprietà con nomi proprietà non validi; consultare "Nome proprietà non valido" a pagina 562.

Cartelle delle proprietà

jms

`jms` contiene campi di intestazione JMS e proprietà JMSX che non possono essere pienamente espresse nell'MQMD. La cartella `jms` è sempre presente in un MQRFH2 JMS.

Tabella 515. Nome della proprietà jms, sinonimo, tipo di dati e cartella

Sinonimo proprietà	Nome proprietà	Tipo dati	Cartella
JMSDestination	jms.Dst	string	<jms><Dst> *destination* </Dst></jms>
JMSExpiration	jms.Exp	i8	<jms><Exp> *expiration* </Exp></jms>
Correlazione JMS	jms.Cid	string	<jms><Cid> *correlationId* </Cid></jms>
Consegna JMS	jms.Dlv	i4	<jms><Dlv> *delivery* </Dlv></jms>
JMSPriority	jms.Pri	i4	<jms><Pri> *priority* </Pri></jms>
JMSReplyTo	jms.Rto	string	<jms><Rto> *replyToURI* </Rto></jms>
JMSTimeStamp	jms.Tms	i8	<jms><Tms> *timestamp* </Tms></jms>
JMSXGroupID	jms.Gid	string	<jms><Gid> *groupId* </Gid></jms>

Tabella 515. Nome della proprietà jms, sinonimo, tipo di dati e cartella (Continua)			
Sinonimo proprietà	**Nome proprietà**	**Tipo dati**	**Cartella**
JMSXGroupSeq	jms.Seq	i4	<jms><Seq> *messageSequenceNo* </Seq></jms>

Non aggiungere le proprie proprietà nella cartella jms.

mcd

mcd contiene proprietà che descrivono il formato del messaggio. Ad esempio, la proprietà Msd del dominio del servizio di messaggi identifica un messaggio JMS come JMSTextMessage, JMSBytesMessage, JMSStreamMessage, JMSMapMessage, JMSObjectMessage o null.

La cartella mcd è sempre presente in un messaggio di JMS contenente un MQRFH2.

È sempre presente in un messaggio contenente un MQRFH2 inviato da IBM Integration Bus. Descrive il dominio, il formato, il tipo e la serie di un messaggio.

Tabella 516. Nome, sinonimo, tipo di dati e cartella della proprietà mcd			
Sinonimo proprietà	**Nome proprietà**	**Tipo dati**	**Cartella**
	mcd.Msd	string	<mcd><Msd>*messageDomain*</Msd></mcd>
	mcd.Set	string	<mcd><Set>*messageDomain*</Set></mcd>
	mcd.Type	string	<mcd><Type>*messageDomain*</Type></mcd>
	mcd.Fmt	string	<mcd><Fmt>*messageDomain*</Fmt></mcd>

Non aggiungere le proprie proprietà nella cartella mcd.

mq_usr

mq_usr contiene proprietà definite dall'applicazione che non sono esposte come JMS proprietà definite dall'utente. Le proprietà che non soddisfano i requisiti JMS possono essere collocate in questa cartella.

È possibile creare proprietà nella cartella mq_usr . Le proprietà create in mq_usr sono le proprietà create in nuove cartelle con l'attributo content='properties' .

sib

sib contiene le proprietà del messaggio di sistema WebSphere Application Server service integration bus (WAS/SIB). Le proprietà sib non sono esposte come proprietà JMS per le applicazioni IBM MQ JMS perché non sono dei tipi supportati. Ad esempio, alcune proprietà sib non possono essere esposte come proprietà JMS perché sono array di byte. Alcune proprietà sib sono esposte alle applicazioni WAS/SIB come proprietà JMS_IBM_* ; queste includono le proprietà dei percorsi di instradamento inverso e di inoltro.

Non aggiungere le proprie proprietà nella cartella sib.

sib_context

sib_context contiene le proprietà dei messaggi di sistema WAS/SIB che non sono esposte alle applicazioni utente WAS/SIB o come proprietà JMS . sib_context contiene proprietà di sicurezza e transazionali utilizzate per i servizi Web.

Non aggiungere le proprie proprietà nella cartella sib_context.

sib_usr

sib_usr contiene le proprietà del messaggio utente WAS/SIB che non sono esposte come proprietà utente JMS poiché non sono di tipi supportati. sib_usr è esposto alle applicazioni WAS/SIB nell'interfaccia SIMessage ; consultare Developing Service Integration.

Il tipo di proprietà sib_usr deve essere bin.hexe il formato del valore deve essere corretto. Se un'applicazione IBM MQ scrive un elemento di tipo bin.hex nella cartella nel formato errato, l'applicazione riceve un IOException. Se il tipo di dati della proprietà non è bin.hex l'applicazione riceve un ClassCastException.

Non tentare di rendere le proprietà utente JMS disponibili per WAS/SIB utilizzando questa cartella; utilizzare invece la cartella usr .

È possibile creare proprietà nella cartella sib_usr .

usr

usr contiene le proprietà di JMS definite dall'applicazione associate al messaggio. La cartella usr è presente solo se un'applicazione ha impostato una proprietà definita dall'applicazione.

usr è la cartella delle proprietà predefinita. Se una proprietà è impostata senza un nome cartella, viene collocata nella cartella usr .

Tabella 517. nome proprietà usr, sinonimo, tipo di dati e cartella

Sinonimo proprietà	Nome proprietà	Tipo dati	Cartella
	usr.contentType	string	<usr><contentType>text/xml; charset=utf-8</contentType></usr>
	usr.endPointURL	string	<usr><endPointURL> *URI* </endPointURL></usr>
	usr.targetService	string	<usr><targetService> *serviceName* </targetService></usr>
	usr.soapAction	string	<usr><soapAction> *name* </soapAction></usr>
	usr.transportVersion	string	<usr><transportVersion> *version* </transportVersion></usr>

È possibile creare proprietà nella cartella usr .

Un messaggio segmentato inserito con MQMF_SEGMENT o MQMF_SEGMENTATION_ALLOWED non può contenere un MQRFH2 con un nome cartella di proprietà definito. MQPUT ha esito negativo con codice di errore 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Nome cartella proprietà non raggruppate

ibm

ibm contiene proprietà utilizzate solo da IBM MQ.

Tabella 518. Nome, sinonimo, tipo di dati e cartella della proprietà ibm

Sinonimo proprietà	Nome proprietà	Tipo dati	Cartella
	ibm.rfp	string	<ibm><rfp>fingerprint</rfp></ibm>

Non aggiungere le proprie proprietà nella cartella ibm.

mq

mq contiene proprietà utilizzate solo da IBM MQ.

Le seguenti limitazioni si applicano alle proprietà nella cartella mq :

- Solo le proprietà nella prima cartella mq significativa nel messaggio vengono utilizzate da MQ; le proprietà in qualsiasi altra cartella mq nel messaggio vengono ignorate.
- Nella cartella sono consentiti solo caratteri UTF-8 a byte singolo. Un carattere multibyte nella cartella può causare un errore di analisi e il messaggio può essere rifiutato.
- Non utilizzare le stringhe di escape nella cartella. Una stringa di escape viene considerata come il valore effettivo dell'elemento.
- Solo il carattere Unicode U+0020 viene considerato come spazio vuoto all'interno della cartella. Tutti gli altri caratteri vengono trattati come significativi e possono causare l'esito negativo dell'analisi della cartella e il rifiuto del messaggio.

Se l'analisi della cartella mq non riesce o se la cartella non rispetta queste limitazioni, il messaggio viene rifiutato con il codice di errore 2527, MQRC_RFH_RESTRICTED_FORMAT_ERR.

Non aggiungere le proprie proprietà nella cartella mq.

mqema

mqema contiene proprietà utilizzate solo da WebSphere Application Server. La cartella è stata sostituita da mqext.

Non aggiungere le proprie proprietà nella cartella mqema.

mqext

mqext contiene i seguenti tipi di proprietà:

- Proprietà utilizzate solo da WebSphere Application Server.
- Proprietà relative al recapito ritardato dei messaggi.

La cartella è presente se l'applicazione ha impostato almeno una delle proprietà definite da IBM o ha utilizzato il ritardo di recapito.

Tabella 519. Nome, sinonimo, tipo di dati e cartella della proprietà mqext

Sinonimo proprietà	Nome proprietà	Tipo dati	Cartella
JMSArmCorrelator	mqext.Arm	string	<mqext><Arm>*armCorrelator*</Arm></mqext>
JMSRMCorrelator	mqext.Wrm	string	<mqext><Wrm>*wrmCorrelator*</Wrm></mqext>
JMSDeliveryTime	mqext.Dlt	i8	<mqext><Dlt>*DeliveryTime*</Dlt></mqext>
JMSDeliveryDelay	mqext.Dly	i8	<mqext><Dly>*DeliveryTime*</Dly></mqext>

Non aggiungere le proprie proprietà nella cartella mqext.

mqps

mqps contiene proprietà che vengono utilizzate solo dalla pubblicazione/sottoscrizione di IBM MQ. La cartella è presente solo se l'applicazione ha impostato almeno una delle proprietà di pubblicazione/sottoscrizione integrate.

Tabella 520. Nome, sinonimo, tipo di dati e cartella della proprietà mqps			
Sinonimo proprietà	**Nome proprietà**	**Tipo dati**	**Cartella**
MQTopicString	mqps.Top	string	<mqps><Top>*topicString*</Top></mqps>
MQSubUserData	mqps.Sud	string	<mqps><Sud>*subscriberUserData...*</Sud></mqps>
MQIsRetained	mqps.Ret	boolean	<mqps><Ret>*isRetained*</Ret></mqps>
MQPubOptions	mqps.Pub	i8	<mqps><Pub>*publicationOptions*</Pub></mqps>
MQPubLevel	mqps.Pbl	i8	<mqps><Pbl>*publicationLevel*</Pbl></mqps>
MQPubTime	mqpse.Pts	string	<mqps><Pts>*publicationTime*</Pts></mqps>
MQPubSeqNum	mqpse.Seq	i8	<mqps><Seq>*publicationSequenceNumber*</Seq></mqps>
MQPubStrIntData	mqpse.Sid	string	<mqps><Sid>*publicationData*</Sid></mqps>
MQPubFormat	mqpse.Pfmt	i8	<mqps><Pfmt>*messageFormat*</Pfmt></mqps>

Non aggiungere le proprie proprietà nella cartella mqps.

mq_svc

mq_svc contiene proprietà utilizzate da SupportPac MA93.

Non aggiungere le proprie proprietà nella cartella mq_svc.

mqtt

mqtt contiene proprietà utilizzate da MQ Telemetry

Tabella 521. nome proprietà mqtt, sinonimo, tipo di dati e cartella			
Sinonimo proprietà	**Nome proprietà**	**Tipo dati**	**Cartella**
	mqtt.clientId	string	<mqtt><clientId> *topicString* </clientId></mqtt>
	mqtt.qos	i4	<mqtt><qos> *qualityOfService* </qos></mqtt>
	mqtt.msgid	string	<mqtt><msgid> *messageIdentifier* </msgid></mqtt>

Non aggiungere le proprie proprietà nella cartella mqtt.

Un messaggio segmentato inserito con MQMF_SEGMENT o MQMF_SEGMENTATION_ALLOWED non può contenere un MQRFH2 con un nome cartella di proprietà non raggruppato. MQPUT ha esito negativo con codice di errore 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Coppie nome-valore

Nel diagramma di sintassi, "Coppie nome - valore" descrive il contenuto di una cartella ordinaria. Una cartella ordinaria contiene gruppi ed elementi. Un elemento è una coppia nome - valore. Un gruppo contiene elementi e altri gruppi.

In termini di strutture ad albero, gli elementi sono nodi foglia e i gruppi sono nodi interni. Un nodo interno e la cartella, che è il nodo root, possono contenere una combinazione di nodi interni e nodi foglia. Un nodo non può essere sia un nodo interno che un nodo foglia contemporaneamente; consultare Figura 2 a pagina 551.

Proprietà

Nel diagramma di sintassi, "Proprietà" descrive il contenuto di una cartella di proprietà. Una cartella di proprietà contiene gruppi e proprietà. Una proprietà è una coppia nome - valore con un attributo del tipo di dati facoltativo. Un gruppo contiene proprietà e altri gruppi.

In termini di strutture ad albero, le proprietà sono nodi foglia e i gruppi sono nodi interni. Un nodo interno e la cartella delle proprietà, che è il nodo root, possono contenere una combinazione di nodi interni e nodi foglia. Un nodo non può essere sia un nodo interno che un nodo foglia contemporaneamente; consultare Figura 2 a pagina 551.

Proprietà

Una proprietà del messaggio è una coppia nome - valore in una cartella di proprietà. Facoltativamente, può includere un attributo del tipo di dati e un attributo della proprietà; per un esempio, consultare il seguente codice. Se l'attributo del tipo di dati viene omesso, il tipo di proprietà è `string`.

```
<pf><p1 dt='i8' > value </p1></pf>
```

Il nome di una proprietà del messaggio è il suo nome percorso completo, con la sintassi XML, <> , sostituita da punti. Ad esempio, `myPropertyFolder1.myGroup1.myGroup2.myProperty1` viene associato ad una stringa *NameValueData* come riportato di seguito. La stringa è formattata per una lettura più semplice.

```
<myPropertyFolder1>
    <myGroup1>
        <myGroup2>
            <myProperty1>value</myProperty1>
        </myGroup2>
    </myGroup1>
</myPropertyFolder1>
```

Una cartella di proprietà può contenere più proprietà. Ad esempio, le proprietà in Figura 7 a pagina 558 vengono mappate alla cartella delle proprietà in Modifiche per isolare la coda di vendita in un nuovo cluster e separare le code di trasmissione del cluster gateway.

```
 myPropertyFolder1.myProperty4
myPropertyFolder1.myGroup1.myGroup2.myProperty1
myPropertyFolder1.myGroup1.myGroup2.myProperty2
myPropertyFolder1.myGroup1.myProperty3
```

Figura 7. Più proprietà con lo stesso nome root

```
<myPropertyFolder1>
    <myProperty4>value</myProperty4>
    <myGroup1>
        <myGroup2>
            <myProperty1>value</myProperty1>
            <myProperty2>value</myProperty2>
        </myGroup2>
        <myProperty3>value</myProperty3>
    </myGroup1>
</myPropertyFolder1>
```

Figura 8. Associazione di più nomi di proprietà

Nome

Un nome deve iniziare con una *Lettera* o un *Sottolineatura*. Non deve contenere un *Due punti*, non deve terminare con un *periodo* e contenere solo *lettere*, *numeri*, *caratteri di sottolineatura*, *trattini*e *punti*. I caratteri validi sono definiti nella specifica XML W3C e consistono essenzialmente in categorie Unicode `Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm,` e `Nd` ; vedere Categorie di caratteri Unicode

Il percorso completo di una coppia proprietà o nome - valore non deve interrompere la regola descritta in “Nome percorso non valido” a pagina 561. I percorsi sono limitati a 4095 byte, non devono contenere caratteri di compatibilità Unicode e non devono iniziare con la stringa XML.

Nome gruppo

Un nome gruppo ha la stessa sintassi di un nome. I nomi gruppo sono facoltativi. Le coppie nome - valore e proprietà possono essere collocate nella root di una cartella. Utilizzare i gruppi se consente di organizzare le proprietà e le coppie nome - valore.

Nome elemento

Un nome elemento ha la stessa sintassi di un nome.

valore elemento

Un valore elemento include tutti gli spazi vuoti tra la tag < *Element name* > e < */Element name* >. Non utilizzare i due caratteri < e & in un valore. Sostituire quindi con < e `&`.

Attributi di Property

Gli attributi della proprietà associano i campi del descrittore della proprietà: le associazioni sono le seguenti:

Supporto

sa (predefinito)
`MQPD_SUPPORT_OPTIONAL`

sr
`MQPD_SUPPORT_REQUIRED`

sx
`MQPD_SUPPORT_REQUIRED_IF_LOCAL`

Contesto

none (valore predefinito)
`MQPD_NO_CONTEXT`

utente
`MQPD_USER _CONTEXT`

CopyOptions

inoltro
MQPD_COPY_FORWARD

risposta
MQPD_COPY_REPLY

report
MQPD_COPY_REPORT

pubblicare
MQPD_COPY_PUBLISH

tutti
MQPD_COPY_ALL

Non utilizzare `all` in combinazione con altre opzioni.

valore predefinito
MQPD_COPY_DEFAULT

non utilizzare `default` in combinazione con altre opzioni. `default` è uguale a `forward` + `report` + `publish`.

Nessuna
MQPD_COPY_NONE

Non utilizzare none insieme ad altre opzioni.

Gli attributi della proprietà Supporto sono applicabili solo alle proprietà nella cartella `mq` .

Gli attributi della proprietà Context e CopyOptions sono applicabili a tutte le cartelle delle proprietà.

Tipi di dati

I tipi di dati MQRFH2 si associano ai tipi di proprietà del messaggio nel modo seguente:

Tabella 522. Associazioni di tipi di dati

MQRFH2 tipo di dati	**Tipo di proprietà del messaggio**
bin.hex	MQBYTE[]
boolean	MQBOOL
i1	MQINT8
i2	MQINT16
i4	MQINT32
i8	MQINT64
r4	MQFLOAT32
r8	MQFLOAT64
string	MQCHAR[]

Si presume che qualsiasi elemento senza un tipo di dati sia di tipo `string`.

Un valore nullo è indicato dall'attributo dell'elemento `xsi:nil='true'`. Non utilizzare l'attributo `xsi:nil='false'` per valori non null. Ad esempio, la seguente proprietà ha un valore null:

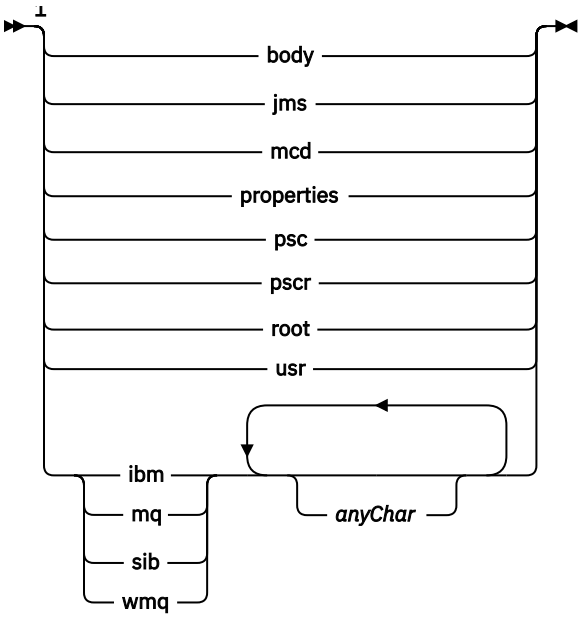
```
<NullProperty
xsi:nil='true'></NullProperty>
```

Una proprietà byte o stringa di caratteri può avere un valore vuoto. Un valore vuoto è rappresentato da un elemento MQRFH2 con un valore elemento di lunghezza zero. Ad esempio, la seguente proprietà ha un valore vuoto:

```
<EmptyProperty></EmptyProperty>
```

Nome cartella o proprietà riservata

Limitare il nome di una cartella o di una cartella di proprietà in modo che non inizi con una delle seguenti stringhe. I prefissi sono riservati per i nomi delle cartelle o delle proprietà creati da IBM.

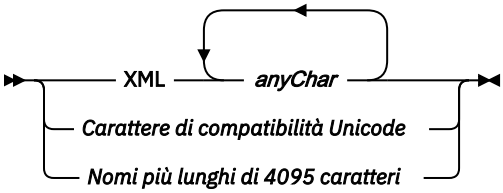


Note:

[1] Il nome di una cartella o di una proprietà riservata contiene una combinazione di lettere minuscole e maiuscole.

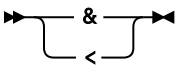
Nome percorso non valido

Limitare il percorso completo di una coppia nome - valore o di una proprietà per non includere alcuna delle seguenti stringhe.



Caratteri non validi

Utilizzare sempre le sequenze di escape & e < invece dei valori letterali "&" e "<".

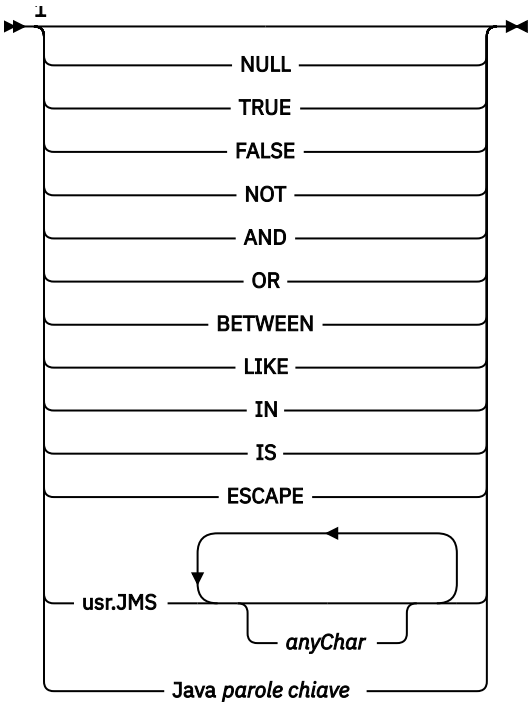


Nomi di proprietà definiti

I nomi delle proprietà definiti sono i nomi delle proprietà definite da IBM MQo da altri prodotti e utilizzati dalle applicazioni IBM MQ e utente. Le proprietà definite esistono solo nelle cartelle delle proprietà definite. I nomi di proprietà definiti sono descritti nella descrizione delle cartelle di proprietà; consultare Cartelle proprietà.

Nome proprietà non valido

Non creare nomi di proprietà che corrispondono alla seguente regola. La regola si applica al percorso completo della proprietà che denomina una proprietà e non solo al nome elemento della proprietà.



Note:

[1] Un nome proprietà non valido può contenere qualsiasi combinazione di lettere maiuscole e minuscole.

Attributi non validi

Le proprietà e le cartelle delle proprietà possono includere solo “Attributi di Property” a pagina 559 e “Tipi di dati” a pagina 560supportati.

Tutti gli attributi di tipo XML non supportati, ad esempio i nomi con valori stringa tra virgolette, inclusi nelle proprietà o nelle cartelle delle proprietà, potrebbero essere rimossi.

Attributi di tipo XML inclusi in cartelle non di proprietà o elementi non di proprietà che rimangono nelle intestazioni MQRFH2 .

MQRMH - Intestazione messaggio di riferimento

La struttura MQRMH definisce il formato di un'intestazione del messaggio di riferimento. Questa intestazione viene utilizzata con le uscite del canale messaggi scritte dall'utente per inviare grandi quantità di dati (denominati *dati di massa*) da un gestore code ad un altro. La differenza rispetto alla messaggistica normale è che i dati di massa non vengono memorizzati in coda; invece, solo un *riferimento* ai dati di massa viene memorizzato nella coda. Ciò riduce la possibilità che le risorse IBM MQ vengano esaurite da un numero ridotto di messaggi estremamente grandi.

Disponibilità

La struttura MQRMH è disponibile sulle piattaforme seguenti:

-  AIX
-  IBM i
-  Linux
-  Windows

e per i client IBM MQ connessi a questi sistemi.

Nome formato

MQFMT_REF_MSG_HEADER

Serie di caratteri e codifica

I dati di caratteri in MQRMH e le stringhe indirizzati dai campi di offset devono trovarsi nella serie di caratteri del gestore code locale; ciò viene fornito dall'attributo del gestore code **CodedCharSetId** . I dati numerici in MQRMH devono essere nella codifica della macchina nativo; ciò viene fornito dal valore di MQENC_NATIVE per il linguaggio di programmazione C.

Impostare la serie di caratteri e la codifica di MQRMH nei campi *CodedCharSetId* e *Encoding* in:

- MQMD (se la struttura MQRMH si trova all'inizio dei dati del messaggio) oppure
- La struttura dell'intestazione che precede la struttura MQRMH (tutti gli altri casi).

Utilizzo

Un'applicazione inserisce un messaggio composto da un MQRMH, ma omettendo i dati di massa. Quando un MCA (message channel agent) legge il messaggio dalla coda di trasmissione, viene richiamata un'uscita messaggio fornita dall'utente per elaborare l'intestazione del messaggio di riferimento. L'uscita può aggiungere al messaggio di riferimento i dati di massa identificati dalla struttura MQRMH, prima che l'MCA invii il messaggio attraverso il canale al gestore code successivo.

All'estremità ricevente, deve esistere un'exit dei messaggi che attende i messaggi di riferimento. Quando viene ricevuto un messaggio di riferimento, l'uscita deve creare l'oggetto dai dati di massa che seguono MQRMH nel messaggio e quindi trasmettere il messaggio di riferimento senza i dati di massa. Il messaggio di riferimento può essere successivamente richiamato da un'applicazione che legge il messaggio di riferimento (senza i dati di massa) da una coda.

Normalmente, la struttura MQRMH è tutto ciò che si trova nel messaggio. Tuttavia, se il messaggio è su una coda di trasmissione, una o più intestazioni aggiuntive precedono la struttura MQRMH.

Un messaggio di riferimento può anche essere inviato ad un elenco di distribuzione. In questo caso, la struttura MQDH e i relativi record precedono la struttura MQRMH quando il messaggio si trova su una coda di trasmissione.

Nota: Non inviare un messaggio di riferimento come messaggio segmentato, poiché l'exit dei messaggi non può elaborarlo correttamente.

Conversione dati

Ai fini della conversione dei dati, la conversione della struttura MQRMH comprende la conversione dei dati dell'ambiente di origine, del nome oggetto di origine, dei dati dell'ambiente di destinazione e del nome oggetto di destinazione. Tutti gli altri byte all'interno di *StrucLength* byte dell'inizio della struttura vengono scartati o hanno valori non definiti dopo la conversione dei dati. I dati di massa vengono convertiti a condizione che tutte le seguenti istruzioni siano vere:

- I dati di massa sono presenti nel messaggio quando viene eseguita la conversione dati.
- Il campo *Format* in MQRMH ha un valore diverso da MQFMT_NONE.
- Esiste un'uscita di conversione dati scritta dall'utente con il nome formato specificato.

Tenere presente, tuttavia, che di solito i dati di massa non sono presenti nel messaggio quando il messaggio si trova su una coda e che di conseguenza i dati di massa vengono convertiti dall'opzione MQGMO_CONVERT.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 523. Campi in MQRMH per MQRMH

Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
StrucId (identificativo della struttura)	ID_STRUC_MQRMH	'RMH¬'
Versione (numero versione struttura)	MQRMH_VERSION_1	1
StrucLength (lunghezza totale di MQRMH, incluse le stringhe alla fine di campi fissi, ma non i dati di massa)	Nessuna	0
Codifica (codifica numerica dei dati di massa)	MQEN_NATIVE	Dipende dall'ambiente
CodedCharSetId (identificativo della serie di caratteri dei dati di massa)	MQCCSI_UNDEFINED	0
Formato (nome formato dei dati di massa)	MQFMT_NONE	Spazi
Indicatori (indicatori del messaggio di riferimento)	MQRMHF_NO_LAST	0
ObjectType (tipo oggetto)	Nessuna	Spazi
ObjectInstanceObjectInstance (identificativo istanza oggetto)	MQOII_NONE	Valori null
SrcEnvLunghezza (lunghezza dei dati dell'ambiente origine)	Nessuna	0
SrcEnvOffset (offset dei dati di ambiente di origine)	Nessuna	0
SrcNameLunghezza (lunghezza del nome oggetto di origine)	Nessuna	0
SrcNameOffset (offset del nome oggetto di origine)	Nessuna	0
DestEnvLunghezza (lunghezza dei dati dell'ambiente di destinazione)	Nessuna	0
DestEnvDestEnv (offset dei dati dell'ambiente di destinazione)	Nessuna	0
DestNameLunghezza (lunghezza del nome oggetto di destinazione)	Nessuna	0
DestNameOffset (offset del nome oggetto di destinazione)	Nessuna	0
DataLogicalDataLogical (lunghezza dei dati di massa)	Nessuna	0

Tabella 523. Campi in MQRMH per MQRMH (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
DataLogicalDataLogical (offset basso dei dati di massa)	Nessuna	`0`
DataLogicalOffset2 (offset elevato di dati di massa)	Nessuna	`0`

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macroMQRMH_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQRMH MyRMH = {MQRMH_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQRMH

```
typedef struct tagMQRMH MQRMH;
struct tagMQRMH {
  MQCHAR4   StrucId;             /* Structure identifier */
  MQLONG    Version;             /* Structure version number */
  MQLONG    StrucLength;         /* Total length of MQRMH, including
                                    strings at end of fixed fields, but
                                    not the bulk data */
  MQLONG    Encoding;            /* Numeric encoding of bulk data */
  MQLONG    CodedCharSetId;      /* Character set identifier of bulk
                                    data */
  MQCHAR8   Format;              /* Format name of bulk data */
  MQLONG    Flags;               /* Reference message flags */
  MQCHAR8   ObjectType;          /* Object type */
  MQBYTE24  ObjectInstanceId;    /* Object instance identifier */
  MQLONG    SrcEnvLength;        /* Length of source environment data */
  MQLONG    SrcEnvOffset;        /* Offset of source environment data */
  MQLONG    SrcNameLength;       /* Length of source object name */
  MQLONG    SrcNameOffset;       /* Offset of source object name */
  MQLONG    DestEnvLength;       /* Length of destination environment
                                    data */
  MQLONG    DestEnvOffset;       /* Offset of destination environment
                                    data */
  MQLONG    DestNameLength;      /* Length of destination object name */
  MQLONG    DestNameOffset;      /* Offset of destination object name */
  MQLONG    DataLogicalLength;   /* Length of bulk data */
  MQLONG    DataLogicalOffset;   /* Low offset of bulk data */
  MQLONG    DataLogicalOffset2;  /* High offset of bulk data */
};
```

Dichiarazione COBOL per MQRMH

```
**   MQRMH structure
  10 MQRMH.
**    Structure identifier
   15 MQRMH-STRUCID            PIC X(4).
**    Structure version number
   15 MQRMH-VERSION            PIC S9(9) BINARY.
**    Total length of MQRMH, including strings at end of fixed fields,
**    but not the bulk data
   15 MQRMH-STRUCLENGTH        PIC S9(9) BINARY.
**    Numeric encoding of bulk data
   15 MQRMH-ENCODING           PIC S9(9) BINARY.
**    Character set identifier of bulk data
   15 MQRMH-CODEDCHARSETID     PIC S9(9) BINARY.
```

```
**    Format name of bulk data
   15 MQRMH-FORMAT             PIC X(8).
**    Reference message flags
   15 MQRMH-FLAGS              PIC S9(9) BINARY.
**    Object type
   15 MQRMH-OBJECTTYPE         PIC X(8).
**    Object instance identifier
   15 MQRMH-OBJECTINSTANCEID   PIC X(24).
**    Length of source environment data
   15 MQRMH-SRCENVLENGTH       PIC S9(9) BINARY.
**    Offset of source environment data
   15 MQRMH-SRCENVOFFSET       PIC S9(9) BINARY.
**    Length of source object name
   15 MQRMH-SRCNAMELENGTH      PIC S9(9) BINARY.
**    Offset of source object name
   15 MQRMH-SRCNAMEOFFSET      PIC S9(9) BINARY.
**    Length of destination environment data
   15 MQRMH-DESTENVLENGTH      PIC S9(9) BINARY.
**    Offset of destination environment data
   15 MQRMH-DESTENVOFFSET      PIC S9(9) BINARY.
**    Length of destination object name
   15 MQRMH-DESTNAMELENGTH     PIC S9(9) BINARY.
**    Offset of destination object name
   15 MQRMH-DESTNAMEOFFSET     PIC S9(9) BINARY.
**    Length of bulk data
   15 MQRMH-DATALOGICALLENGTH  PIC S9(9) BINARY.
**    Low offset of bulk data
   15 MQRMH-DATALOGICALOFFSET  PIC S9(9) BINARY.
**    High offset of bulk data
   15 MQRMH-DATALOGICALOFFSET2 PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQRMH

```
dcl
 1 MQRMH based,
  3 StrucId            char(4),       /* Structure identifier */
  3 Version            fixed bin(31), /* Structure version number */
  3 StrucLength        fixed bin(31), /* Total length of MQRMH,
                                         including strings at end of
                                         fixed fields, but not the bulk
                                         data */
  3 Encoding           fixed bin(31), /* Numeric encoding of bulk
                                         data */
  3 CodedCharSetId     fixed bin(31), /* Character set identifier of
                                         bulk data */
  3 Format             char(8),       /* Format name of bulk data */
  3 Flags              fixed bin(31), /* Reference message flags */
  3 ObjectType         char(8),       /* Object type */
  3 ObjectInstanceId   char(24),      /* Object instance identifier */
  3 SrcEnvLength       fixed bin(31), /* Length of source environment
                                         data */
  3 SrcEnvOffset       fixed bin(31), /* Offset of source environment
                                         data */
  3 SrcNameLength      fixed bin(31), /* Length of source object name */
  3 SrcNameOffset      fixed bin(31), /* Offset of source object name */
  3 DestEnvLength      fixed bin(31), /* Length of destination
                                         environment data */
  3 DestEnvOffset      fixed bin(31), /* Offset of destination
                                         environment data */
  3 DestNameLength     fixed bin(31), /* Length of destination object
                                         name */
  3 DestNameOffset     fixed bin(31), /* Offset of destination object
                                         name */
  3 DataLogicalLength  fixed bin(31), /* Length of bulk data */
  3 DataLogicalOffset  fixed bin(31), /* Low offset of bulk data */
  3 DataLogicalOffset2 fixed bin(31); /* High offset of bulk data */
```

Dichiarazione High Level Assembler per MQRMH

```
MQRMH                   DSECT
MQRMH_STRUCID           DS   CL4   Structure identifier
MQRMH_VERSION           DS   F     Structure version number
MQRMH_STRUCLENGTH       DS   F     Total length of MQRMH, including
*                                  strings at end of fixed fields, but
*                                  not the bulk data
MQRMH_ENCODING          DS   F     Numeric encoding of bulk data
MQRMH_CODEDCHARSETID    DS   F     Character set identifier of bulk
```

```
*                                    data
MQRMH_FORMAT              DS   CL8   Format name of bulk data
MQRMH_FLAGS               DS   F     Reference message flags
MQRMH_OBJECTTYPE          DS   CL8   Object type
MQRMH_OBJECTINSTANCEID    DS   XL24  Object instance identifier
MQRMH_SRCENVLENGTH        DS   F     Length of source environment data
MQRMH_SRCENVOFFSET        DS   F     Offset of source environment data
MQRMH_SRCNAMELENGTH       DS   F     Length of source object name
MQRMH_SRCNAMEOFFSET       DS   F     Offset of source object name
MQRMH_DESTENVLENGTH       DS   F     Length of destination environment
*                                    data
MQRMH_DESTENVOFFSET       DS   F     Offset of destination environment
*                                    data
MQRMH_DESTNAMELENGTH      DS   F     Length of destination object name
MQRMH_DESTNAMEOFFSET      DS   F     Offset of destination object name
MQRMH_DATALOGICALLENGTH   DS   F     Length of bulk data
MQRMH_DATALOGICALOFFSET   DS   F     Low offset of bulk data
MQRMH_DATALOGICALOFFSET2  DS   F     High offset of bulk data
*
MQRMH_LENGTH              EQU  *-MQRMH
                          ORG  MQRMH
MQRMH_AREA                DS   CL(MQRMH_LENGTH)
```

Dichiarazione Visual Basic per MQRMH

```
Type MQRMH
  StrucId            As String*4 'Structure identifier'
  Version            As Long     'Structure version number'
  StrucLength        As Long     'Total length of MQRMH, including'
                                 'strings at end of fixed fields, but'
                                 'not the bulk data'
  Encoding           As Long     'Numeric encoding of bulk data'
  CodedCharSetId     As Long     'Character set identifier of bulk data'
  Format             As String*8 'Format name of bulk data'
  Flags              As Long     'Reference message flags'
  ObjectType         As String*8 'Object type'
  ObjectInstanceId   As MQBYTE24 'Object instance identifier'
  SrcEnvLength       As Long     'Length of source environment data'
  SrcEnvOffset       As Long     'Offset of source environment data'
  SrcNameLength      As Long     'Length of source object name'
  SrcNameOffset      As Long     'Offset of source object name'
  DestEnvLength      As Long     'Length of destination environment'
                                 'data'
  DestEnvOffset      As Long     'Offset of destination environment'
                                 'data'
  DestNameLength     As Long     'Length of destination object name'
  DestNameOffset     As Long     'Offset of destination object name'
  DataLogicalLength  As Long     'Length of bulk data'
  DataLogicalOffset  As Long     'Low offset of bulk data'
  DataLogicalOffset2 As Long     'High offset of bulk data'
End Type
```

StrucId (MQCHAR4) per MQRMH

Questo è l'identificativo della struttura dell'intestazione del messaggio di riferimento. È sempre un campo di immissione. Il valore è MQRMH_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQRMH
Identificativo per la struttura dell'intestazione del messaggio di riferimento.

Per il linguaggio di programmazione C, viene definita anche la costante MQRMH_STRUC_ID_ARRAY. Ha lo stesso valore di MQRMH_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQRMH

Il numero di versione della struttura. Il valore deve essere:

MQRMH_VERSION_1
Version-1 fa riferimento alla struttura dell'intestazione del messaggio.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQRMH_CURRENT_
La versione corrente della struttura dell'intestazione del messaggio di riferimento.

Il valore iniziale di questo campo è MQRMH_VERSION_1.

StrucLength (MQLONG) per MQRMH
La lunghezza totale di MQRMH, incluse le stringhe alla fine dei campi fissi, ma non i dati di massa.

Il valore iniziale di questo campo è zero.

Codifica (MQLONG) per MQRMH

Specifica la codifica numerica dei dati di massa; non si applica ai dati numerici nella stessa struttura MQRMH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati.

Il valore iniziale di questo campo è MQENC_NATIVE.

CodedCharSetId (MQLONG) per MQRMH

Specifica l'identificativo della serie di caratteri dei dati di massa; non viene applicato ai dati di carattere nella stessa struttura MQRMH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. È possibile utilizzare il seguente valore speciale:

MQCCSI_INHERIT
I dati carattere nei dati che seguono questa struttura si trovano nella stessa serie di caratteri di questa struttura.

Il gestore code modifica questo valore nella struttura inviata nel messaggio nell'effettivo identificativo della serie di caratteri della struttura. Se non si verifica alcun errore, il valore MQCCSI_INHERIT non viene restituito dalla chiamata MQGET.

Non utilizzare MQCCSI_INHERIT se il valore del campo `PutApplType` in MQMD è MQAT_BROKER.

Questo valore è supportato nei seguenti ambienti:

- AIX
- IBM i
- Linux
- Windows

e per i client IBM MQ connessi a questi sistemi.

Il valore iniziale di questo campo è MQCCSI_UNDEFINED.

Formato (MQCHAR8) per MQRMH

Specifica il nome del formato dei dati di massa.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono le stesse del campo *`Format`* in MQMD.

Il valore iniziale di questo campo è MQFMT_NONE.

Indicatori (MQLONG) per MQRMH

Questi sono indicatori di messaggi di riferimento. Sono definiti i seguenti indicatori:

MQRMHF_LAST
Questo indicatore indica che il messaggio di riferimento rappresenta o contiene l'ultima parte dell'oggetto di riferimento.

MQRMHF_NO_LAST
Il messaggio di riferimento non contiene o rappresenta l'ultima parte dell'oggetto. MQRMHF_NOT_LAST aiuta la documentazione del programma. Non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Il valore iniziale di questo campo è MQRMHF_NOT_LAST.

ObjectType (MQCHAR8) per MQRMH

Questo è un nome che l'uscita del messaggio può utilizzare per riconoscere i tipi di messaggio di riferimento che supporta. Il nome deve essere conforme alle stesse regole del campo *Format* , consultare "Formato (MQCHAR8) per MQRMH" a pagina 568.

Il valore iniziale di questo campo è di 8 spazi.

ID ObjectInstance(MQBYTE24) per MQRMH

Utilizzare questo campo per identificare una specifica istanza di oggetto. Se non è necessario, impostarlo sul seguente valore:

MQOII_NONE
Nessun identificativo istanza oggetto specificato. Il valore è zero binario per la lunghezza del campo.

Per il linguaggio di programmazione C, è definita anche la costante MQOII_NONE_ARRAY; ha lo stesso valore di MQOII_NONE, ma è un array di caratteri invece di una stringa.

La lunghezza di questo campo è fornita da MQ_OBJECT_INSTANCE_ID_LENGTH. Il valore iniziale di questo campo è MQOII_NONE.

Lunghezza SrcEnv(MQLONG) per MQRMH

La lunghezza dei dati dell'ambiente di origine. Se questo campo è zero, non ci sono dati di ambiente di origine e *SrcEnvOffset* viene ignorato.

Il valore iniziale di questo campo è 0.

Offset SrcEnv(MQLONG)

Questo campo specifica l'offset dei dati dell'ambiente origine dall'inizio della struttura MQRMH. I dati dell'ambiente di origine possono essere specificati dal creatore del messaggio di riferimento, se tali dati sono noti al creatore. Ad esempio, su Windows i dati dell'ambiente di origine potrebbero essere il percorso di directory dell'oggetto contenente i dati di massa. Tuttavia, se il creatore non conosce i dati dell'ambiente di origine, l'uscita del messaggio fornita dall'utente deve determinare le informazioni sull'ambiente necessarie.

La lunghezza dei dati dell'ambiente di origine è fornita da *SrcEnvLength* ; se questa lunghezza è zero, non vi sono dati dell'ambiente di origine e *SrcEnvOffset* viene ignorato. Se presenti, i dati dell'ambiente di origine devono trovarsi completamente all'interno di *StrucLength* byte dall'inizio della struttura.

Le applicazioni non devono presumere che i dati di ambiente vengano avviati immediatamente dopo l'ultimo campo fisso nella struttura o che siano contigui con uno qualsiasi dei dati indirizzati dai campi *SrcNameOffset*, *DestEnvOffset*e *DestNameOffset* .

Il valore iniziale di questo campo è 0.

Lunghezza SrcName(MQLONG) per MQRMH

La lunghezza del nome oggetto di origine. Se questo campo è zero, non vi è alcun nome oggetto di origine e *SrcNameOffset* viene ignorato.

Il valore iniziale di questo campo è 0.

SrcNameOffset (MQLONG) per MQRMH

Questo campo specifica lo scostamento del nome oggetto di origine dall'inizio della struttura MQRMH. Il nome dell'oggetto di origine può essere specificato dal creatore del messaggio di riferimento, se tali dati sono noti al creatore. Tuttavia, se il creatore non conosce il nome dell'oggetto di origine, l'uscita del messaggio fornito dall'utente deve identificare l'oggetto a cui accedere.

La lunghezza del nome oggetto di origine è fornita da *SrcNameLength* ; se questa lunghezza è zero, non esiste alcun nome oggetto di origine e *SrcNameOffset* viene ignorato. Se presente, il nome dell'oggetto di origine deve trovarsi completamente all'interno di *StrucLength* byte dall'inizio della struttura.

Le applicazioni non devono presumere che il nome dell'oggetto di origine sia contiguo con i dati indirizzati dai campi *SrcEnvOffset*, *DestEnvOffset*e *DestNameOffset* .

Il valore iniziale di questo campo è 0.

DestEnvLunghezza (MQLONG) per MQRMH

Questa è la lunghezza dei dati dell'ambiente di destinazione. Se questo campo è zero, non ci sono dati di ambiente di destinazione e *DestEnvOffset* viene ignorato.

Offset DestEnv(MQLONG) per MQRMH

Questo campo specifica l'offset dei dati di ambiente di destinazione dall'inizio della struttura MQRMH. I dati di ambiente di destinazione possono essere specificati dal creatore del messaggio di riferimento, se tali dati sono noti al creatore. Ad esempio, su Windows i dati dell'ambiente di destinazione potrebbero essere il percorso di directory dell'oggetto in cui devono essere memorizzati i dati di massa. Tuttavia, se il creatore non conosce i dati dell'ambiente di destinazione, è responsabilità dell'uscita messaggi fornita dall'utente determinare le informazioni sull'ambiente necessarie.

La lunghezza dei dati dell'ambiente di destinazione è fornita da *DestEnvLength* ; se questa lunghezza è zero, non vi sono dati di ambiente di destinazione e *DestEnvOffset* viene ignorato. Se presenti, i dati dell'ambiente di destinazione devono trovarsi completamente all'interno di *StrucLength* byte dall'inizio della struttura.

Le applicazioni non devono presumere che i dati dell'ambiente di destinazione siano contigui con i dati indirizzati dai campi *SrcEnvOffset*, *SrcNameOffset*e *DestNameOffset* .

Il valore iniziale di questo campo è 0.

Lunghezza DestName(MQLONG) per MQRMH

La lunghezza del nome oggetto di destinazione. Se questo campo è zero, non c'è alcun nome oggetto di destinazione e *DestNameOffset* viene ignorato.

Offset DestName(MQLONG) per MQRMH

Questo campo specifica lo scostamento del nome oggetto di destinazione dall'inizio della struttura MQRMH. Il nome dell'oggetto di destinazione può essere specificato dal creatore del messaggio di riferimento, se tali dati sono noti al creatore. Tuttavia, se il creatore non conosce il nome dell'oggetto di destinazione, è responsabilità dell'uscita messaggi fornita dall'utente identificare l'oggetto da creare o modificare.

La lunghezza del nome oggetto di destinazione è fornita da *DestNameLength* ; se questa lunghezza è zero, non esiste alcun nome oggetto di destinazione e *DestNameOffset* viene ignorato. Se presente, il nome dell'oggetto di destinazione deve trovarsi completamente all'interno di *StrucLength* byte dall'inizio della struttura.

Le applicazioni non devono presumere che il nome dell'oggetto di destinazione sia contiguo ai dati indirizzati dai campi *SrcEnvOffset*, *SrcNameOffset*e *DestEnvOffset* .

Il valore iniziale di questo campo è 0.

DataLogicalLength (MQLONG) per MQRMH

Il campo *DataLogicalLength* specifica la lunghezza dei dati di massa a cui fa riferimento la struttura MQRMH.

Se i dati di massa sono effettivamente presenti nel messaggio, i dati iniziano con un offset di *StrucLength* byte dall'inizio della struttura MQRMH. La lunghezza dell'intero messaggio meno *StrucLength* fornisce la lunghezza dei dati di massa presenti.

Se i dati sono presenti nel messaggio, *DataLogicalLength* specifica la quantità di tali dati rilevante. Il caso normale è che *DataLogicalLength* abbia lo stesso valore della lunghezza dei dati presenti nel messaggio.

Se la struttura MQRMH rappresenta i dati rimanenti nell'oggetto (a partire dall'offset logico specificato), è possibile utilizzare il valore zero per *DataLogicalLength*, purché i dati di massa non siano effettivamente presenti nel messaggio.

Se non è presente alcun dato, la fine di MQRMH coincide con la fine del messaggio.

Il valore iniziale di questo campo è 0.

Offset DataLogical(MQLONG) per MQRMH

Questo campo specifica l'offset minimo dei dati di massa dall'inizio dell'oggetto di cui fanno parte i dati di massa. L'offset dei dati di massa dall'inizio dell'oggetto è denominato *offset logico*. Questo non è l'offset fisico dei dati di massa dall'inizio della struttura MQRMH; tale offset è fornito da *StrucLength*.

Per consentire l'invio di oggetti di grandi dimensioni utilizzando messaggi di riferimento, lo scostamento logico è diviso in due campi e lo scostamento logico effettivo è dato dalla somma di questi due campi:

- *DataLogicalOffset* rappresenta il resto ottenuto quando l'offset logico è diviso per 1 000 000 000. È quindi un valore compreso nell'intervallo tra 0 e 999 999 999.
- *DataLogicalOffset2* rappresenta il risultato ottenuto quando l'offset logico è diviso per 1 000 000 000. È quindi il numero di multipli completi di 1 000 000 000 che esistono nell'offset logico. Il numero di multipli è compreso tra 0 e 999 999 999.

Il valore iniziale di questo campo è 0.

DataLogicalOffset2 (MQLONG) per MQRMH

Questo campo specifica l'offset elevato dei dati di massa dall'inizio dell'oggetto di cui fanno parte i dati di massa. È un valore compreso tra 0 e 999 999 999. Consultare *DataLogicalOffset* per i dettagli.

Il valore iniziale di questo campo è 0.

MQRR - Record risposta

Utilizzare la struttura MQRR per ricevere il codice di completamento e il codice motivo risultanti dall'operazione di apertura o di inserimento per una singola coda di destinazione, quando la destinazione è un elenco di distribuzione. MQRR è una struttura di output per le chiamate MQOPEN, MQPUT e MQPUT1 .

Disponibilità

La struttura MQRR è disponibile sulle piattaforme seguenti:

- AIX
- IBM i
- Linux
- Windows

e per i client IBM MQ connessi a questi sistemi.

Serie di caratteri e codifica

I dati in MQRR devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da MQENC_NATIVE. Tuttavia, se l'applicazione è in esecuzione come client MQ MQI, la struttura deve essere nella serie di caratteri e nella codifica del client.

Utilizzo

Fornendo un array di queste strutture sulle chiamate MQOPEN e MQPUT o sulla chiamata MQPUT1 , è possibile determinare i codici di completamento e i codici motivo per tutte le code in un elenco di distribuzione quando il risultato della chiamata è misto, ossia quando la chiamata ha esito positivo per alcune code nell'elenco ma ha esito negativo per altre. Il codice motivo MQRC_MULTIPLE_REASON dalla chiamata indica che i record di risposta (se forniti dall'applicazione) sono stati impostati dal gestore code.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 524. Campi in MQRR

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
CompCode (codice di completamento per la coda)	MQCC_OK	0
Motivo (codice motivo per la coda)	MQRC_NONE	0

Note:

1. Nel linguaggio di programmazione C, la variabile macroMQRR_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQRR MyRR = {MQRR_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQRR

```
typedef struct tagMQRR MQRR;
struct tagMQRR {
  MQLONG  CompCode;  /* Completion code for queue */
  MQLONG  Reason;    /* Reason code for queue */
};
```

Dichiarazione COBOL per MQRR

```
**   MQRR structure
  10 MQRR.
**    Completion code for queue
   15 MQRR-COMPCODE PIC S9(9) BINARY.
**    Reason code for queue
   15 MQRR-REASON   PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQRR

```
dcl
```

```
 1 MQRR based,
  3 CompCode fixed bin(31), /* Completion code for queue */
  3 Reason   fixed bin(31); /* Reason code for queue */
```

Dichiarazione Visual Basic per MQRR

```
Type MQRR
  CompCode As Long 'Completion code for queue'
  Reason   As Long 'Reason code for queue'
End Type
```

CompCode (MQLONG) per MQRR

Questo è il codice di completamento risultante dall'operazione di apertura o di inserimento per la coda con il nome specificato dall'elemento corrispondente nell'array di strutture MQOR fornito nella chiamata MQOPEN o MQPUT1 .

Questo è sempre un campo di output. Il valore iniziale di questo campo è MQCC_OK.

Motivo (MQLONG) per MQRR

Questo è il codice di errore risultante dall'operazione di apertura o di inserimento per la coda con il nome specificato dall'elemento corrispondente nell'array di strutture MQOR fornito nella chiamata MQOPEN o MQPUT1 .

Questo è sempre un campo di output. Il valore iniziale di questo campo è MQRC_NONE.

MQSCO - Opzioni di configurazione SSL/TLS

La struttura MQSCO, insieme ai campi TLS nella struttura MQCD, consente a un'applicazione in esecuzione come IBM MQ MQI client di specificare le opzioni di configurazione che controllano l'utilizzo di TLS per la connessione client quando il protocollo del canale è TCP/IP. La struttura è un parametro di input sulla chiamata MQCONNX.

Disponibilità

La struttura MQSCO è disponibile sui client seguenti:

- AIX
- IBM i
- Linux
- Windows

Se il protocollo del canale per il canale client non è TCP/IP, la struttura MQSCO viene ignorata.

Serie di caratteri e codifica

I dati in MQSCO devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e nella codifica del gestore code locale fornita da MQENC_NATIVE.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 525. Campi in MQSCO		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
StrucId (identificativo della struttura)	MQSCO_STRUC_ID	'SCO¬'
Versione (numero versione struttura)	MQSCO_CURRENT_VERSION	1
KeyRepository (ubicazione del repository chiavi)	Nessuna	Stringa null o spazi vuoti
CryptoHardware (dettagli dell'hardware crittografico)	Nessuna	Stringa null o spazi vuoti
AuthInfoRecCount (numero di record MQAIR presenti)	Nessuna	0
AuthInfoRecOffset (offset del primo record MQAIR dall'inizio di MQSCO)	Nessuna	0
AuthInfoRecPtr (indirizzo del primo record MQAIR)	Nessuna	Puntatore null o byte null
Nota: I seguenti due campi vengono ignorati se *Version* è minore di MQSCO_VERSION_2.		
KeyResetCount (conteggio reimpostazioni chiave segreta TLS)	MQSCO_RESET_COUNT_DEFAULT	0
"FipsRequired (MQLONG) per MQSCO" a pagina 579 (utilizzare algoritmi di crittografia certificati FIPS in IBM MQ)	MQSSL_FIPS_NO	0
Nota: I seguenti due campi vengono ignorati se *Version* è minore di MQSCO_VERSION_3.		
EncryptionPolicySuiteB (utilizzare solo algoritmi di crittografia Suite B)	MQ_SUITE_B_NONE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE	1, 0, 0, 0
Nota: I seguenti due campi vengono ignorati se *Version* è minore di MQSCO_VERSION_4.		
Politica CertificateVal (politica di convalida del certificato)	MQ_CERT_VAL_POLICY_DEFAULT	0
Nota: I seguenti due campi vengono ignorati se *Version* è minore di MQSCO_VERSION_5.		
CertificateLabel (dettagli sull'etichetta di certificato utilizzata)	Nessuna	Stringa null o spazi vuoti
Nota: I restanti campi vengono ignorati se *Version* è minore di MQSCO_VERSION_6.		
KeyRepoPasswordPtr (indirizzo della password del repository delle chiavi TLS)	Nessuna	Puntatore null o byte null
KeyRepoPasswordOffset (offset della password del repository delle chiavi TLS)	Nessuna	0

Tabella 525. Campi in MQSCO (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
KeyRepoPasswordLength (lunghezza della password del repository chiavi TLS)	Nessuna	0

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macroMQSCO_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQSCO MySCO = {MQSCO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQSCO

```
typedef struct tagMQSCO MQSCO;
struct tagMQSCO {
   MQCHAR4    StrucId;                   /* Structure identifier */
   MQLONG     Version;                   /* Structure version number */
   MQCHAR256  KeyRepository;             /* Location of TLS key */
                                         /* repository */
   MQCHAR256  CryptoHardware;            /* Cryptographic hardware */
                                         /* configuration string */
   MQLONG     AuthInfoRecCount;          /* Number of MQAIR records */
                                         /* present */
   MQLONG     AuthInfoRecOffset;         /* Offset of first MQAIR */
                                         /* record from start of */
                                         /* MQSCO structure */
   PMQAIR     AuthInfoRecPtr;            /* Address of first MQAIR */
                                         /* record */
/* Ver:1 */
   MQLONG     KeyResetCount;             /* Number of unencrypted */
                                         /* bytes sent/received */
                                         /* before secret key is */
                                         /* reset */
   MQLONG     FipsRequired;              /* Using FIPS-certified */
/* Ver:2 */
                                         /* algorithms */
   MQLONG     EncryptionPolicySuiteB[4]; /* Use only Suite B */
/* Ver:3 */
                                         /* cryptographic algorithms */
   MQLONG     CertificateValPolicy;      /* Certificate validation */
                                         /* policy */
/* Ver:4 */
   MQCHAR64   CertificateLabel;          /* Certificate label */
/* Ver:5 */
   MQPTR      KeyRepoPasswordPtr;        /* Address of key */
                                         /* repository password */
   MQLONG     KeyRepoPasswordOffset;     /* Offset of key repository */
                                         /* password */
   MQLONG     KeyRepoPasswordLength;     /* Length of key repository */
                                         /* password */
/* Ver:6 */
  };
```

Dichiarazione COBOL per MQSCO

```
**   MQSCO structure
  10 MQSCO.
**    Structure identifier
   15 MQSCO-STRUCID                 PIC X(4).
**    Structure version number
   15 MQSCO-VERSION                 PIC S9(9) BINARY.
```

```
**    Location of TLS key repository
   15 MQSCO-KEYREPOSITORY            PIC X(256).
**    Cryptographic hardware configuration string
   15 MQSCO-CRYPTOHARDWARE           PIC X(256).
**    Number of MQAIR records present
   15 MQSCO-AUTHINFORECCOUNT         PIC S9(9) BINARY.
**    Offset of first MQAIR record from start of MQSCO structure
   15 MQSCO-AUTHINFORECOFFSET        PIC S9(9) BINARY.
**    Address of first MQAIR record
   15 MQSCO-AUTHINFORECPTR           POINTER.
** Version 1 **
**    Number of unencrypted bytes sent/received before secret key is
**    reset
   15 MQSCO-KEYRESETCOUNT            PIC S9(9) BINARY.
**    Using FIPS-certified algorithms
   15 MQSCO-FIPSREQUIRED             PIC S9(9) BINARY.
** Version 2 **
**    Use only Suite B cryptographic algorithms
   15 MQSCO-ENCRYPTIONPOLICYSUITEB   PIC S9(9) BINARY OCCURS 4.
** Version 3 **
**    Certificate validation policy setting
   15 MQSCO-CERTIFICATEVALPOLICY     PIC S9(9) BINARY.
** Version 4 **
**    SSL/TLS certificate label
   15 MQSCO-CERTIFICATELABEL         PIC X(64).
** Version 5 **
**    Add padding to ensure that pointers start on correct
**    boundaries
   15 FILLER                         PIC S9(9) BINARY VALUE 0.
**   Address of key repository password
   15 MQSCO-KEYREPOPASSWORDPTR       POINTER.
**   Offset of key repository password
   15 MQSCO-KEYREPOPASSWORDOFFSET    PIC S9(9) BINARY.
**   Length of key repository password
   15 MQSCO-KEYREPOPASSWORDLENGTH    PIC S9(9) BINARY.
** Version 6 **
```

Dichiarazione PL/I per MQSCO

```
dcl
 1 MQSCO based,
  3 StrucId                   char(4),       /* Structure identifier */
  3 Version                   fixed bin(31), /* Structure version number */
  3 KeyRepository             char(256),     /* Location of TLS key
                                                repository */
  3 CryptoHardware            char(256),     /* Cryptographic hardware
                                                configuration string */
  3 AuthInfoRecCount          fixed bin(31), /* Number of MQAIR records
                                                present */
  3 AuthInfoRecOffset         fixed bin(31), /* Offset of first MQAIR record
                                                from start of MQSCO structure */
  3 AuthInfoRecPtr            pointer,       /* Address of first MQAIR record */
  3 KeyResetCount             fixed bin(31), /* Key reset count */
/* Version 1 */
  3 FipsRequired              fixed bin(31), /* FIPS required */
/* Version 2 */
  3 EncryptionPolicySuiteB (4) fixed bin(31), /* Suite B encryption policy */
/* Version 3 */
  3 CertificateValPolicy      fixed bin(31), /* Certificate validation policy */
/* Version 4 */
  3 CertificateLabel          char(64),      /* SSL/TLS certificate label */
/* Version 5 */
  3 KeyRepoPasswordPtr        pointer,       /* Address of key repository
                                                password */
  3 KeyRepoPasswordOffset     fixed bin(31), /* Offset of key repository
                                                password */
  3 KeyRepoPasswordLength     fixed bin(31); /* Length of key repository
                                                password */
/* Version 6 */
```

Dichiarazione Visual Basic per MQSCO

```
Type MQSCO
  StrucId           As String*4   'Structure identifier'
  Version           As Long       'Structure version number'
  KeyRepository     As String*256 'Location of TLS key repository'
  CryptoHardware    As String*256 'Cryptographic hardware configuration'
```

```
                                    'string'
  AuthInfoRecCount  As Long         'Number of MQAIR records present'
  AuthInfoRecOffset As Long         'Offset of first MQAIR record from'
                                    'start of MQSCO structure'
  AuthInfoRecPtr    As MQPTR        'Address of first MQAIR record'
  KeyResetCount     As Long         'Number of unencrypted bytes sent/received before secret key
is reset'
'Version 1'
  FipsRequired      As Long         'Mandatory FIPS CipherSpecs?'
'Version 2'
End Type
```

Riferimenti correlati

“MQCNO - Opzioni di connessione” a pagina 321

La struttura MQCNO consente all'applicazione di specificare le opzioni relative alla connessione al gestore code. La struttura è un parametro di input / output sulla chiamata MQCONNX.

StrucId (MQCHAR4) per MQSCO

Si tratta dell'identificatore della struttura delle opzioni di configurazione SSL/TLS. È sempre un campo di immissione. Il valore è MQSCO_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQSCO
Identificativo per la struttura delle opzioni di configurazione SSL/TLS.

Per il linguaggio di programmazione C, viene definita anche la costante MQSCO_STRUC_ID_ARRAY. Ha lo stesso valore di MQSCO_STRUC_ID, ma è un array di caratteri anziché una stringa.

Versione (MQLONG) per MQSCO

Questo è il numero di versione della struttura; il valore deve essere:

MQSCO_VERSION_1
Version-1 Struttura delle opzioni di configurazione TLS.

MQSCO_VERSION_2
Version-2 Struttura delle opzioni di configurazione TLS.

MQSCO_VERSION_3
Version-3 Struttura delle opzioni di configurazione TLS.

MQSCO_VERSION_4
Version-4 Struttura delle opzioni di configurazione TLS.

MQSCO_VERSION_5
Version-5 Struttura delle opzioni di configurazione TLS.

MQSCO_VERSION_6
Version-6 Struttura delle opzioni di configurazione TLS.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQSCO_CURRENT_
La versione corrente della struttura delle opzioni di configurazione TLS.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQSCO_VERSION_1.

Multi *KeyRepository (MQCHAR256) per MQSCO*

Questo campo è rilevante solo per IBM MQ MQI clients in esecuzione su IBM i, AIX, Linux, and Windows sistemi. Specifica l'ubicazione del file database di chiavi in cui sono memorizzate le chiavi e i certificati. Se il suffisso del file non è specificato, viene aggiunto automaticamente un suffisso `.kdb` .

Ogni file di database delle chiavi può avere un *file stash delle password*associato. Il file stash contiene le password codificate utilizzate per consentire l'accesso programmatico al database delle chiavi. Il file stash delle password deve risiedere nella stessa directory e avere lo stesso file system del database delle chiavi e deve terminare con il suffisso `.sth`.

Ad esempio, se il file del database delle chiavi è `/xxx/yyy/key.kdb`, il file stash delle parole d'ordine deve essere `/xxx/yyy/key.sth`, dove `xxx` e `yyy` rappresentano i nomi delle directory.

La password del database delle chiavi può essere specificata anche utilizzando il campo *KeyRepoPasswordPtr* o *KeyRepoPasswordOffset* della struttura MQSCO.

Se il valore è più breve della lunghezza del campo, terminare il valore con un carattere null o riempirlo con spazi vuoti fino alla lunghezza del campo. Il valore non viene selezionato; se si verifica un errore nell'accesso al repository delle chiavi, la chiamata non riesce con codice di errore MQRC_KEY_REPOSITORY_ERROR.

Per eseguire una connessione TLS da un IBM MQ MQI client, impostare *KeyRepository* su un nome file database di chiavi valido.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da MQ_SSL_KEY_REPOSITORY_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e gli spazi in altri linguaggi di programmazione.

CryptoHardware (MQCHAR256) per MQSCO

Questo campo fornisce i dettagli di configurazione per l'hardware crittografico collegato al sistema client.

Impostare il campo su una stringa del seguente formato oppure lasciarlo vuoto o null:

```
GSK_PKCS11=the PKCS #11 driver path and file name;the PKCS #11
token label;the PKCS #11 token password;symmetric cipher setting;
```

Per utilizzare l'hardware crittografico conforme all'interfaccia PKCS #11 , ad esempio, è necessario specificare IBM 4960 o IBM 4764, il percorso del driver PKCS #11 , l'etichetta del token PKCS #11 e le stringhe della password del token PKCS #11 , ciascuna delle quali termina con un punto e virgola.

Il percorso del driver PKCS #11 è un percorso assoluto della libreria condivisa che fornisce supporto per la scheda PKCS #11 . Il nome file del driver PKCS #11 è il nome della libreria condivisa. Un esempio del valore richiesto per il percorso e il nome file PKCS #11 è:

```
/usr/lib/pkcs11/PKCS11_API.so
```

L'etichetta token PKCS #11 deve corrispondere all'etichetta con cui è stato configurato l'hardware.

Se non è richiesta alcuna configurazione hardware crittografica, impostare il campo su uno spazio vuoto o null.

Se il valore è più breve della lunghezza del campo, terminare il valore con un carattere null o riempirlo con spazi vuoti fino alla lunghezza del campo. Se il valore non è valido o causa un errore quando viene utilizzato per la configurazione dell'hardware crittografico, la chiamata ha esito negativo con codice di errore MQRC_CRYPTO_HARDWARE_ERROR.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da MQ_SSL_CRYPTO_HARDWARE_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e gli spazi in altri linguaggi di programmazione.

AuthInfoRecCount (MQLONG) per MQSCO

Questo è il numero di record di informazioni di autenticazione (MQAIR) indirizzati dai campi *AuthInfoRecPtr* o *AuthInfoRecOffset* . Per ulteriori informazioni, consultare“MQAIR - Record informazioni di autenticazione” a pagina 274. Il valore deve essere maggiore o uguale a zero. Se il valore non è valido, la chiamata ha esito negativo con codice motivo MQRC_AUTH_INFO_REC_COUNT_ERROR.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

AuthInfoRecOffset (MQLONG) per MQSCO

Questo è lo scostamento, in byte, del primo record delle informazioni di autenticazione dall'inizio della struttura MQSCO. L'offset può essere positivo o negativo. Il campo viene ignorato se *AuthInfoRecCount* è zero.

È possibile utilizzare *AuthInfoRecOffset* o *AuthInfoRecPtr* per specificare i record MQAIR, ma non entrambi; consultare la descrizione del campo *AuthInfoRecPtr* per i dettagli.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

AuthInfoRecPtr (PMQAIR) per MQSCO

Questo è l'indirizzo del primo record di informazioni di autenticazione. Il campo viene ignorato se *AuthInfoRecCount* è zero.

È possibile fornire l'array di record MQAIR in uno dei seguenti due modi:

- Utilizzando il campo puntatore *AuthInfoRecPtr*

  In questo caso, l'applicazione può dichiarare un array di record MQAIR separato dalla struttura MQSCO e impostare *AuthInfoRecPtr* sull'indirizzo dell'array.

  Considerare l'utilizzo di *AuthInfoRecPtr* per i linguaggi di programmazione che supportano il tipo di dati del puntatore in un modo che sia portabile in ambienti differenti (ad esempio, il linguaggio di programmazione C).
- Utilizzando il campo offset *AuthInfoRecOffset*

  In questo caso, l'applicazione deve dichiarare una struttura composta contenente un MQSCO seguito dall'array di record MQAIR e impostare *AuthInfoRecOffset* sull'offset del primo record nell'array dall'inizio della struttura MQSCO. Verificare che questo valore sia corretto e che abbia un valore che possa essere utilizzato all'interno di un MQLONG (il linguaggio di programmazione più restrittivo è COBOL, per cui l'intervallo valido è compreso tra -999 999 999 e +999 999 999).

  Considerare l'utilizzo di *AuthInfoRecOffset* per linguaggi di programmazione che non supportano il tipo di dati puntatore o che implementano il tipo di dati puntatore in un modo non portabile in ambienti differenti (ad esempio, il linguaggio di programmazione COBOL).

Indipendentemente dalla tecnica scelta, è possibile utilizzare solo uno tra *AuthInfoRecPtr* e *AuthInfoRecOffset* ; la chiamata ha esito negativo con codice motivo MQRC_AUTH_INFO_REC_ERROR se entrambi sono diversi da zero.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e, in caso contrario, una stringa di byte completamente null.

Nota: Su piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata.

Conteggio KeyReset(MQLONG) per MQSCO

Rappresenta il numero totale di byte non codificati inviati e ricevuti all'interno di una conversazione TLS prima che la chiave segreta venga rinegoziata.

Il numero di byte include le informazioni di controllo inviate da MCA.

Se si specifica un conteggio di reimpostazione della chiave segreta TLS compreso tra 1 byte e 32 KB, i canali TLS utilizzeranno un conteggio di reimpostazione della chiave segreta di 32 KB. Ciò consente di evitare il costo di elaborazione di un numero eccessivo di reimpostazioni della chiave che si verificherebbe per valori di reimpostazione della chiave segreta TLS di piccole dimensioni.

Questo è un campo di immissione. Il valore è un numero compreso tra 0 e 999 999 999, con un valore predefinito pari a 0. Utilizzare il valore 0 per indicare che le chiavi segrete non vengono mai rinegoziate.

FipsRequired (MQLONG) per MQSCO

IBM MQ può essere configurato con hardware crittografico in modo che i moduli di codifica utilizzati siano quelli forniti dal prodotto hardware; questi possono essere certificati FIPS ad un determinato livello a seconda del prodotto hardware crittografico in uso. Utilizzare questo campo per specificare che solo gli algoritmi certificati FIPS vengono utilizzati se la crittografia viene fornita nel software fornito da IBM MQ.

Nota: Su AIX, Linux, and Windows, IBM MQ fornisce la conformità FIPS 140-2 tramite il modulo crittografico IBM Crypto for C (ICC) . Il certificato per questo modulo è stato spostato nello stato cronologico. I clienti devono visualizzare il certificatoIBM Crypto for C (ICC) ed essere a conoscenza di eventuali consigli forniti da NIST. Un modulo FIPS 140-3 di sostituzione è attualmente in corso e il relativo stato può essere visualizzato ricercandolo in NIST CMVP modules in process list.

IBM MQ Operator 3.2.0 e l'immagine del contenitore del gestore code 9.4.0.0 sono basati su UBI 9. La conformità FIPS 140-3 è attualmente in sospeso e il suo stato può essere visualizzato ricercando "Red Hat Enterprise Linux 9 - OpenSSL FIPS Provider" in NIST CMVP modules in process list.

Quando IBM MQ è installato, viene installata anche un'implementazione della crittografia TLS che fornisce alcuni moduli certificati FIPS.

I valori possono essere:

MQSSL_FIPS_NO
Questo è il valore predefinito. Quando impostato su questo valore:

- È possibile utilizzare qualsiasi CipherSpec supportato su una particolare piattaforma.
- Se si esegue senza l'utilizzo dell'hardware crittografico, i CipherSpecs vengono eseguiti utilizzando la crittografia certificata FIPS 140-2 sulle piattaforme IBM MQ .

  Per un elenco di CipherSpecscertificati FIPS, consultare la tabella descritta in Abilitazione di CipherSpecs.

SÌ MQSSL_FIPS
Quando è impostato su questo valore, a meno che non si stia utilizzando l'hardware crittografico per eseguire la crittografia, è possibile essere certi che

- Solo gli algoritmi di crittografia certificati FIPS possono essere utilizzati in CipherSpec che si applica a questa connessione client.
- Le connessioni del canale TLS in entrata e in uscita hanno esito positivo solo se vengono utilizzate determinate specifiche di cifratura.

  Per ulteriori informazioni, consultare Abilitazione di CipherSpecs .

Nota: Laddove possibile, se sono configurati CipherSpecs solo FIPS, il client MQI rifiuta le connessioni che specificano una CipherSpec non FIPS con `MQRC_SSL_INITIALIZATION_ERROR`. IBM MQ non garantisce di rifiutare tutte queste connessioni ed è responsabilità dell'utente determinare se la propria configurazione IBM MQ è conforme a FIPS.

EncryptionPolicySuiteB (MQLONG) per MQSCO

Questo campo specifica se viene utilizzata la crittografia conforme a Suite B e quale livello di intensità viene utilizzato. Il valore può essere uno o più dei seguenti:

- MQ_SUITE_B_NONE

  La crittografia conforme alla suite B non viene utilizzata.
- MQ_SUITE_B_128_BIT

  Viene utilizzata la sicurezza della suite B a 128 bit.
- MQ_SUITE_B_192_BIT

  Viene utilizzata la sicurezza della suite B a 192 bit.

Nota: L'utilizzo di MQ_SUITE_B_NONE con qualsiasi altro valore in questo campo non è valido.

Politica CertificateVal(MQLONG) per MQSCO

Questo campo specifica quale tipo di politica di convalida del certificato viene utilizzato.

Il campo può essere impostato su uno dei seguenti valori:

MQ_CERT_VAL_POLICY_ANY
Applicare ciascuna delle politiche di convalida del certificato supportate dalla libreria dei socket sicuri. Accettare la catena di certificati se una delle politiche considera valida la catena di certificati.

MQ_CERT_VAL_POLICY_RFC5280
Applicare solo la politica di convalida del certificato conforme a RFC5280 . Questa impostazione fornisce una convalida più rigorosa rispetto all'impostazione ANY, ma rifiuta alcuni certificati digitali meno recenti.

V 9.4.0 V 9.4.0 **MQ_CERT_VAL_POLICY_NONE**
Non applicare alcuna politica di convalida del certificato. Questa impostazione è solo per applicazioni client e accetta il certificato del server TLS senza convalidare la catena di attendibilità.

Il valore iniziale di questo campo è MQ_CERT_VAL_POLICY_ANY

CertificateLabel (MQCHAR64) per MQSCO
Questo campo fornisce i dettagli dell'etichetta del certificato utilizzata.

IBM MQ inizializza il valore predefinito per il campo *CertificateLabel* come spazi vuoti.

Viene interpretato al runtime come il valore predefinito ed è compatibile con le versioni precedenti.

Ad esempio, se si specifica una versione di MQSCO inferiore a 5.0o si utilizza il valore predefinito di spazi vuoti per il campo *CertificateLabel* , si utilizza il valore predefinito preesistente di `ibmwebspheremq`*`user_id`*.

Multi ***KeyRepoPasswordPtr (MQPTR) per MQSCO***
Questo è l'indirizzo in byte della passphrase del repository chiavi TLS.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e, in caso contrario, una stringa di byte completamente null. Questo campo viene ignorato se *`Version`* è minore di MQSCO_VERSION_6.

Questo campo è rilevante solo per IBM MQ MQI clients in esecuzione su IBM i, AIX, Linux, and Windows sistemi.

La passphrase del repository delle chiavi può essere specificata come una stringa di testo semplice o una passphrase che è stata codificata utilizzando il programma di utilità **`runmqicred`** .

Se si fornisce una passphrase codificata, specificare la chiave iniziale utilizzata per codificare la passphrase nella struttura MQCSP fornita dalla stessa applicazione client.

La passphrase del repository di chiavi specificata utilizzando questo campo sovrascrive qualsiasi passphrase del repository di chiavi specificata utilizzando la variabile di ambiente *MQKEYRPWD* o la proprietà *SSLKeyRepositoryPassword* nella sezione SSL del file di configurazione del client.

È possibile utilizzare *`KeyRepoPasswordOffset`* o *`KeyRepoPasswordPtr`* per specificare la passphrase del repository delle chiavi, ma non entrambe.

Attività correlate
Fornitura di una chiave iniziale per il client IBM MQ MQI su AIX, Linux e Windows
Protezione delle password nei file di configurazione del componente IBM MQ
Riferimenti correlati
runmqicred (proteggi password client IBM MQ)
“InitialKeyPtr (MQPTR) per MQCSP” a pagina 347
L'indirizzo per la chiave iniziale per il sistema di protezione password.

Multi ***KeyRepoPasswordOffset (MQLONG) per MQSCO***
Questo è l'offset in byte della passphrase del repository chiavi TLS dall'inizio della struttura MQSCO. L'offset può essere positivo o negativo.

È possibile utilizzare *KeyRepoPasswordOffset* o *KeyRepoPasswordPtr* per specificare la passphrase del repository delle chiavi, ma non entrambe. Per ulteriori informazioni, consultare la descrizione del campo *KeyRepoPasswordPtr* .

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è minore di MQSCO_VERSION_6.

Multi *KeyRepoPasswordLength (MQLONG) per MQSCO*

Questa è la lunghezza della passphrase del repository chiavi TLS.

Su IBM i, la lunghezza massima della passphrase del repository delle chiavi è 128 caratteri. Se la passphrase del repository delle chiavi è maggiore della lunghezza massima consentita, la connessione non riesce con MQRC_KEY_REPOSITORY_ERROR.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *Version* è minore di MQSCO_VERSION_6.

MQSD - Descrittore sottoscrizione

La struttura MQSD viene utilizzata per specificare i dettagli relativi alla sottoscrizione che si sta effettuando. La struttura è un parametro di input / output nella chiamata MQSUB.Per ulteriori informazioni, consultare Note sull'utilizzo di MQSUB.

Disponibilità

La struttura MQSD è disponibile sulle piattaforme seguenti:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Windows Windows
- z/OS z/OS

e per IBM MQ MQI clients collegati a questi sistemi.

Versione

La versione corrente di MQSD è MQSD_VERSION_1.

Serie di caratteri e codifica

I dati in MQSD devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codificazione del gestore code locale fornita da MQENC_NATIVE. Tuttavia, se l'applicazione è in esecuzione come client MQ MQI, la struttura deve essere nella serie di caratteri e nella codifica del client.

Sottoscrizioni gestite

Se un'applicazione non ha la necessità specifica di utilizzare una particolare coda come destinazione per le pubblicazioni che corrispondono alla relativa sottoscrizione, può utilizzare la funzione di sottoscrizione gestita. Se un'applicazione sceglie di utilizzare una sottoscrizione gestita, il gestore code informa il sottoscrittore della destinazione in cui vengono inviati i messaggi pubblicati, fornendo un handle dell'oggetto come output dalla chiamata MQSUB. Per ulteriori informazioni, consultare Hobj (MQHOBJ) - input/output.

Quando la sottoscrizione viene rimossa, il gestore code si impegna anche a ripulire i messaggi che non sono stati richiamati dalla destinazione gestita, nelle seguenti situazioni:

- Quando la sottoscrizione viene eliminata - utilizzando MQCLOSE con MQCO_REMOVE_SUB - e l'Hobj gestito viene chiuso.
- Per implicito si intende quando la connessione viene persa per un'applicazione che utilizza una sottoscrizione non durevole (MQSO_NON_DURABLE)
- Per scadenza quando una sottoscrizione viene rimossa perché è scaduta e l'Hobj gestito è chiuso.

È necessario utilizzare le sottoscrizioni gestite con sottoscrizioni non durevoli, in modo che questa ripulitura possa verificarsi e in modo che i messaggi per le sottoscrizioni non durevoli chiuse non occupano spazio nel gestore code. Le sottoscrizioni durevoli possono anche utilizzare destinazioni gestite.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQSD	'SD¬¬'
Versione (numero versione struttura)	MQSD_VERSION_1	1
Opzioni (opzioni)	MQSO_NON_DURABLE	0
ObjectName (nome oggetto)	Nessuna	Stringa null o spazi vuoti
AlternateUserId (ID utente alternativo)	Nessuna	Stringa null o spazi vuoti
AlternateSecurityId (ID sicurezza alternativo)	MQSID_NONE	Valori null
SubExpiry (scadenza sottoscrizione)	MQEI_UNLIMITED	-1
ObjectString (stringa oggetto)	Nessuna	Nomi e valori come definiti per MQCHARV
SubName (nome sottoscrizione)	Nessuna	Nomi e valori come definiti per MQCHARV
SubUserData (dati utente sottoscrizione)	Nessuna	Nomi e valori come definiti per MQCHARV
SubCorrelId (ID correlazione sottoscrizione)	MQCI_NONE	Valori null
PubPriority (priorità di pubblicazione)	MQPRI_PRIORITY_AS_Q_DEF	-3
PubAccountingToken (token di account di pubblicazione)	MQACT_NONE	Valori null
PubAppIdentityData (dati di identità dell'applicazione di pubblicazione)	Nessuna	Stringa null o spazi vuoti
SelectionString (stringa che fornisce criteri di selezione)	Nessuna	Nomi e valori come definiti per MQCHARV
SubLevel (livello sottoscrizione)	Nessuna	1
ResObjectString (nome oggetto lungo)	Nessuna	Nomi e valori come definiti per MQCHARV

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.
3. Nel linguaggio di programmazione C, la variabile macroMQSD_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura:

```
MQSD MySD = {MQSD_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQSD

```
typedef struct tagMQSD MQSD;
struct tagMQSD {
  MQCHAR4   StrucId;              /* Structure identifier */
  MQLONG    Version;              /* Structure version number */
  MQLONG    Options;              /* Options associated with subscribing */
  MQCHAR48  ObjectName;           /* Object name */
  MQCHAR12  AlternateUserId;      /* Alternate user identifier */
  MQBYTE40  AlternateSecurityId;  /* Alternate security identifier */
  MQLONG    SubExpiry;            /* Expiry of Subscription */
  MQCHARV   ObjectString;         /* Object Long name */
  MQCHARV   SubName;              /* Subscription name */
  MQCHARV   SubUserData;          /* Subscription User data */
  MQBYTE24  SubCorrelId;          /* Correlation Id related to this subscription */
  MQLONG    PubPriority;          /* Priority set in publications */
  MQBYTE32  PubAccountingToken;   /* Accounting Token set in publications */
  MQCHAR32  PubApplIdentityData;  /* Appl Identity Data set in publications */
  MQCHARV   SelectionString;      /* Message selector structure */
  MQLONG    SubLevel;             /* Subscription level */
  MQCHARV   ResObjectString;      /* Resolved Long object name*/
  /* Ver:1 */
};
```

Dichiarazione COBOL per MQSD

```
** Address of variable length string
   20  MQSD-OBJECTSTRING-VSPTR         POINTER.
** Offset of variable length string
   20  MQSD-OBJECTSTRING-VSOFFSET      PIC S9(9) BINARY.
** size of buffer
   20  MQSD-OBJECTSTRING-VSBUFSIZE     PIC S9(9) BINARY.
** Length of variable length string
   20  MQSD-OBJECTSTRING-VSLENGTH      PIC S9(9) BINARY.
** CCSID of variable length string
   20  MQSD-OBJECTSTRING-VSCCSID       PIC S9(9) BINARY.
** Subscription name
 15  MQSD-SUBNAME.
** Address of variable length string
   20  MQSD-SUBNAME-VSPTR              POINTER.
** Offset of variable length string
   20  MQSD-SUBNAME-VSOFFSET           PIC S9(9) BINARY.
** size of buffer
   20  MQSD-SUBNAME-VSBUFSIZE          PIC S9(9) BINARY.
** Length of variable length string
   20  MQSD-SUBNAME-VSLENGTH           PIC S9(9) BINARY.
** CCSID of variable length string
   20  MQSD-SUBNAME-VSCCSID            PIC S9(9) BINARY.
** Subscription User data
 15  MQSD-SUBUSERDATA.
** Address of variable length string
```

```
   20  MQSD-SUBUSERDATA-VSPTR          POINTER.
** Offset of variable length string
   20  MQSD-SUBUSERDATA-VSOFFSET       PIC S9(9) BINARY.
** size of buffer
   20  MQSD-SUBUSERDATA-VSBUFSIZE      PIC S9(9) BINARY.
** Length of variable length string
   20  MQSD-SUBUSERDATA-VSLENGTH       PIC S9(9) BINARY.
** CCSID of variable length string
   20  MQSD-SUBUSERDATA-VSCCSID        PIC S9(9) BINARY.
** Correlation Id related to this subscription
 15  MQSD-SUBCORRELID                  PIC X(24).
** Priority set in publications
 15  MQSD-PUBPRIORITY                  PIC S9(9) BINARY.
** Accounting Token set in publications
 15  MQSD-PUBACCOUNTINGTOKEN           PIC X(32).
** Appl Identity Data set in publications
 15  MQSD-PUBAPPLIDENTITYDATA          PIC X(32).
** Message Selector
 15 MQSD-SELECTIONSTRING.
** Address of variable length string
   20  MQSD-SELECTIONSTRING-VSPTR     POINTER.
** Offset of variable length string
   20  MQSD-SELECTIONSTRING-VSOFFSET       PIC S9(9) BINARY.
** size of buffer
   20  MQSD-SELECTIONSTRING-VSBUFSIZE      PIC S9(9) BINARY.
** Length of variable length string
   20  MQSD-SELECTIONSTRING-VSLENGTH       PIC S9(9) BINARY.
** CCSID of variable length string
   20  MQSD-SELECTIONSTRING-VSCCSID        PIC S9(9) BINARY.
** Selection criteria
   20  MQSD-SELECTIONSTRING-SUBLEVEL       PIC S9(9) BINARY.
** Long object name
   20  MQSD-SELECTIONSTRING-RESOBJSTRING   PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQSD

```
dcl
1 MQSD based,
3 StrucId       char(4),    /* Structure identifier */
3 Version       fixed bin(31), /* Structure version number */
3 Options       fixed bin(31), /* Options associated with subscribing */
3 ObjectName     char(48),   /* Object name */
3 AlternateUserId   char(12),   /* Alternate user identifier */
3 AlternateSecurityId char(40),   /* Alternate security identifier */
3 SubExpiry      fixed bin(31), /* Expiry of Subscription */
3 ObjectString,           /* Object Long name */
5 VSPtr       pointer,    /* Address of variable length string */
5 VSOffset      fixed bin(31), /* Offset of variable length string */
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID      fixed bin(31); /* CCSID of variable length string */
3 SubName,              /* Subscription name */
5 VSPtr       pointer,    /* Address of variable length string */
5 VSOffset      fixed bin(31), /* Offset of variable length string */
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID      fixed bin(31); /* CCSID of variable length string */
3 SubUserData,            /* Subscription User data */
5 VSPtr       pointer,    /* Address of variable length string */
5 VSOffset      fixed bin(31), /* Offset of variable length string */
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID      fixed bin(31), /* CCSID of variable length string */
3 SubCorrelId     char(24),   /* Correlation Id related to this subscription */
3 PubPriority     fixed bin(31), /* Priority set in publications */
3 PubAccountingToken char(32),   /* Accounting Token set in publications */
3 PubApplIdentityData char(32),   /* Appl Identity Data set in publications */
3 SelectionString,          /* Message Selection */
5 VSPtr       pointer,    /* Address of variable length string */
5 VSOffset      fixed bin(31), /* Offset of variable length string */
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID      fixed bin(31), /* CCSID of variable length string */
3 SubLevel      fixed bin(31), /* Subscription level */
3 ResObjectString,          /* Resolved Long object name */
5 VSPtr       pointer,    /* Address of variable length string */
5 VSOffset      fixed bin(31), /* Offset of variable length string */
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID      fixed bin(31); /* CCSID of variable length string */
```

Dichiarazione High Level Assembler per MQSD

```
MQSD              DSECT
MQSD_STRUCID          DS  CL4  Structure identifier
MQSD_VERSION          DS  F   Structure version number
MQSD-OPTIONS          DS  F   Options associated with subscribing
MQSD_OBJECTNAME        DS  CL48 Object name
MQSD_ALTERNATEUSERID      DS  CL12 Alternate user identifier
MQSD_ALTERNATESECURITYID    DS  CL40 Alternate security identifier
MQSD_SUBEXPIRY         DS  F   Expiry of Subscription
MQSD_OBJECTSTRING       DS  0F  Object Long name
MQSD_OBJECTSTRING_VSPTR    DS  F   Address of variable length string
MQSD_OBJECTSTRING_VSOFFSET   DS  F   Offset of variable length string
MQSD_OBJECTSTRING_VSBUFSIZE  DS  F   size of buffer
MQSD_OBJECTSTRING_VSLENGTH   DS  F   Length of variable length string
MQSD_OBJECTSTRING_VSCCSID   DS  F   CCSID of variable length string
MQSD_OBJECTSTRING_LENGTH    EQU  *-MQSD_OBJECTSTRING
ORG  MQSD_OBJECTSTRING
MQSD_OBJECTSTRING_AREA     DS  CL(MQSD_OBJECTSTRING_LENGTH)
*
MQSD_SUBNAME          DS  0F  Subscription name
MQSD_SUBNAME_VSPTR       DS  F   Address of variable length string
MQSD_SUBNAME_VSOFFSET     DS  F   Offset of variable length string
MQSD_SUBNAME_VSBUFSIZE     DS  F   size of buffer
MQSD_SUBNAME_VSLENGTH     DS  F   Length of variable length string
MQSD_SUBNAME_VSCCSID      DS  F   CCSID of variable length string
MQSD_SUBNAME_LENGTH      EQU  *-MQSD_SUBNAME
ORG  MQSD_SUBNAME
MQSD_SUBNAME_AREA       DS  CL(MQSD_SUBNAME_LENGTH)
*
MQSD_SUBUSERDATA        DS  0F  Subscription User data
MQSD_SUBUSERDATA_VSPTR     DS  F   Address of variable length string
MQSD_SUBUSERDATA_VSOFFSET   DS  F   Offset of variable length string
MQSD_SUBUSERDATA_VSBUFSIZE   DS  F   size of buffer
MQSD_SUBUSERDATA_VSLENGTH   DS  F   Length of variable length string
MQSD_SUBUSERDATA_VSCCSID    DS  F   CCSID of variable length string
MQSD_SUBUSERDATA_LENGTH    EQU  *-MQSD_SUBUSERDATA
ORG  MQSD_SUBUSERDATA
MQSD_SUBUSERDATA_AREA     DS  CL(MQSD_SUBUSERDATA_LENGTH)
*
MQSD_SUBCORRELID        DS  CL24 Correlation Id related to this subscription
MQSD_PUBPRIORITY        DS  F   Priority set in publications
MQSD_PUBACCOUNTINGTOKEN    DS  CL32 Accounting Token set in publications
MQSD_PUBAPPLIDENTITYDATA    DS  CL32 Appl Identity Data set in publications
*
MQSD_SELECTIONSTRING      DS  F   Message Selector
MQSD_SELECTIONSTRING_VSPTR   DS  F   Address of variable length string
MQSD_SELECTIONSTRING_VSOFFSET DS  F   Offset of variable length string
MQSD_SELECTIONSTRING_VSBUFSIZE DS  F   size of buffer
MQSD_SELECTIONSTRING_VSLENGTH DS  F   Length of variable length string
MQSD_SELECTIONSTRING_VSCCSID  DS  F   CCSID of variable length string
MQSD_SELECTIONSTRING_LENGTH  EQU *- MQSD_SELECTIONSTRING
ORG MQSD_SELECTIONSTRING
MQSD_SELECTIONSTRING_AREA   DS  CL(MQSD_SELECTIONSTRING_LENGTH)
*
MQSD-SUBLEVEL         DS  F   Subscription level
*
MQSD_RESOBJECTSTRING      DS  F   Resolved Long object name
MQSD_RESOBJECTSTRING_VSPTR   DS  F   Address of variable length string
MQSD_RESOBJECTSTRING_VSOFFSET DS  F   Offset of variable length string
MQSD_RESOBJECTSTRING_VSBUFSIZE DS  F   size of buffer
MQSD_RESOBJECTSTRING_VSLENGTH DS  F   Length of variable length string
MQSD_RESOBJECTSTRING_VSCCSID  DS  F   CCSID of variable length string
MQSD_RESOBJECTSTRING_LENGTH  EQU *- MQSD_RESOBJECTSTRING
ORG MQSD_RESOBJECTSTRING
MQSD_RESOBJECTSTRING_AREA   DS  CL(MQSD_RESOBJECTSTRING_LENGTH)
*
MQSD_LENGTH          EQU  *-MQSD
ORG  MQSD
MQSD_AREA           DS  CL(MQSD_LENGTH)
```

StrucId (MQCHAR4) per MQSD

Questo è l'identificatore della struttura del descrittore di sottoscrizione. È sempre un campo di immissione. Il suo valore è MQSD_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQSD
Identificativo per la struttura del descrittore di sottoscrizione.

Per il linguaggio di programmazione C, viene definita anche la costante MQSD_STRUC_ID_ARRAY. Questo ha lo stesso valore di MQSD_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQSD

Questo è il numero di versione della struttura; il valore deve essere:

MQSD_VERSION_1
Version-1 La struttura del descrittore di sottoscrizione.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQSD_CURRENT_
La versione corrente della struttura del descrittore di sottoscrizione.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQSD_VERSION_1.

Opzioni (MQLONG) per MQSD

Fornisce opzioni per controllare l'azione della chiamata MQSUB.

È possibile specificare almeno una delle seguenti opzioni:

- ALTER MQSO
- RESUME MQSO
- CREA_MQSO

Per specificare più di un'opzione, aggiungere i valori insieme (non aggiungere la stessa costante più di una volta) o combinare i valori utilizzando l'operazione OR bit per bit (se il linguaggio di programmazione supporta le operazioni bit).

Le combinazioni che non sono valide sono riportate in questo argomento; tutte le altre combinazioni sono valide.

Opzioni di accesso o creazione: le opzioni di accesso e creazione controllano se viene creata una sottoscrizione o se viene restituita o modificata una sottoscrizione esistente. È necessario specificare almeno una di queste opzioni.

Tabella 526. Combinazioni valide di opzioni di accesso e creazione

Combinazione di opzioni	**Note**
CREA_MQSO	Crea una sottoscrizione se non esiste. Questa combinazione non riesce se la sottoscrizione esiste.
RESUME MQSO	Riprende una sottoscrizione esistente. Questa combinazione non riesce se non esiste alcuna sottoscrizione.
MQSO_CREATE + MQSO_RESUME	Crea una sottoscrizione se non esiste e ne riprende una corrispondente, se esiste. Questa combinazione è utile quando viene utilizzata in una applicazione eseguita più volte.
MQSO_ALTER (vedere nota)	Riprende una sottoscrizione esistente, modificando i campi in modo che corrispondano a quelli specificati in MQSD. Questa combinazione non riesce se non esiste alcuna sottoscrizione.

Tabella 526. Combinazioni valide di opzioni di accesso e creazione (Continua)	
Combinazione di opzioni	**Note**
MQSO_CREATE + MQSO_ALTER (vedere nota)	Crea una sottoscrizione se non esiste e ne riprende una corrispondente, se esiste, modificando gli eventuali campi in modo che corrispondano a quelli specificati in MQSD. Questa combinazione è utile quando viene utilizzata in una applicazione che desidera assicurarsi che la propria sottoscrizione si trovi in un determinato stato prima di continuare.

Nota:

Le opzioni che specificano MQSO_ALTER possono anche specificare MQSO_RESUME, ma questa combinazione non ha alcun effetto aggiuntivo sulla sola specifica di MQSO_ALTER. MQSO_ALTER implica MQSO_RESUME, perché il richiamo di MQSUB per modificare una sottoscrizione implica che anche la sottoscrizione verrà ripresa. Ma non è vero il contrario: riprendere un abbonamento non significa modificarlo.

CREA_MQSO

Creare una nuova sottoscrizione per l'argomento specificato. Se esiste una sottoscrizione che utilizza lo stesso *SubName* , la chiamata ha esito negativo con MQRC_SUB_ALREADY_EXISTS. Questo errore può essere evitato combinando l'opzione MQSO_CREATE con MQSO_RESUME. *SubName* non è sempre necessario. Per ulteriori dettagli, consultare la descrizione di tale campo.

La combinazione di MQSO_CREATE con MQSO_RESUME restituisce un handle a una sottoscrizione pre - esistente per il *SubName* specificato, se ne viene trovata una; se non esiste una sottoscrizione, ne viene creata una nuova utilizzando tutti i campi forniti in MQSD.

MQSO_CREATE può anche essere combinato con MQSO_ALTER per ottenere un effetto simile.

RESUME MQSO

Restituisce un handle a una sottoscrizione preesistente che corrisponde a quella specificata da *SubName*. Non vengono apportate modifiche agli attributi delle sottoscrizioni corrispondenti e vengono restituiti nell'output nella struttura MQSD. Vengono utilizzati solo i seguenti campi MQSD: StrucId, Versione, Opzioni, AlternateUserId e AlternateSecurityId e SubName.

La chiamata ha esito negativo con codice motivo MQRC_NO_SUBSCRIPTION se non esiste una sottoscrizione corrispondente al nome completo della sottoscrizione. Questo errore può essere evitato combinando l'opzione MQSO_CREATE con MQSO_RESUME.

L'ID utente della sottoscrizione è l'ID utente che ha creato la sottoscrizione oppure, se è stato successivamente modificato da un ID utente diverso, è l'ID utente della modifica più recente. Se viene utilizzato un ID AlternateUsere l'utilizzo di ID utente alternativi è consentito per tale utente, l'ID utente alternativo viene registrato come l'ID utente che ha creato la sottoscrizione invece dell'ID utente con cui è stata effettuata la sottoscrizione.

Se esiste una sottoscrizione corrispondente creata senza l'opzione MQSO_ANY_USERID e l'ID utente della sottoscrizione è diverso da quello dell'applicazione che richiede un handle per la sottoscrizione, la chiamata ha esito negativo con codice motivo MQRC_IDENTITY_MISMATCH.

Se una sottoscrizione corrispondente esiste ed è attualmente in uso, la chiamata non riesce con MQRC_SUBSCRIPTION_IN_USE.

Se la sottoscrizione indicata in SubName non è una sottoscrizione valida da riprendere o modificare da un'applicazione, la chiamata ha esito negativo con MQRC_INVALID_SUBSCRIPTION.

MQSO_RESUME è implicito in MQSO_ALTER, quindi non è necessario combinarlo con tale opzione. Tuttavia, la combinazione delle due opzioni non causa un errore.

ALTER MQSO

Restituire un handle a una sottoscrizione preesistente con il nome sottoscrizione completo corrispondente a quello specificato dal nome in *SubName*. Tutti gli attributi della sottoscrizione diversi da quelli specificati in MQSD vengono modificati nella sottoscrizione a meno che la modifica non sia disconsentita per tale attributo. I dettagli sono riportati nella descrizione di ciascun attributo e riepilogati nella seguente tabella. Se si tenta di modificare un attributo che non può essere modificato o di modificare una sottoscrizione che ha impostato l'opzione MQSO_IMMUTABLE, la chiamata ha esito negativo con il codice di errore riportato nella seguente tabella.

La chiamata non riesce con codice motivo MQRC_NO_SUBSCRIPTION se non esiste una sottoscrizione corrispondente al nome completo della sottoscrizione. È possibile evitare questo errore combinando l'opzione MQSO_CREATE con MQSO_ALTER.

La combinazione di MQSO_CREATE con MQSO_ALTER restituisce un handle a una sottoscrizione preesistente per il *SubName* specificato, se ne viene trovata una; se non esiste alcuna sottoscrizione, ne viene creata una nuova utilizzando tutti i campi forniti in MQSD.

L'ID utente della sottoscrizione è l'ID utente che ha creato la sottoscrizione oppure, se successivamente viene modificato da un ID utente differente, è l'ID utente della modifica più recente e corretta. Se viene utilizzato un ID AlternateUsere l'utilizzo di ID utente alternativi è consentito per tale utente, l'ID utente alternativo viene registrato come l'ID utente che ha creato la sottoscrizione invece dell'ID utente con cui è stata effettuata la sottoscrizione.

Se esiste una sottoscrizione corrispondente creata senza l'opzione MQSO_ANY_USERID e l'ID utente della sottoscrizione è diverso da quello dell'applicazione che richiede un handle per la sottoscrizione, la chiamata non riesce con codice motivo MQRC_IDENTITY_MISMATCH.

Se una sottoscrizione corrispondente esiste ed è attualmente in uso, la chiamata non riesce con MQRC_SUBSCRIPTION_IN_USE.

Se la sottoscrizione indicata in SubName non è una sottoscrizione valida da riprendere o modificare da un'applicazione, la chiamata ha esito negativo con MQRC_INVALID_SUBSCRIPTION.

La tabella riportata di seguito mostra la capacità di MQSO_ALTER di modificare i valori degli attributi in MQSD e MQSUB.

Tabella 527. Attributi in MQSD e MQSUB che possono essere modificati

Descrittore tipo di dati o chiamata funzione	Nome campo	È possibile modificare questo attributo utilizzando MQSO_ALTER	Codice di errore
MQSD	Opzioni di durata	No	MQRC_DURABILITY_NOT_ALTERABLE
MQSD	Opzioni di destinazione	Sì	Nessuna
MQSD	Opzioni di registrazione	Sì (vedere la nota “1” a pagina 590)	MQRC_GROUPING_NOT_ALTERABLE se si tenta di modificare MQSO_GROUP_SUB
MQSD	Opzioni di pubblicazione	Sì (vedere la nota “2” a pagina 590)	Nessuna
MQSD	Opzioni carattere jolly	No	MQRC_TOPIC_NOT_ALTERABLE
MQSD	Altre opzioni	No (vedere nota “3” a pagina 590)	Nessuna
MQSD	ObjectName	No	MQRC_TOPIC_NOT_ALTERABLE
MQSD	AlternateUserid	No (vedere nota “4” a pagina 590)	Nessuna
MQSD	AlternateSecurityId	No (vedere nota “4” a pagina 590)	Nessuna
MQSD	SubExpiry	Sì	Nessuna
MQSD	ObjectString	No	MQRC_TOPIC_NOT_ALTERABLE
MQSD	SubName	No (vedere nota “5” a pagina 590)	Nessuna
MQSD	SubUserData	Sì	Nessuna

Tabella 527. Attributi in MQSD e MQSUB che possono essere modificati (Continua)			
Descrittore tipo di dati o chiamata funzione	**Nome campo**	**È possibile modificare questo attributo utilizzando MQSO_ALTER**	**Codice di errore**
MQSD	SubCorrelId	Sì (vedere la nota “6” a pagina 590)	MQRC_GROUPING_NOT_ALTERABLE quando si trova in una sottoscrizione raggruppata
MQSD	PubPriority	Sì	Nessuna
MQSD	Token PubAccounting	Sì	Nessuna
MQSD	PubApplIdentityData	Sì	Nessuna
MQSD	SubLevel	No	MQRC_SUBLEVEL_NOT_ALTERABLE
MQSUB	HOBJ	Sì (vedere la nota “6” a pagina 590)	MQRC_GROUPING_NOT_ALTERABLE quando si trova in una sottoscrizione raggruppata

Note:

1. MQSO_GROUP_SUB non può essere modificato.
2. MQSO_NEW_PUBLICATIONS_ONLY non può essere modificato perché non fa parte della sottoscrizione
3. Queste opzioni non fanno parte della sottoscrizione
4. Questo attributo non fa parte della sottoscrizione
5. Questo attributo è l'identità della sottoscrizione che si sta modificando
6. Modificabile tranne quando fa parte di un sottoinsieme raggruppato (MQSO_GROUP_SUB)

Opzioni di durata: le seguenti opzioni controllano la durata della sottoscrizione. È possibile specificare solo una di queste opzioni. Se si sta modificando una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, non è possibile modificare la durata della sottoscrizione. Al ritorno da una chiamata MQSUB che utilizza MQSO_RESUME, viene impostata l'opzione di durata appropriata.

MQSO_DURATA
Richiedere che la sottoscrizione a questo argomento resti fino a che non viene esplicitamente rimossa utilizzando MQCLOSE con l'opzione MQCO_REMOVE_SUB. Se questa sottoscrizione non viene esplicitamente rimossa, rimarrà anche dopo la chiusura della connessione delle applicazioni al gestore code.

Se viene richiesta una sottoscrizione durevole a un argomento definito come non consentendo sottoscrizioni durevoli, la chiamata non riesce con MQRC_DURABILITY_NOT_ALLOWED.

MQSO_NON_DURABLE
Richiedere che la sottoscrizione a questo argomento venga rimossa quando la connessione delle applicazioni al gestore code viene chiusa, se non è già stata esplicitamente rimossa. MQSO_NON_DURABLE è l'opposto dell'opzione MQSO_DURABLE ed è definito per la documentazione del programma. È il valore predefinito se non viene specificato nessuno dei due.

Opzioni di destinazione: la seguente opzione controlla la destinazione a cui vengono inviate le pubblicazioni per un argomento a cui è stata effettuata la sottoscrizione. Se si modifica una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, è possibile modificare la destinazione utilizzata per le pubblicazioni per la sottoscrizione. Al ritorno da una chiamata MQSUB utilizzando MQSO_RESUME, questa opzione è impostata, se appropriato.

MQSO_MANAGED

Richiedere che la destinazione a cui vengono inviate le pubblicazioni sia gestita dal gestore code.

L'handle dell'oggetto restituito in *Hobj* rappresenta una coda gestita del gestore code ed è da utilizzare con le successive chiamate MQGET, MQCB, MQINQ o MQCLOSE.

Non è possibile fornire un handle dell'oggetto restituito da una chiamata MQSUB precedente nel parametro **Hobj** quando MQSO_MANAGED non viene specificato.

MQSO_NO_MULTICAST

Richiedere che la destinazione a cui vengono inviate le pubblicazioni non sia un indirizzo di gruppo multicast. Questa opzione è valida solo se combinata con l'opzione MQSO_MANAGED. Quando viene fornito un handle per una coda nel parametro **Hobj** , non è possibile utilizzare il multicast per questa sottoscrizione e l'opzione non è valida.

Se l'argomento è definito per consentire solo le sottoscrizioni multicast, utilizzando l'impostazione `MCAST(ONLY)` , la chiamata ha esito negativo con codice motivo MQRC_MULTICAST_REQUIRED.

Opzione ambito: la seguente opzione controlla l'ambito della sottoscrizione effettuata. Se si modifica una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, questa opzione dell'ambito della sottoscrizione non può essere modificata. Quando si ritorna da una chiamata MQSUB utilizzando MQSO-RESUME, viene impostata l'opzione di ambito appropriata.

MQSO_SCOPE_QMGR

Questa sottoscrizione viene effettuata solo sul gestore code locale. Nessuna sottoscrizione proxy viene distribuita ad altri gestori code nella rete. Solo le pubblicazioni pubblicate in questo gestore code vengono inviate a questo sottoscrittore. Ciò sovrascrive qualsiasi comportamento impostato utilizzando l'attributo dell'argomento SUBSCOPE.

Nota: Se non è impostato, l'ambito della sottoscrizione è determinato dall'attributo dell'argomento SUBSCOPE.

Opzioni di registrazione: le seguenti opzioni controllano i dettagli della registrazione effettuata al gestore code per questa sottoscrizione. Se si modifica una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, queste opzioni di registrazione possono essere modificate. Al ritorno da una chiamata MQSUB che utilizza MQSO_RESUME, vengono impostate le opzioni di registrazione appropriate.

SUB_GROUP_MQSO

Questa sottoscrizione deve essere raggruppata con altre sottoscrizioni dello stesso SubLevel utilizzando la stessa coda e specificando lo stesso ID di correlazione in modo che tutte le pubblicazioni per argomenti che potrebbero causare la fornitura di più di un messaggio di pubblicazione al gruppo di sottoscrizioni, a causa della sovrapposizione di una serie di stringhe argomento utilizzate, provochino la consegna di un solo messaggio alla coda. Se questa opzione non viene utilizzata, allora ogni sottoscrizione univoca (identificata da SubName) che corrisponde viene fornita con una copia della pubblicazione che potrebbe significare che più di una copia della pubblicazione può essere inserita nella coda condivisa da un numero di sottoscrizioni.

Solo la sottoscrizione più significativa nel gruppo viene fornita con una copia della pubblicazione. La sottoscrizione più significativa si basa sul nome dell'argomento completo fino al punto in cui viene trovato un carattere jolly. Se viene utilizzata una combinazione di schemi di caratteri jolly all'interno del gruppo, è importante solo la posizione del carattere jolly. Si consiglia di non combinare diversi schemi di caratteri jolly in un gruppo di sottoscrizioni che condividono la stessa coda.

Quando si crea una nuova sottoscrizione raggruppata, deve ancora avere un SubNameunivoco, ma se corrisponde al nome dell'argomento completo di una sottoscrizione esistente nel gruppo, la chiamata ha esito negativo con MQRC_DUPLICATE_GROUP_SUB.

Se la sottoscrizione più significativa nel gruppo specifica anche MQSO_NOT_OWN_PUBS e questa è una pubblicazione dalla stessa applicazione, non viene consegnata alcuna pubblicazione alla coda.

Quando si modifica una sottoscrizione effettuata con questa opzione, non è possibile modificare i campi che implicano il raggruppamento, Hobj nella chiamata MQSUB (che rappresenta la coda e il nome gestore code) e l'ID SubCorrel. Se si tenta di modificarli, la chiamata ha esito negativo con MQRC_GROUPING_NOT_ALTERABLE.

Questa opzione deve essere combinata con MQSO_SET_CORREL_ID con un ID SubCorrelnon impostato su MQCI_NONE e non può essere combinato con MQSO_MANAGED.

IDER_ANY_MQSO

Quando viene specificato MQSO_ANY_USERID, l'identit ... del sottoscrittore non Š limitata a un singolo ID utente. Ciò consente a qualsiasi utente di modificare o riprendere la sottoscrizione quando

dispone dell'autorizzazione appropriata. Solo un singolo utente può avere la sottoscrizione in qualsiasi momento. Un tentativo di riprendere l'utilizzo di una sottoscrizione attualmente in uso da parte di un'altra applicazione causa l'esito negativo della chiamata con MQRC_SUBSCRIPTION_IN_USE.

Per aggiungere questa opzione ad una sottoscrizione esistente, la chiamata MQSUB (utilizzando MQSO_ALTER) deve provenire dallo stesso ID utente della sottoscrizione originale.

Se una chiamata MQSUB fa riferimento a una sottoscrizione esistente con MQSO_ANY_USERID impostato e l'ID utente differisce dalla sottoscrizione originale, la chiamata ha esito positivo solo se il nuovo ID utente dispone dell'autorizzazione per sottoscrivere l'argomento. Una volta completato correttamente, le pubblicazioni future per questo sottoscrittore vengono inserite nella coda dei sottoscrittori con il nuovo ID utente impostato nel messaggio di pubblicazione.

Non specificare MQSO_ANY_USERID e MQSO_FIXED_USERID. Se non viene specificato alcun valore, il valore predefinito è MQSO_FIXED_USERID.

IDUSER_FIX_MQSO

Quando viene specificato MQSO_FIXED_USERID, la sottoscrizione può essere modificata o ripresa solo dall'ultimo ID utente per modificare la sottoscrizione. Se la sottoscrizione non è stata modificata, è l'ID utente che ha creato la sottoscrizione.

Se un verbo MQSUB fa riferimento a una sottoscrizione esistente con MQSO_ANY_USERID impostato e modifica la sottoscrizione utilizzando MQSO_ALTER per utilizzare l'opzione MQSO_FIXED_USERID, l'ID utente della sottoscrizione è ora fisso con questo nuovo ID utente. La chiamata ha esito positivo solo se il nuovo ID utente dispone dell'autorizzazione per sottoscrivere l'argomento.

Se un ID utente diverso da quello registrato come proprietario di una sottoscrizione tenta di riprendere o modificare una richiesta MQSO_FIXED_USERID, la chiamata ha esito negativo con MQRC_IDENTITY_MISMATCH. L'ID utente proprietario di una sottoscrizione può essere visualizzato utilizzando il comando DISPLAY SBSTATUS.

Non specificare MQSO_ANY_USERID e MQSO_FIXED_USERID. Se non viene specificato alcun valore, il valore predefinito è MQSO_FIXED_USERID.

Opzioni di pubblicazione: le seguenti opzioni controllano il modo in cui le pubblicazioni vengono inviate a questo sottoscrittore. Se si altera una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, è possibile modificare queste opzioni di pubblicazione.

MQSO_NOT_OWN_PUBS
Comunica al broker che l'applicazione non desidera visualizzare le proprie pubblicazioni. Le pubblicazioni vengono considerate originate dalla stessa applicazione se gli handle di connessione sono gli stessi. Al ritorno da una chiamata MQSUB utilizzando MQSO_RESUME, questa opzione è impostata, se appropriato.

MQSO_NUOVA_PUBBLICAZIONI_SOLO
Nessuna pubblicazione attualmente conservata deve essere inviata, quando viene creata questa sottoscrizione, solo nuove pubblicazioni. Questa opzione si applica solo quando viene specificata MQSO_CREATE. Eventuali modifiche successive ad una sottoscrizione non alterano il flusso di pubblicazioni e, pertanto, tutte le pubblicazioni conservate su un argomento, saranno già state inviate al sottoscrittore come nuove pubblicazioni.

Se questa opzione viene specificata senza MQSO_CREATE, la chiamata ha esito negativo con MQRC_OPTIONS_ERROR. Al ritorno da una chiamata MQSUB che utilizza MQSO_RESUME, questa opzione non viene impostata anche se la sottoscrizione è stata creata utilizzando questa opzione.

Se questa opzione non viene utilizzata, i messaggi precedentemente conservati vengono inviati alla coda di destinazione fornita. Se questa azione ha esito negativo a causa di un errore, MQRC_RETAINED_MSG_Q_ERROR o MQRC_RETAINED_NOT_DELIVERED, la creazione della sottoscrizione non riesce.

MQSO_PUBLICATIONS_ON_REQUEST
L'impostazione di questa opzione indica che il sottoscrittore richiederà le informazioni in modo specifico quando richiesto. Il gestore code non invia messaggi non richiesti al sottoscrittore. La pubblicazione conservata (o probabilmente più pubblicazioni se viene specificato un carattere

jolly nell'argomento) viene inviata al sottoscrittore ogni volta che viene effettuata una chiamata MQSUBRQ utilizzando l'handle Hsub da una precedente chiamata MQSUB. Non viene inviata alcuna pubblicazione come risultato della chiamata MQSUB utilizzando questa opzione. Al ritorno da una chiamata MQSUB utilizzando MQSO_RESUME, questa opzione è impostata, se appropriato.

Questa opzione non è valida in combinazione con un SubLevel maggiore di 1.

Opzioni di lettura anticipata: le opzioni seguenti controllano se i messaggi non persistenti vengono inviati a un'applicazione prima dell'applicazione che li richiede.

MQSO_READ_AHEAD_AS_Q_DEF
Se la chiamata MQSUB utilizza un handle gestito, l'attributo read ahead predefinito della coda modello associata all'argomento sottoscritto determina se i messaggi vengono inviati all'applicazione prima che l'applicazione li richieda.

Questo è il valore predefinito.

MQSO_NO_READ_AHEAD
Se la chiamata MQSUB utilizza un handle gestito, i messaggi non vengono inviati all'applicazione prima che l'applicazione li richieda.

MQSO_READ_AHEAD
Se la chiamata MQSUB utilizza un handle gestito, i messaggi potrebbero essere inviati all'applicazione prima che l'applicazione li richieda.

Nota:

Le seguenti note si applicano alle opzioni di lettura anticipata:

1. È possibile specificare solo una di queste opzioni. Se vengono specificati sia MQOO_READ_AHEAD che MQOO_NO_READ_AHEAD, viene restituito il codice di errore MQRC_OPTIONS_ERROR. Queste opzioni sono applicabili solo se viene specificato MQSO_MANAGED.
2. Non sono applicabili per MQSUB quando viene inoltrata una coda che è stata aperta in precedenza. La lettura anticipata potrebbe non essere abilitata quando richiesto. Le opzioni MQGET utilizzate nella prima chiamata MQGET potrebbero impedire l'abilitazione della lettura anticipata. Inoltre, la lettura anticipata è disabilitata quando il client si connette a un gestore code in cui la lettura anticipata non è supportata. Se l'applicazione non è in esecuzione come client IBM MQ , queste opzioni vengono ignorate.

Opzioni carattere jolly: le seguenti opzioni controllano il modo in cui i caratteri jolly vengono interpretati nella stringa fornita nel campo ObjectString di MQSD. È possibile specificare solo una di queste opzioni. Se si altera una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, queste opzioni del carattere jolly non possono essere modificate. Al ritorno da una chiamata MQSUB che utilizza MQSO_RESUME, viene impostata l'opzione del carattere jolly appropriata.

MQSO_WILDCARD_CHAR
I caratteri jolly operano solo sui caratteri all'interno della stringa argomento.

Il comportamento definito da MQSO_WILDCARD_CHAR viene mostrato nella seguente tabella.

Tabella 528. Come vengono interpretati i caratteri jolly

Carattere speciale	**Comportamento**
Barra (/)	Nessun significato, solo un altro carattere
Asterisco (*)	Carattere jolly, zero o più caratteri
Punto interrogativo (?)	Carattere jolly, 1 carattere
Segno percentuale (%)	Carattere escape per consentire ai caratteri (*), (?) o (%) di essere utilizzati in una stringa e non essere interpretati come un carattere speciale, ad esempio, (% *), (%?) o (%%).

Ad esempio, la pubblicazione sul seguente argomento:

```
/level0/level1/level2/level3/level4
```

corrisponde ai sottoscrittori utilizzando i seguenti argomenti:

```
*
/*
/ level0/level1/level2/level3/*
/ level0/level1/*/level3/level4
/ level0/level1/le?el2/level3/level4
```

Nota: Questo utilizzo di caratteri jolly fornisce esattamente il significato fornito in IBM MQ V6 e WebSphere MB V6 quando si utilizzano i messaggi formattati MQRFH1 per la pubblicazione / sottoscrizione. Si consiglia di non utilizzarlo per le applicazioni appena scritte e di utilizzarlo solo per le applicazioni precedentemente in esecuzione rispetto a tale versione e che non sono state modificate per utilizzare il comportamento del carattere jolly predefinito come descritto in MQSO_WILDCARD_TOPIC.

MQSO_WILDCARD_TOPIC

I caratteri jolly operano solo sugli elementi argomento all'interno della stringa argomento. Questo è il comportamento predefinito se non ne viene scelto alcuno.

Il comportamento richiesto da MQSO_WILDCARD_TOPIC è mostrato nella seguente tabella:

Tabella 529. Come vengono interpretati i caratteri jolly

Carattere speciale	**Comportamento**
(/)	Separatore livello argomento
Cancelletto (#)	Carattere jolly: più livelli di argomento
Segno più (+)	Carattere jolly: livello argomento singolo

Note:

I caratteri jolly (+) e (#) non vengono considerati come caratteri jolly se vengono mischiati con altri caratteri (inclusi se stessi) all'interno di un livello di argomento. Nella seguente stringa, i caratteri (#) e (+) vengono considerati come normali caratteri.

```
level0/level1/#+/level3/level#
```

Ad esempio, la pubblicazione sul seguente argomento:

```
/level0/level1/level2/level3/level4
```

corrisponde ai sottoscrittori utilizzando i seguenti argomenti:

```
#
/#
/ level0/level1/level2/level3/#
/ level0/level1/+/level3/level4
```

Altre opzioni: le opzioni seguenti controllano il modo in cui viene emessa la chiamata API anziché la sottoscrizione. Al ritorno da una chiamata MQSUB utilizzando MQSO_RESUME, queste opzioni non vengono modificate. Per ulteriori dettagli, vedere “ID AlternateUser(MQCHAR12) per MQSD” a pagina 596.

MQSO_ALTERNATE_USER_AUTHORITY

Il campo ID AlternateUsercontiene un identificativo utente da utilizzare per convalidare questa chiamata MQSUB. La chiamata può avere esito positivo solo se questo ID AlternateUserè autorizzato ad aprire l'oggetto con le opzioni di accesso specificate, indipendentemente dal fatto che l'identificativo utente con cui l'applicazione è in esecuzione sia autorizzato a farlo.

ID_CORREL_SET_MQSO

La sottoscrizione deve utilizzare l'identificativo di correlazione fornito nel campo *SubCorrelId* . Se questa opzione non viene specificata, un identificativo di correlazione viene creato automaticamente dal gestore code al momento della sottoscrizione e viene restituito all'applicazione nel campo *SubCorrelId* . Per ulteriori informazioni, consultare “ID SubCorrel(MQBYTE24) per MQSD” a pagina 599 .

Questa opzione non può essere combinata con MQSO_MANAGED.

MQSO_SET_IDENTITY_CONTEXT

La sottoscrizione utilizza il token di account e i dati di identità dell'applicazione forniti nei campi *PubAccountingToken* e *PubApplIdentityData* .

Se viene specificata questa opzione, viene eseguito lo stesso controllo di autorizzazione come se si accedesse alla coda di destinazione utilizzando una chiamata MQOPEN con MQOO_SET_IDENTITY_CONTEXT, tranne nel caso in cui venga utilizzata anche l'opzione MQSO_MANAGED, nel qual caso non vi è alcun controllo di autorizzazione sulla coda di destinazione.

Se questa opzione non viene specificata, le pubblicazioni inviate a questo sottoscrittore hanno le informazioni di contesto predefinite associate come segue:

Tabella 530. Informazioni di contesto predefinite per pubblicazioni inviate a questo sottoscrittore

Campo in MQMD	**Valore utilizzato**
UserIdentifier	L'ID utente associato alla sottoscrizione nel momento in cui è stata effettuata la sottoscrizione.
AccountingToken	Determinato dall'ambiente, se possibile; impostare su MQACT_NONE in caso contrario.
ApplIdentityData	Imposta su spazi vuoti

Questa opzione è valida solo con MQSO_CREATE e MQSO_ALTER. Se utilizzato con MQSO_RESUME, i campi *PubAccountingToken* e *PubApplIdentityData* vengono ignorati, quindi questa opzione non ha alcun effetto.

Se una sottoscrizione viene modificata senza utilizzare questa opzione dove precedentemente la sottoscrizione ha fornito le informazioni di contesto di identità, vengono generate le informazioni di contesto predefinite per la sottoscrizione modificata.

Se una sottoscrizione che consente a ID utente differenti di utilizzarla con l'opzione MQSO_ANY_USERID, viene ripresa da un ID utente differente, viene generato il contesto di identità predefinito per il nuovo ID utente che ora possiede la sottoscrizione e vengono consegnate le pubblicazioni successive contenenti il nuovo contesto di identità.

MQSO_FAIL_IF_QUIESCING

La chiamata MQSUB ha esito negativo se il gestore code è in stato di inattività. Su z/OS, per un'applicazione CICS o IMS , questa opzione forza anche l'esito negativo della chiamata MQSUB se la connessione è in stato di inattività.

ObjectName (MQCHAR48) per MQSD

È il nome dell'oggetto argomento come definito sul gestore code locale.

Il nome può contenere i seguenti caratteri:

- Caratteri alfabetici maiuscoli (da A a Z)
- Caratteri alfabetici minuscoli (da a a z)
- Cifre numeriche (da 0 a 9)
- Punto (.), barra (/), sottolineatura (_), percentuale (%)

Il nome non deve contenere spazi iniziali o intermedi, ma può contenere spazi finali. Utilizzare un carattere null per indicare la fine dei dati significativi nel nome; il valore null e i caratteri che lo seguono vengono trattati come spazi vuoti. Le seguenti limitazioni si applicano agli ambienti indicati:

- Sui sistemi che utilizzano EBCDIC Katakana, non è possibile utilizzare caratteri minuscoli.
- Su z/OS:
  - Evitare i nomi che iniziano o terminano con un carattere di sottolineatura; non possono essere elaborati dalle operazioni e dai pannelli di controllo.
  - Il carattere percentuale ha un significato speciale per RACF. Se RACF viene utilizzato come gestore della sicurezza esterno, i nomi non devono contenere la percentuale. In tal caso, tali nomi non vengono inclusi nei controlli di sicurezza quando vengono utilizzati i profili generici RACF .
- Su IBM i, i nomi contenenti caratteri minuscoli, barra o percentuale, devono essere racchiusi tra virgolette quando vengono specificati nei comandi. Questi apici non devono essere specificati per i nomi che si verificano come campi nelle strutture o come parametri nelle chiamate.

ObjectName viene utilizzato per formare il nome completo dell'argomento.

Il nome completo dell'argomento può essere creato da due campi differenti: *ObjectName* e *ObjectString*. Per i dettagli sul modo in cui questi due campi vengono utilizzati, consultare Combinazione di stringhe argomento.

Se non è possibile trovare l'oggetto identificato dal campo *ObjectName* , la chiamata ha esito negativo con codice motivo MQRC_UNKNOWN_OBJECT_NAME anche se è presente una stringa specificata in *ObjectString*.

Al ritorno da una chiamata MQSUB utilizzando l'opzione MQSO_RESUME, questo campo non viene modificato.

La lunghezza di questo campo è fornita da MQ_TOPIC_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

Se si altera una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, il nome dell'oggetto argomento sottoscritto non può essere modificato. Questo campo e il campo *ObjectString* possono essere omessi. Se vengono forniti, devono risolversi nello stesso nome completo dell'argomento. In caso contrario, la chiamata ha esito negativo con MQRC_TOPIC_NOT_ALTERABLE.

ID AlternateUser(MQCHAR12) per MQSD

Se si specifica MQSO_ALTERNATE_USER_AUTHORITY, questo campo contiene un identificativo utente alternativo utilizzato per controllare l'autorizzazione per la sottoscrizione e per l'output nella coda di destinazione (specificato nel parametro **Hobj** della chiamata MQSUB), al posto dell'identificativo utente con cui l'applicazione è attualmente in esecuzione.

Se l'operazione ha esito positivo, l'identificativo utente specificato in questo campo viene registrato come l'identificativo utente proprietario della sottoscrizione al posto dell'identificativo utente con cui è attualmente in esecuzione l'applicazione.

Se viene specificato MQSO_ALTERNATE_USER_AUTHORITY e questo campo è completamente vuoto fino al primo carattere null o alla fine del campo, la sottoscrizione può avere esito positivo solo se non è necessaria alcuna autorizzazione utente per la sottoscrizione a questo argomento con le opzioni specificate o la coda di destinazione per l'output.

Se MQSO_ALTERNATE_USER_AUTHORITY non viene specificato, questo campo viene ignorato.

Le seguenti differenze esistono negli ambienti indicati:

- Su z/OS, per controllare l'autorizzazione per la sottoscrizione vengono utilizzati solo i primi 8 caratteri dell'ID AlternateUser. Tuttavia, l'identificativo utente corrente deve essere autorizzato a specificare questo particolare identificativo utente alternativo; per questo controllo vengono utilizzati tutti i 12 caratteri dell'identificativo utente alternativo. L'identificativo utente deve contenere solo i caratteri consentiti dal gestore della sicurezza esterno.

Al ritorno da una chiamata MQSUB utilizzando MQSO_RESUME, questo campo non viene modificato.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da MQ_USER_ID_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 12 caratteri vuoti in altri linguaggi di programmazione.

AlternateSecurityID (MQBYTE40) per MQSD

Questo è un identificativo di sicurezza che viene passato con l'ID AlternateUseral servizio di autorizzazione per consentire l'esecuzione di controlli di autorizzazione appropriati.

L'ID AlternateSecurityviene utilizzato solo se viene specificato MQSO_ALTERNATE_USER_AUTHORITY e il campo ID AlternateUsernon è completamente vuoto fino al primo carattere null o alla fine del campo.

Al ritorno da una chiamata MQSUB utilizzando MQSO_RESUME, questo campo non viene modificato.

Per ulteriori informazioni, consultare la descrizione di “ID AlternateSecurity(MQBYTE40) per MQOD” a pagina 504 nel tipo di dati MQOD.

SubExpiry (MQLONG) per MQSD

Questo è il tempo espresso in decimi di secondo dopo il quale scade la sottoscrizione. Nessun' altra pubblicazione corrisponderà a questa sottoscrizione una volta trascorso questo intervallo. Non appena una sottoscrizione scade, le pubblicazioni non vengono più inviate alla coda. Tuttavia, le pubblicazioni già presenti non sono influenzate in alcun modo. *SubExpiry* non ha effetto sulla scadenza della pubblicazione.

Viene riconosciuto il seguente valore speciale:

MQEI_UNLIMITED
La sottoscrizione ha un tempo di scadenza illimitato.

Se si modifica una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, è possibile modificare la scadenza della sottoscrizione.

Al ritorno da una chiamata MQSUB utilizzando l'opzione MQSO_RESUME, questo campo è impostato sulla scadenza originale della sottoscrizione e non sul tempo di scadenza rimanente.

ObjectString (MQCHARV) per MQSD

Questo è il nome oggetto lungo da utilizzare.

Il *ObjectString* viene utilizzato per formare il nome completo dell'argomento.

Il nome completo dell'argomento può essere creato da due campi differenti: *ObjectName* e *ObjectString*. Per i dettagli sul modo in cui questi due campi vengono utilizzati, consultare Combinazione di stringhe argomento.

La lunghezza massima di *ObjectString* è 10240.

Se *ObjectString* non viene specificato correttamente, in base alla descrizione di come utilizzare la struttura MQCHARV o se supera la lunghezza massima, la chiamata ha esito negativo con codice di errore MQRC_OBJECT_STRING_ERROR.

Questo è un campo di immissione. I valori iniziali dei campi in questa struttura sono gli stessi della struttura MQCHARV.

Se sono presenti caratteri jolly in *ObjectString* , l'interpretazione di tali caratteri jolly può essere controllata utilizzando le opzioni dei caratteri jolly specificate nel campo Opzioni di MQSD.

Al ritorno da una chiamata MQSUB utilizzando l'opzione MQSO_RESUME, questo campo non viene modificato. Il nome completo dell'argomento utilizzato viene restituito nel campo *ResObjectString* se viene fornito un buffer.

Se si modifica una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, il nome esteso dell'oggetto argomento sottoscritto non può essere modificato. Questo campo e il campo *ObjectName* possono essere omessi. Se vengono forniti, devono essere risolti con lo stesso nome argomento completo oppure la chiamata ha esito negativo con MQRC_TOPIC_NOT_ALTERABLE.

SubName (MQCHARV) per MQSD

Specifica il nome della sottoscrizione. Questo campo è richiesto solo se *Options* specifica l'opzione MQSO_DURABLE, ma se fornita verrà utilizzata anche dal gestore code per MQSO_NON_DURABLE.

Se specificato, *SubName* deve essere univoco all'interno del gestore code, poiché è il metodo utilizzato per identificare la sottoscrizione.

La lunghezza massima di *SubName* è 10240.

Questo campo ha due scopi. Per una sottoscrizione MQSO_DURABLE, utilizzare questo campo per identificare una sottoscrizione in modo da poterla riprendere una volta creata se è stato chiuso l'handle per la sottoscrizione (utilizzando l'opzione MQCO_KEEP_SUB) o se è stata disconnessa dal gestore code. Questa operazione viene eseguita utilizzando la chiamata MQSUB con l'opzione MQSO_RESUME. Viene visualizzato anche nella visualizzazione amministrativa delle sottoscrizioni nel campo SUBID in DISPLAY SBSTATUS.

Se *SubName* viene specificato in modo non corretto, in base alla descrizione di come utilizzare la struttura MQCHARV , viene tralasciato quando è richiesto (ovvero *SubName*. *VSLength* è zero) o, se supera la lunghezza massima, la chiamata ha esito negativo con codice motivo MQRC_SUB_NAME_ERROR.

Questo è un campo di immissione. I valori iniziali dei campi in questa struttura sono gli stessi della struttura MQCHARV.

Se si modifica una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, il nome della sottoscrizione non può essere modificato, perché è il campo di identificazione utilizzato per trovare la sottoscrizione di riferimento. Non viene modificato nell'output da una chiamata MQSUB con l'opzione MQSO_RESUME.

Dati SubUser(MQCHARV) per MQSD

Specifica i dati utente della sottoscrizione. I dati forniti nella sottoscrizione in questo campo verranno inclusi come proprietà del messaggio di dati MQSubUserdi ogni pubblicazione inviata a questa sottoscrizione.

La lunghezza massima di *SubUserData* è 10240.

Se *SubUserData* viene specificato in modo non corretto, in base alla descrizione di come utilizzare la struttura MQCHARV o se supera la lunghezza massima, la chiamata ha esito negativo con codice motivo MQRC_SUB_USER_DATA_ERROR.

Questo è un campo di immissione. I valori iniziali dei campi in questa struttura sono gli stessi della struttura MQCHARV.

Se si modifica una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, è possibile modificare i dati utente della sottoscrizione.

Questo campo a lunghezza variabile viene restituito all'output di una chiamata MQSUB utilizzando l'opzione MQSO_RESUME, se viene fornito un buffer e in *VSBufLen*è presente una lunghezza buffer positiva. Se nella chiamata non viene fornito alcun buffer, nel campo *VSLength* di MQCHARV viene

restituita solo la lunghezza della data dell'utente della sottoscrizione. Se il buffer fornito è più piccolo dello spazio richiesto per restituire il campo, nel buffer fornito vengono restituiti solo *VSBufLen* byte.

ID SubCorrel(MQBYTE24) per MQSD

Questo campo contiene un identificativo di correlazione comune a tutte le pubblicazioni che corrispondono a questa sottoscrizione.

Attenzione: un identificatore di correlazione può essere passato solo tra i gestori code in un cluster di pubblicazione / sottoscrizione, non una gerarchia.

Tutte le pubblicazioni inviate per corrispondere a questa sottoscrizione contengono questo identificativo di correlazione nel descrizione del messaggio. Se più sottoscrizioni richiamano le proprie pubblicazioni dalla stessa coda, l'utilizzo di MQGET mediante l'identificativo di correlazione consente di ottenere solo le pubblicazioni per una specifica sottoscrizione. Questo identificativo di correlazione può essere generato dal gestore code o dall'utente.

Se l'opzione MQSO_SET_CORREL_ID non è specificata, l'identificativo di correlazione viene generato dal gestore code e questo campo è un campo di output contenente l'identificativo di correlazione che verrà impostato in ogni messaggio pubblicato per questa sottoscrizione. L'identificativo di correlazione generato è costituito da un identificativo di prodotto a 4 byte (AMQX o CSQM in ASCII o EBCDIC) seguito da un'implementazione specifica del prodotto di una stringa univoca.

Se l'opzione MQSO_SET_CORREL_ID è specificata, l'identificativo di correlazione viene generato dall'utente e questo campo è un campo di input contenente l'identificativo di correlazione da impostare in ogni pubblicazione per questa sottoscrizione. In questo caso, se il campo contiene MQCI_NONE, l'identificatore di correlazione impostato in ogni messaggio pubblicato per questa sottoscrizione è l'identificatore di correlazione creato dall'inserimento originale del messaggio.

Se viene specificata l'opzione MQSO_GROUP_SUB e l'identificatore di correlazione specificato è lo stesso di una sottoscrizione raggruppata esistente che utilizza la stessa coda e una stringa di argomento sovrapposta, solo la sottoscrizione più significativa nel gruppo viene fornita con una copia della pubblicazione.

La lunghezza di questo campo è fornita da MQ_CORREL_ID_LENGTH. Il valore iniziale di questo campo è MQCI_NONE.

Se si sta modificando una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, e questo campo è un campo di input, l'identificativo di correlazione della sottoscrizione può essere modificato, a meno che la sottoscrizione non sia una sottoscrizione raggruppata, ovvero, è stata creata utilizzando l'opzione MQSO_GROUP_SUB, nel qual caso l'identificativo di correlazione della sottoscrizione non può essere modificato.

Al ritorno da una chiamata MQSUB utilizzando MQSO_RESUME, questo campo è impostato sull'identificativo di correlazione corrente per la sottoscrizione.

PubPriority (MQLONG) per MQSD

Questo è il valore che sarà nel campo *Priority* di MQMD (Message Descriptor) di tutti i messaggi di pubblicazione corrispondenti a questa sottoscrizione. Per ulteriori informazioni sul campo *Priority* in MQMD, consultare “Priorità (MQLONG) per MQMD” a pagina 462.

Il valore deve essere maggiore o uguale a zero; zero è la priorità più bassa. È inoltre possibile utilizzare i seguenti valori speciali:

MQPRI_PRIORITY_AS_Q_DEF

Quando una coda di sottoscrizione viene fornita nel campo *Hobj* nella chiamata MQSUB e non è un handle gestito, la priorità per il messaggio viene presa dall'attributo **DefPriority** di questa coda. Se la coda è una coda cluster o esiste più di una definizione nel percorso di risoluzione del nome della coda, la priorità viene determinata quando il messaggio di pubblicazione viene inserito nella coda come descritto per “Priorità (MQLONG) per MQMD” a pagina 462.

Se la chiamata MQSUB utilizza un handle gestito, la priorità per il messaggio viene ricavata dall'attributo **DefPriority** della coda modello associata all'argomento sottoscritto.

MQPRI_PRIORITY_AS_PUBLISHED

La priorità del messaggio è la priorità della pubblicazione originale. Questo è il valore iniziale del campo.

Se si modifica una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, è possibile modificare il *Priority* di eventuali messaggi di pubblicazione futuri.

Al ritorno da una chiamata MQSUB utilizzando MQSO_RESUME, questo campo è impostato sulla priorità corrente utilizzata per la sottoscrizione.

Token PubAccounting(MQBYTE32) per MQSD

Questo è il valore che sarà nel campo *AccountingToken* di MQMD (Message Descriptor) di tutti i messaggi di pubblicazione corrispondenti a questa sottoscrizione. *AccountingToken* fa parte del contesto di identità del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio. Per ulteriori informazioni sul campo *AccountingToken* in MQMD, consultare "AccountingToken (MQBYTE32) per MQMD" a pagina 471

È possibile utilizzare il seguente valore speciale per il campo *PubAccountingToken* :

MQACT_NONE

Nessun token di account specificato.

Il valore è zero binario per la lunghezza del campo.

Per il linguaggio di programmazione C, viene definita anche la costante MQACT_NONE_ARRAY, che ha lo stesso valore di MQACT_NONE, ma è un insieme di caratteri anziché una stringa.

Se l'opzione MQSO_SET_IDENTITY_CONTEXT non viene specificata, il token di account viene generato dal gestore code come informazioni di contesto predefinite e questo campo è un campo di output che contiene il *AccountingToken* che verrà impostato in ogni messaggio pubblicato per questa sottoscrizione.

Se viene specificata l'opzione MQSO_SET_IDENTITY_CONTEXT, il token di account viene generato dall'utente e questo campo è un campo di input che contiene il *AccountingToken* da impostare in ogni pubblicazione per questa sottoscrizione.

La lunghezza di questo campo è fornita da MQ_ACCOUNTING_TOKEN_LENGTH. Il valore iniziale di questo campo è MQACT_NONE.

Se si altera una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, il valore di *AccountingToken* in eventuali messaggi di pubblicazione futuri può essere modificato.

Al ritorno da una chiamata MQSUB utilizzando MQSO_RESUME, questo campo è impostato sul *AccountingToken* corrente utilizzato per la sottoscrizione.

PubApplIdentityData (MQCHAR32) per MQSD

Questo è il valore che si trova nel campo *ApplIdentityData* di MQMD (Message Descriptor) di tutti i messaggi di pubblicazione corrispondenti a questa sottoscrizione. *ApplIdentityData* fa parte del contesto di identità del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio. Per ulteriori informazioni sul campo *ApplIdentityData* in MQMD, consultare "Dati ApplIdentity(MQCHAR32) per MQMD" a pagina 472

Se l'opzione MQSO_SET_IDENTITY_CONTEXT non viene specificata, il *ApplIdentityData* impostato in ogni messaggio pubblicato per questa sottoscrizione è vuoto, come informazioni di contesto predefinite.

Se viene specificata l'opzione MQSO_SET_IDENTITY_CONTEXT, il *PubApplIdentityData* viene generato dall'utente e questo campo è un campo di input che contiene il *ApplIdentityData* da impostare in ciascuna pubblicazione per questa sottoscrizione.

La lunghezza di questo campo è fornita da MQ_APPL_IDENTITY_DATA_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 32 caratteri vuoti in altri linguaggi di programmazione.

Se si modifica una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, è possibile modificare il *ApplIdentityData* di eventuali messaggi di pubblicazione futuri.

Al ritorno da una chiamata MQSUB utilizzando MQSO_RESUME, questo campo è impostato sul *ApplIdentityData* corrente utilizzato per la sottoscrizione.

SelectionString (MQCHARV) per MQSD

Questa è la stringa utilizzata per fornire i criteri di selezione utilizzati durante la sottoscrizione per i messaggi da un argomento.

Questo campo di lunghezza variabile verrà restituito all'output da una chiamata MQSUB utilizzando l'opzione MQSO_RESUME, se viene fornito un buffer, ed è presente anche una lunghezza del buffer positiva in VSBufSize. Se nella chiamata non viene fornito alcun buffer, nel campo VSLength di MQCHARV verrà restituita solo la lunghezza della stringa di selezione. Se il buffer fornito è inferiore allo spazio richiesto per restituire il campo, nel buffer fornito vengono restituiti solo VSBufSize byte.

Se *SelectionString* viene specificato in modo non corretto, in base alla descrizione di come utilizzare la struttura "MQCHARV - Stringa a lunghezza variabile" a pagina 299 o se supera la lunghezza massima, la chiamata ha esito negativo con codice motivo MQRC_SELECTION_STRING_ERROR.

L'utilizzo di SelectionString è descritto in Selettori.

SubLevel (MQLONG) per MQSD

Questo è il livello associato alla sottoscrizione. Le pubblicazioni vengono consegnate a questa sottoscrizione solo se si trova nella serie di sottoscrizioni con il valore SubLevel più alto minore o uguale al PubLevel utilizzato al momento della pubblicazione. Tuttavia, se una pubblicazione è stata conservata, non è più disponibile per i sottoscrittori a livelli superiori perché viene ripubblicata in PubLevel 1.

Il valore deve essere compreso tra zero e 9. Zero è il livello più basso.

Il valore iniziale di questo campo è 1.

Per ulteriori informazioni, consultare Intercettazione delle pubblicazioni.

Se si modifica una sottoscrizione esistente utilizzando l'opzione MQSO_ALTER, il SubLevel non può essere modificato.

La combinazione di un SubLevel con un valore maggiore di 1 con l'opzione MQSO_PUBLICATIONS_ON_REQUEST non è consentita.

Al ritorno da una chiamata MQSUB che utilizza MQSO_RESUME, questo campo è impostato al livello corrente utilizzato per la sottoscrizione.

Stringa ResObject(MQCHARV) per MQSD

Questo è il nome oggetto lungo dopo che il gestore code ha risolto il nome fornito in *ObjectName*.

Se il nome oggetto lungo viene fornito in *ObjectString* e non viene fornito nulla in *ObjectName*, il valore restituito in questo campo è uguale a quello fornito in *ObjectString*.

Se questo campo viene omesso (ovvero ResObjectString.VSBufSize è zero), *ResObjectString* non viene restituito, ma la lunghezza viene restituita in ResObjectString.VSLength. Se la lunghezza è inferiore alla stringa ResObjectcompleta, viene troncata e restituisce il numero massimo di caratteri a destra che possono rientrare nella lunghezza fornita.

Se *ResObjectString* non viene specificato correttamente, in base alla descrizione di come utilizzare la struttura MQCHARV o se supera la lunghezza massima, la chiamata ha esito negativo con codice motivo MQRC_RES_OBJECT_STRING_ERROR.

MQSMPO - Imposta opzioni proprietà messaggio

La struttura **MQSMPO** consente alle applicazioni di specificare le opzioni che controllano il modo in cui vengono impostati i messaggi. La struttura è un parametro di immissione sulla chiamata **MQSETMP** .

Disponibilità

Tutti i sistemi IBM MQ e client IBM MQ .

Serie di caratteri e codifica

I dati in **MQSMPO** devono essere nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (**MQENC_NATIVE**).

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 531. Campi in MQSMPO

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQSMPO	'SMPO'
Versione (numero versione struttura)	MQSMPO_VERSION_1	1
Opzioni (opzioni)	MQSMPO_NONE	0
ValueEncoding (codifica valore proprietà)	MQEN_NATIVE	Dipende dall'ambiente
ValueCCSID (serie di caratteri valore proprietà)	APPL MQCCSI	-3

Note:

1. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.
2. Nel linguaggio di programmazione C, la variabile macroMQSMPO_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura:

```
MQSMPO MySMPO = {MQSMPO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQSMPO

```
typedef struct tagMQSMPO MQSMPO;
struct tagMQSMPO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of MQSETMP */
  MQLONG    ValueEncoding;   /* Encoding of Value */
  MQLONG    ValueCCSID;      /* Character set identifier of Value */
};
```

Dichiarazione COBOL per MQSMPO

```
**   MQSMPO structure
```

```
  10 MQSMPO.
**    Structure identifier
   15 MQSMPO-STRUCID       PIC X(4).
**    Structure version number
   15 MQSMPO-VERSION       PIC S9(9) BINARY.
**    Options that control the action of MQSETMP
   15 MQSMPO-OPTIONS       PIC S9(9) BINARY.
**    Encoding of VALUE
   15 MQSMPO-VALUEENCODING PIC S9(9) BINARY.
**    Character set identifier of VALUE
   15 MQSMPO-VALUECCSID    PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQSMPO

```
dcl
 1 MQSMPO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action of MQSETMP */
  3 ValueEncoding  fixed bin(31), /* Encoding of Value */
  3 ValueCCSID     fixed bin(31), /* Character set identifier of Value */
```

Dichiarazione High Level Assembler per MQSMPO

```
MQSMPO                DSECT
MQSMPO_STRUCID        DS   CL4   Structure identifier
MQSMPO_VERSION        DS   F     Structure version number
MQSMPO_OPTIONS        DS   F     Options that control the action of
*                                MQSETMP
MQSMPO_VALUEENCODING  DS   F     Encoding of VALUE
MQSMPO_VALUECCSID     DS   F     Character set identifier of VALUE
MQSMPO_LENGTH         EQU  *-MQSMPO
MQSMPO_AREA           DS   CL(MQSMPO_LENGTH)
```

StrucId (MQCHAR4) per MQSMPO

Questo è l'identificativo della struttura delle opzioni della proprietà di impostazione del messaggio. È sempre un campo di immissione. Il valore è MQSMPO_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQSMPO

L'identificativo per la struttura di opzioni della proprietà del messaggio impostato.

Per il linguaggio di programmazione C, viene definita anche la costante **MQSMPO_STRUC_ID_ARRAY** . Ha lo stesso valore di **MQSMPO_STRUC_ID**, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQSMPO

Questo è il numero di versione della struttura; il valore deve essere:

MQSMPO_VERSION_1
Version-1 imposta la struttura delle opzioni delle proprietà del messaggio.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQSMPO_CURRENT_
Versione corrente della struttura di opzioni della proprietà del messaggio impostato.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **MQSMPO_VERSION_1**.

Opzioni (MQLONG) per MQSMPO

Opzioni di ubicazione

Le opzioni riportate di seguito sono relative alla posizione relativa della proprietà rispetto al cursore della proprietà:

MQSMPO_SET_FIRST
Imposta il valore della prima proprietà che corrisponde al nome specificato oppure, se non esiste, aggiunge una nuova proprietà dopo tutte le altre proprietà con una gerarchia corrispondente.

MQSMPO_SET_PROP_UNDER_CURSOR
Imposta il valore della proprietà a cui punta il cursore della proprietà. La proprietà indicata dal cursore della proprietà è quella che è stata interrogata l'ultima volta utilizzando l'opzione MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

Il cursore della proprietà viene reimpostato quando la gestione del messaggio viene riutilizzata su una chiamata MQGET o quando la gestione del messaggio viene specificata nel campo *MsgHandle* della struttura MQGMO o MQPMO su una chiamata MQPUT.

Se questa opzione viene utilizzata quando il cursore della proprietà non è ancora stato stabilito o se il puntatore della proprietà al cursore della proprietà è stato eliminato, la chiamata ha esito negativo con codice di completamento MQCC_FAILED e codice motivo MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_SET_PROP_BEFORE_CURSOR
Imposta una nuova proprietà prima della proprietà indicata dal cursore della proprietà. La proprietà indicata dal cursore della proprietà è quella che è stata interrogata l'ultima volta utilizzando l'opzione MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

Il cursore della proprietà viene reimpostato quando la gestione del messaggio viene riutilizzata su una chiamata MQGET o quando la gestione del messaggio viene specificata nel campo *MsgHandle* della struttura MQGMO o MQPMO su una chiamata MQPUT.

Se questa opzione viene utilizzata quando il cursore della proprietà non è ancora stato stabilito o se il puntatore della proprietà al cursore della proprietà è stato eliminato, la chiamata ha esito negativo con codice di completamento MQCC_FAILED e codice motivo MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_SET_PROP_AFTER_CURSOR
Imposta una nuova proprietà dopo la proprietà indicata dal cursore della proprietà. La proprietà indicata dal cursore della proprietà è quella che è stata interrogata l'ultima volta utilizzando l'opzione MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

Il cursore della proprietà viene reimpostato quando la gestione del messaggio viene riutilizzata su una chiamata MQGET o quando la gestione del messaggio viene specificata nel campo *MsgHandle* della struttura MQGMO o MQPMO su una chiamata MQPUT.

Se questa opzione viene utilizzata quando il cursore della proprietà non è ancora stato stabilito o se il puntatore della proprietà al cursore della proprietà è stato eliminato, la chiamata ha esito negativo con codice di completamento MQCC_FAILED e codice motivo MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_APPEND_PROPERTY
Fa sì che una nuova proprietà venga aggiunta dopo tutte le altre proprietà con una gerarchia corrispondente. Se esiste almeno una proprietà che corrisponde al nome specificato, viene aggiunta una nuova proprietà alla fine dopo la fine di tale elenco di proprietà.

Questa opzione consente di creare un elenco di proprietà con lo stesso nome.

Se non è necessaria alcuna delle opzioni descritte, utilizzare la seguente opzione:

MQSMPO_NONE
Nessuna opzione specificata.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQSMPO_SET_FIRST.

ValueEncoding (MQLONG) per MQSMPO

La codifica del valore della proprietà da impostare se il valore è numerico.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **MQENC_NATIVE**.

ValueCCSID (MQLONG) per MQSMPO

La serie di caratteri del valore della proprietà da impostare se il valore è una stringa di caratteri.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **MQCCSI_APPL**.

MQSRO - Opzioni di richieste di sottoscrizione

La struttura MQSRO consente di specificare le opzioni che controllano il modo in cui viene effettuata una richiesta di sottoscrizione. La struttura è un parametro di input / output sulla chiamata MQSUBRQ.

Disponibilità

La struttura MQSRO è disponibile sulle piattaforme seguenti:

- AIX
- IBM i
- Linux
- Windows
- z/OS

e per IBM MQ MQI clients collegati a questi sistemi.

Versione

La versione corrente di MQSRO è MQSRO_VERSION_1.

Serie di caratteri e codifica

I dati in MQSRO devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da MQENC_NATIVE. Tuttavia, se l'applicazione è in esecuzione come client MQ MQI, la struttura deve essere nella serie di caratteri e nella codifica del client.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQSRO	'SRO¬'
Versione (numero versione struttura)	MQSRO_VERSION_1	1
Opzioni (opzioni)	MQSRO_NONE	0
NumPubs (numero di pubblicazioni)	Nessuna	0

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macroMQSRO_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura:

```
MQSRO MySRO = {MQSRO_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQSRO

```
typedef struct tagMQSRO MQSRO;
struct tagMQSRO {
  MQCHAR4   StrucId;              /* Structure identifier */
  MQLONG    Version;              /* Structure version number */
  MQLONG    Options;              /* Options that control the action of MQSUBRQ */
  MQLONG    NumPubs;              /* Number of publications sent */
  /* Ver:1 */
};
```

Dichiarazione COBOL per MQSRO

```
** MQSRO structure
 10  MQSRO.
** Structure identifier
 15  MQSRO-STRUCID             PIC X(4).
** Structure version number
 15  MQSRO-VERSION             PIC S9(9) BINARY.
** Options that control the action of MQSUBRQ
 15  MQSRO-OPTIONS             PIC S9(9) BINARY.
** Number of publications sent
 15  MQSRO-NUMPUBS             PIC S9(9) BINARY.
```

Dichiarazione PL/I per MQSRO

```
dcl
 1 MQSRO based,
   3 StrucId        char(4),       /* Structure identifier */
   3 Version        fixed bin(31), /* Structure version number */
   3 Options        fixed bin(31), /* Options that control the action of MQSUBRQ */
   3 NumPubs        fixed bin(31); /* Number of publications sent */
```

Dichiarazione High Level Assembler per MQSRO

```
MQSRO                   DSECT
MQSRO_STRUCID           DS    CL4   Structure identifier
MQSRO_VERSION           DS    F     Structure version number
MQSRO_OPTIONS           DS    F     Options that control the action of MQSUBRQ
MQSRO_NUMPUBS           DS    F     Number of publications sent
*
MQSRO_LENGTH            EQU   *-MQSRO
                        ORG   MQSRO
MQSRO_AREA              DS    CL(MQSRO_LENGTH)
```

StrucId (MQCHAR4) per MQSRO

Questo è l'identificativo della struttura di opzioni della richiesta di sottoscrizione. È sempre un campo di immissione. Il valore è MQSRO_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQSRO
Identificativo per la struttura di opzioni della richiesta di sottoscrizione.

Per il linguaggio di programmazione C, viene definita anche la costante MQSRO_STRUC_ID_ARRAY. Ha lo stesso valore di MQSRO_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQSRO

Questo è il numero di versione della struttura; il valore deve essere:

MQSRO_VERSION_1
Version-1 Struttura Opzioni richiesta di sottoscrizione.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQSRO_CURRENT_
La versione corrente della struttura delle opzioni della richiesta di sottoscrizione.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MQSRO_VERSION_1.

Opzioni (MQLONG) per MQSRO

È necessario specificare una delle seguenti opzioni. È possibile specificare una sola opzione.

MQSRO_FAIL_IF_QUIESCING

La chiamata MQSUBRQ ha esito negativo se il gestore code è in stato di inattività. Su z/OS, per un'applicazione CICS o IMS , questa opzione forza anche l'esito negativo della chiamata MQSUBRQ se la connessione è in uno stato di inattività.

Opzione predefinita: se l'opzione descritta precedentemente non è richiesta, è necessario utilizzare la seguente opzione:

MQSRO_NONE

Utilizzare questo valore per indicare che non sono state specificate altre opzioni; tutte le opzioni assumono i propri valori predefiniti.

MQSRO_NONE aiuta la documentazione del programma. Sebbene non sia previsto che questa opzione venga utilizzata con altre, poiché il suo valore è zero, questo utilizzo non può essere rilevato.

NumPubs (MQLONG) per MQSRO

Si tratta di un campo di emissione, restituito all'applicazione per indicare il numero di pubblicazioni inviate alla coda di sottoscrizione come risultato di questa chiamata. Sebbene questo numero di pubblicazioni sia stato inviato come risultato di questa chiamata, non vi è alcuna garanzia che questo numero di messaggi sarà disponibile per l'applicazione, soprattutto se si tratta di messaggi non persistenti.

Potrebbe essere presente più di una pubblicazione se l'argomento sottoscritto conteneva un carattere jolly. Se non erano presenti caratteri jolly nella stringa dell'argomento quando è stata creata la sottoscrizione rappresentata da *Hsub* , viene inviata al massimo una pubblicazione come risultato di questa chiamata.

MQSTS - Struttura di report di stato

La struttura MQSTS è un parametro di output del comando MQSTAT. Il comando MQSTAT viene utilizzato per richiamare le informazioni sullo stato. Queste informazioni vengono restituite in una struttura MQSTS.

Serie di caratteri e codifica

I dati carattere in MQSTS si trovano nella serie di caratteri del gestore code locale; ciò viene fornito dall'attributo del Gestore code *CodedCharSetId* . I dati numerici in MQSTS sono nella codifica della macchina nativa; ciò è fornito da *Codifica*.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 532. Campi in MQSTS

Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
StrucId (identificativo della struttura)	ID_STRUC_MQST	'STAT¬'
Versione (numero versione struttura)	MQSTS_VERSION_1	1
CompCode (codice di completamento del primo errore)	MQCC_OK	0
Motivo (codice motivo del primo errore)	MQRC_NONE	0
PutSuccessConteggio (numero di chiamate di inserimento asincrone riuscite)	Nessuna	0
PutWarningCount (numero di chiamate di immissione asincrona che avevano avvertenze)	Nessuna	0
PutFailureConteggio (numero di chiamate di inserimento asincrone non riuscite)	Nessuna	0
ObjectType (tipo di oggetto in errore)	MQOT_Q	1
ObjectName (nome dell'oggetto in errore)	Nessuna	Stringa null o spazi vuoti
ObjectQMgrNome (nome del gestore code proprietario dell'oggetto in errore	Nessuna	Stringa null o spazi vuoti
ResolvedObjectNome (nome risolto della coda di destinazione)	Nessuna	Stringa null o spazi vuoti
ResolvedQMgrNome (nome risolto del gestore code di destinazione)	Nessuna	Stringa null o spazi vuoti
Nota: I campi rimanenti vengono ignorati se la versione è inferiore a MQSTS_VERSION_2.		
ObjectString (nome oggetto lungo dell'oggetto in errore)	MQCHARV_PREDEFINITO	{NULL,0,0,0,-3}
SubName (nome sottoscrizione della sottoscrizione non riuscita)	MQCHARV_PREDEFINITO	{NULL,0,0,0,-3}
OpenOptions (opzioni aperte associate all'errore)	Nessuna	0
SubOptions(opzioni di sottoscrizione associate all'errore)	Nessuna	0

Tabella 532. Campi in MQSTS (Continua)

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.
3. Nel linguaggio di programmazione C, la variabile macro MQSTS_DEFAULT contiene i valori elencati nella tabella. Può essere utilizzato nel seguente modo per fornire valori iniziali per i campi nella struttura:

```
MQSTS MySTS = {MQSTS_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQSTS

```
typedef struct tagMQSTS MQSTS;
struct tagMQSTS {
 MQCHAR4   StrucId;            /* Structure identifier */
 MQLONG    Version;            /* Structure version number */
 MQLONG    CompCode;           /* Completion Code of first error */
 MQLONG    Reason;             /* Reason Code of first error */
 MQLONG    PutSuccessCount;    /* Number of Async calls succeeded */
 MQLONG    PutWarningCount;    /* Number of Async calls had warnings */
 MQLONG    PutFailureCount;    /* Number of Async calls had failures */
 MQLONG    ObjectType;         /* Failing object type */
 MQCHAR48  ObjectName;         /* Failing object name */
 MQCHAR48  ObjectQMgrName;     /* Failing object queue manager name */
 MQCHAR48  ResolvedObjectName; /* Resolved name of destination queue */
 MQCHAR48  ResolvedQMgrName;   /* Resolved name of destination qmgr */
/* Ver:1 */
 MQCHARV   ObjectString;       /* Failing object long name */
 MQCHARV   SubName;            /* Failing subscription name */
 MQLONG    OpenOptions;        /* Failing open options */
 MQLONG    SubOptions;         /* Failing subscription options */
/* Ver:2 */
};
```

Dichiarazione COBOL per MQSTS

```
      ** MQSTS structure
       10  MQSTS.
      ** Structure identifier
       15  MQSTS-STRUCID PIC X(4).
      ** Structure version number
       15  MQSTS-VERSION PIC S9(9) BINARY.
      ** Completion Code of first error
       15  MQSTS-COMPCODE PIC S9(9) BINARY.
      ** Reason Code of first error
       15  MQSTS-REASON PIC S9(9) BINARY.
      ** Number of Async put calls succeeded
       15  MQSTS-PUTSUCCESSCOUNT PIC S9(9) BINARY.
      ** Number of Async put calls had warnings
       15  MQSTS-PUTWARNINGCOUNT PIC S9(9) BINARY.
      ** Number of Async put calls had failures
       15  MQSTS-PUTFAILURECOUNT PIC S9(9) BINARY.
      ** Failing object type
       15  MQSTS-OBJECTTYPE PIC S9(9) BINARY.
      ** Failing object name
       15  MQSTS-OBJECTNAME PIC X(48).
      ** Failing object queue manager
       15  MQSTS-OBJECTQMGRNAME PIC X(48).
      ** Resolved name of destination queue
```

```
       15  MQSTS-RESOLVEDOBJECTNAME PIC X(48).
      ** Resolved name of destination qmgr
       15  MQSTS-RESOLVEDQMGRNAME PIC X(48).
      ** Ver:1 **
      ** Failing object long name
       15  MQSTS-OBJECTSTRING.
      ** Address of variable length string
       20  MQSTS-OBJECTSTRING-VSPTR POINTER.
      ** Offset of variable length string
       20  MQSTS-OBJECTSTRING-VSOFFSET PIC S9(9) BINARY.
      ** Size of buffer
       20  MQSTS-OBJECTSTRING-VSBUFSIZE PIC S9(9) BINARY.
      ** Length of variable length string
       20  MQSTS-OBJECTSTRING-VSLENGTH PIC S9(9) BINARY.
      ** CCSID of variable length string
       20  MQSTS-OBJECTSTRING-VSCCSID PIC S9(9) BINARY.
      ** Failing subscription name
       15  MQSTS-SUBNAME.
      ** Address of variable length string
       20  MQSTS-SUBNAME-VSPTR POINTER.
      ** Offset of variable length string
       20  MQSTS-SUBNAME-VSOFFSET PIC S9(9) BINARY.
      ** Size of buffer
       20  MQSTS-SUBNAME-VSBUFSIZE PIC S9(9) BINARY.
      ** Length of variable length string
       20  MQSTS-SUBNAME-VSLENGTH PIC S9(9) BINARY.
      ** CCSID of variable length string
       20  MQSTS-SUBNAME-VSCCSID PIC S9(9) BINARY.
      ** Failing open options
       15  MQSTS-OPENOPTIONS PIC S9(9) BINARY.
      ** Failing subscription options
       15  MQSTS-SUBOPTIONS PIC S9(9) BINARY.
      ** Ver:2 **
```

Dichiarazione PL/I per MQSTS

```
dcl
   1 MQSTS based,
     3 StrucId             char(4),       /* Structure identifier */
      3 Version            fixed bin(31), /* Structure version number */
      3 CompCode           fixed bin(31), /* Completion code */
      3 Reason             fixed bin(31), /* Reason code */
      3 PutSuccessCount    fixed bin(31), /* Put success count */
      3 PutWarningCount    fixed bin(31), /* Put warning count */
      3 PutFailureCount    fixed bin(31), /* Put failure count */
      3 ObjectType         fixed bin(31), /* Object type */
      3 ObjectName         char(48), /* Object name */
      3 ObjectQmgrName     char(48), /* Object queue manager */
      3 ResolvedObjectName char(48), /* Resolved Object name */
      3 ResolvedQmgrName   char(48); /* Resolved Object queue manager */
  /* Ver:1 */
      3 ObjectString,                /* Failing object long name */
         5 VSPtr pointer,            /* Address of variable length string */
         5 VSOffset fixed bin(31),   /* Offset of variable length string */
         5 VSBufSize fixed bin(31),  /* Size of buffer */
         5 VSLength fixed bin(31),   /* Length of variable length string */
         5 VSCCSID fixed bin(31);    /* CCSID of variable length string */
      3 SubName,                     /* Failing subscription name */
         5 VSPtr pointer,            /* Address of variable length string */
         5 VSOffset fixed bin(31),   /* Offset of variable length string */
         5 VSBufSize fixed bin(31),  /* Size of buffer */
         5 VSLength fixed bin(31),   /* Length of variable length string */
         5 VSCCSID fixed bin(31);    /* CCSID of variable length string */
     3 OpenOptions fixed bin(31),    /* Failing open options */
     3 SubOptions fixed bin(31);     /* Failing subscription options */
  /* Ver:2 */
```

Dichiarazione High Level Assembler per MQSTS

```
    MQSTS                       DSECT
    MQSTS_STRUCID               DS    CL4   Structure identifier
    MQSTS_VERSION               DS    F     Structure version number
    MQSTS_COMPCODE              DS    F     Completion code
    MQSTS_REASON                DS    F     Reason code
    MQSTS_PUTSUCCESSCOUNT       DS    F     Success count
    MQSTS_PUTWARNINGCOUNT       DS    F     Warning count
    MQSTS_PUTFAILURECOUNT       DS    F     Failure count
```

```
MQSTS_OBJTYPE                 DS    F     Object type
MQSTS_OBJNAME                 DS    CL48  Object name
MQSTS_OBJQMGR                 DS    CL48  Object queue manager
MQSTS_ROBJNAME                DS    CL48  Resolved object name
MQSTS_ROBJQMGR                DS    CL48  Resolved object queue manager
MQSTS_OBJECTSTRING            DS    0F   Force fullword alignment
MQSTS_OBJECTSTRING_VSPTR      DS    A  Address of variable length string
MQSTS_OBJECTSTRING_VSOFFSET   DS    F  Offset of variable length string
MQSTS_OBJECTSTRING_VSBUFSIZE  DS    F  Size of buffer
MQSTS_OBJECTSTRING_VSLENGTH   DS    F  Length of variable length string
MQSTS_OBJECTSTRING_VSCCSID    DS    F  CCSID of variable length string
MQSTS_OBJECTSTRING_LENGTH     EQU   *-MQSTS_OBJECTSTRING
                                    ORG   MQSTS_OBJECTSTRING
MQSTS_OBJECTSTRING_AREA       DS    CL(MQSTS_OBJECTSTRING_LENGTH)
*
MQSTS_SUBNAME                 DS    0F Force fullword alignment
MQSTS_SUBNAME_VSPTR           DS    A  Address of variable length string
MQSTS_SUBNAME_VSOFFSET        DS    F  Offset of variable length string
MQSTS_SUBNAME_VSBUFSIZE       DS    F  Size of buffer
MQSTS_SUBNAME_VSLENGTH        DS    F  Length of variable length string
MQSTS_SUBNAME_VSCCSID         DS    F  CCSID of variable length string
MQSTS_SUBNAME_LENGTH          EQ    *-MQSTS_SUBNAME
                                    ORG   MQSTS_SUBNAME
MQSTS_SUBNAME_AREA            DS    CL(MQSTS_SUBNAME_LENGTH)
*
MQSTS_OPENOPTIONS             DS    F  Failing open options
MQSTS_SUBOPTIONS              DS    F  Failing subscription option
MQSTS_LENGTH                  EQU   *-MQSTS
                              ORG   MQSTS
MQSTS_AREA                    DS    CL(MQSTS_LENGTH)
```

Riferimenti correlati

“MQSTAT - Richiamo delle informazioni di stato” a pagina 803
Utilizzare la chiamata MQSTAT per richiamare le informazioni sullo stato. Il tipo di informazioni di stato restituite è determinato dal valore Tipo specificato nella chiamata.

StrucId (MQCHAR4) per MQSTS

Questo è l'identificativo della struttura della struttura di report di stato. È sempre un campo di immissione. Il valore è MQSTS_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQST
Identificatore per la struttura di report di stato.

Per il linguaggio di programmazione C, viene definita anche la costante MQSTS_STRUC_ID_ARRAY . Ha lo stesso valore di MQSTS_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQSTS

Il numero di versione della struttura.

Il valore deve essere:

MQSTS_VERSION_1
Struttura di report di stato versione 1.

MQSTS_VERSION_2
Struttura di report di stato versione 2.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQSTS_CURRENT_
Versione corrente della struttura di report di stato. La versione corrente è MQSTS_VERSION_2.

Version è sempre un campo di immissione. Il valore iniziale è MQSTS_VERSION_1.

CompCode (MQLONG) per MQSTS

Il codice di completamento dell'operazione riportata.

L'interpretazione di CompCode dipende dal valore del parametro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Questo è il codice di completamento risultante da una precedente operazione di inserimento asincrona sull'oggetto specificato in ObjectName.

RICONNESSIONE_TIPO_MQSTAT

Se la connessione è in fase di riconnessione o non è riuscita a riconnettersi, questo è il codice di completamento che ha causato l'avvio della riconnessione.

Se la connessione è attualmente connessa, il valore è MQCC_OK.

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Se la connessione non è riuscita a riconnettersi, questo è il codice di completamento che ha causato l'esito negativo della riconnessione.

Se la connessione è attualmente connessa o si riconnette, il valore è MQCC_OK.

CompCode è sempre un campo di output. Il valore iniziale è MQCC_OK.

Motivo (MQLONG) per MQSTS

Il codice motivo dell'operazione su cui si sta eseguendo la notifica.

L'interpretazione di Reason dipende dal valore del parametro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Questo è il codice di errore risultante da una precedente operazione di inserimento asincrona sull'oggetto specificato in ObjectName.

RICONNESSIONE_TIPO_MQSTAT

Se la connessione è in fase di riconnessione o non è riuscita a riconnettersi, questo è il codice di errore che ha causato l'avvio della riconnessione.

Se la connessione è attualmente connessa, il valore è MQRC_NONE.

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Se la connessione non è riuscita a riconnettersi, questo è il codice di errore che ha causato l'esito negativo della riconnessione.

Se la connessione è attualmente connessa o si riconnette, il valore è MQRC_NONE.

Reason è un campo di output. Il valore iniziale è MQRC_NONE.

Conteggio PutSuccess(MQLONG) per MQSTS

Il numero di operazioni di inserimento asincrone riuscite.

Il valore di PutSuccessCount dipende dal valore del parametro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Il numero di operazioni di inserimento asincrone nell'oggetto denominato nella struttura MQSTS completate con MQCC_OK.

RICONNESSIONE_TIPO_MQSTAT

Zero.

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Zero.

PutSuccessCount è un campo di output. Il valore iniziale è zero.

Conteggio PutWarning(MQLONG) per MQSTS

Il numero di operazioni di inserimento asincrone terminate con un'avvertenza.

Il valore di PutWarningCount dipende dal valore del parametro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Il numero di operazioni di inserimento asincrone nell'oggetto denominato nella struttura MQSTS completate con MQCC_WARNING.

RICONNESSIONE_TIPO_MQSTAT

Zero.

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Zero.

PutWarningCount è un campo di output. Il valore iniziale è zero.

Conteggio PutFailure(MQLONG) per MQSTS

Il numero di operazioni di inserimento asincrone non riuscite.

Il valore di PutFailureCount dipende dal valore del parametro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Il numero di operazioni di inserimento asincrone nell'oggetto denominato nella struttura MQSTS completate con MQCC_FAILED.

RICONNESSIONE_TIPO_MQSTAT

Zero.

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Zero.

PutFailureCount è un campo di output. Il valore iniziale è zero.

ObjectType (MQLONG) per MQSTS

Il tipo di oggetto indicato in *ObjectName* su cui viene creato il report.

I valori possibili di ObjectType sono elencati in "MQOT_ * (Tipi di oggetti e tipi di oggetti estesi)" a pagina 165.

ObjectType è un campo di output. Il valore iniziale è MQOT_Q.

ObjectName (MQCHAR48) per MQSTS

Il nome dell'oggetto su cui si sta eseguendo il report.

L'interpretazione di ObjectName dipende dal valore del parametro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Questo è il nome della coda o dell'argomento utilizzato nell'operazione di inserimento, il cui errore viene riportato nei campi *CompCode* e *Reason* della struttura MQSTS .

RICONNESSIONE_TIPO_MQSTAT

Se la connessione è in fase di riconnessione, questo è il nome del gestore code associato alla connessione.

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Se la connessione non è riuscita a riconnettersi, questo è il nome dell'oggetto che ha causato l'esito negativo della riconnessione. Il motivo dell'errore viene riportato nei campi *CompCode* e *Reason* nella struttura MQSTS .

ObjectName è un campo di output. Il suo valore iniziale è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

Nome ObjectQMgr(MQCHAR48) per MQSTS

Il nome del gestore code su cui viene eseguita la notifica.

L'interpretazione di `ObjectQMgrName` dipende dal valore del parametro `MQSTAT` **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Questo è il nome del gestore code su cui è definito l'oggetto *ObjectName* . Un nome che è completamente vuoto fino al primo carattere null o alla fine del campo indica il gestore code a cui è connessa l'applicazione (il gestore code locale).

RICONNESSIONE_TIPO_MQSTAT

Il campo **ObjectQMgrName** contiene il nome di un gestore code a cui viene richiesta la riconnessione oppure è vuoto se non viene specificato alcun gestore code. Se possibile, il client tenta di riconnettersi a un gestore code con quel nome.

Vuoto.

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Se la connessione non è riuscita a riconnettersi, questo è il nome dell'oggetto che ha causato l'esito negativo della riconnessione. Il motivo dell'errore viene riportato nei campi *CompCode* e *Reason* nella struttura `MQSTS` .

`ObjectQMgrName` è un campo di output. Il suo valore è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

Nome ResolvedObject(MQCHAR48) per MQSTS

Il nome dell'oggetto denominato in *ObjectName* dopo che il gestore code locale ha risolto il nome.

L'interpretazione di `ResolvedObjectName` dipende dal valore del parametro `MQSTAT` **Type** .

MQSTAT_TYPE_ASYNC_ERROR

`ResolvedObjectName` è il nome dell'oggetto denominato in *ObjectName* dopo che il gestore code locale ha risolto il nome. Il nome restituito è il nome di un oggetto che esiste sul gestore code identificato da *ResolvedQMgrName*.

RICONNESSIONE_TIPO_MQSTAT

Vuoto.

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Vuoto.

`ResolvedObjectName` è un campo di output. Il suo valore iniziale è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

Il nome ResolvedQMgr(MQCHAR48) per MQSTS

Il nome del gestore code di destinazione dopo che il gestore code locale ha risolto il nome.

L'interpretazione di `ResolvedQMgrName` dipende dal valore del parametro `MQSTAT` **Type** .

MQSTAT_TYPE_ASYNC_ERROR

`ResolvedQMgrName` è il nome del gestore code di destinazione dopo che il gestore code locale ha risolto il nome. Il nome restituito è il nome del gestore code proprietario dell'oggetto identificato da *ResolvedObjectName*. *ResolvedQMgrName* potrebbe essere il nome del gestore code locale.

RICONNESSIONE_TIPO_MQSTAT

Vuoto.

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Vuoto.

ResolvedQMgrName è sempre un campo di output. Il suo valore iniziale è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

ObjectString (MQCHARV) per MQSTS

Nome oggetto lungo dell'oggetto in errore su cui si sta eseguendo il report. Presente solo nella versione 2 di MQSTS o superiore.

L'interpretazione di ObjectString dipende dal valore del parametro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Questo è il nome dell'oggetto lungo della coda o dell'argomento utilizzato nell'operazione MQPUT , che non è riuscita.

RICONNESSIONE_TIPO_MQSTAT

Stringa di lunghezza zero

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Questo è il nome oggetto lungo dell'oggetto che ha causato la mancata riuscita della riconnessione.

ObjectString è un campo di output. Il valore iniziale è una stringa di lunghezza zero.

SubName (MQCHARV) per MQSTS

Il nome della sottoscrizione non riuscita. Presente solo nella versione 2 di MQSTS o superiore.

L'interpretazione di SubName dipende dal valore del parametro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Stringa di lunghezza zero.

RICONNESSIONE_TIPO_MQSTAT

Stringa di lunghezza zero.

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Il nome della sottoscrizione che ha causato la mancata riuscita della riconnessione. Se non è disponibile alcun nome di sottoscrizione o se l'errore non è correlato a una sottoscrizione, si tratta di una stringa di lunghezza zero.

SubName è un campo di output. Il valore iniziale è una stringa di lunghezza zero.

OpenOptions (MQLONG) per MQSTS

Il OpenOptions utilizzato per aprire l'oggetto su cui viene creato il report. Presente solo nella versione 2 di MQSTS o superiore.

Il valore di OpenOptions dipende dal valore del parametro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Zero.

RICONNESSIONE_TIPO_MQSTAT

Zero.

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Il OpenOptions utilizzato quando si è verificato l'errore. Il motivo dell'errore viene riportato nei campi *CompCode* e *Reason* nella struttura MQSTS .

OpenOptions è un campo di output. Il valore iniziale è zero.

SubOptions (MQLONG) per MQSTS

Il SubOptions utilizzato per aprire la sottoscrizione in errore. Presente solo nella versione 2 di MQSTS o superiore.

L'interpretazione di SubOptions dipende dal valore del parametro MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Zero.

RICONNESSIONE_TIPO_MQSTAT

Zero.

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Il SubOptions utilizzato quando si è verificato l'errore. Se l'errore non è relativo alla sottoscrizione di un argomento, il valore restituito è zero.

SubOptions è un campo di output. Il valore iniziale è zero.

MQTM - Messaggio trigger

La struttura MQTM descrive i dati nel messaggio del trigger inviato dal gestore code a un'applicazione di controllo trigger quando si verifica un evento trigger per una coda.Questa struttura fa parte di IBM MQ Trigger Monitor Interface (TMI), che è una delle interfacce del framework IBM MQ .

Nome formato

MQFMT_TRIGGER.

Serie di caratteri e codifica

I dati del carattere in MQTM si trovano nella serie di caratteri del gestore code che genera MQTM. I dati numerici in MQTM si trovano nella codifica macchina del gestore code che genera MQTM.

La serie di caratteri e la codifica di MQTM sono fornite dai campi *CodedCharSetId* e *Encoding* in:

- MQMD (se la struttura MQTM si trova all'inizio dei dati del messaggio) oppure
- La struttura dell'intestazione che precede la struttura MQTM (tutti gli altri casi).

Utilizzo

Un'applicazione trigger - monitor potrebbe dover passare alcune o tutte le informazioni nel messaggio trigger all'applicazione avviata dall'applicazione trigger - monitor. Le informazioni che potrebbero essere necessarie per l'applicazione avviata includono *QName*, *TriggerData*e *UserData*. L'applicazione di controllo dei trigger può passare la struttura MQTM direttamente all'applicazione avviata oppure passare una struttura MQTMC2 , in base a quanto consentito dall'ambiente e conveniente per l'applicazione avviata. Per informazioni su MQTMC2, consultare “MQTMC2 - Messaggio di attivazione 2 (formato carattere)” a pagina 622.

- z/OS Su z/OS, per un'applicazione MQAT_CICS avviata utilizzando la transazione CKTI, l'intera struttura del messaggio trigger MQTM viene resa disponibile alla transazione avviata; le informazioni possono essere richiamate utilizzando il comando EXEC CICS RETRIEVE.
- IBM i Su IBM i, l'applicazione di controllo dei trigger fornita con IBM MQ passa una struttura MQTMC2 all'applicazione avviata.

Per informazioni sull'utilizzo dei trigger, consultare Avvio delle applicazioni IBM MQ mediante i trigger.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 533. Campi in MQTM per MQTM

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQTM	'TM¬¬'
Versione (numero versione struttura)	MQTM_VERSION_1	1
QName (nome della coda attivata)	Nessuna	Stringa null o spazi vuoti
ProcessName (nome dell'oggetto processo)	Nessuna	Stringa null o spazi vuoti
TriggerData (dati trigger)	Nessuna	Stringa null o spazi vuoti
ApplType (tipo di applicazione	Nessuna	0
ApplId (identificativo applicazione)	Nessuna	Stringa null o spazi vuoti
EnvData (dati di ambiente)	Nessuna	Stringa null o spazi vuoti
UserData (dati utente)	Nessuna	Stringa null o spazi vuoti

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.
3. Nel linguaggio di programmazione C, la variabile macroMQTM_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQTM MyTM = {MQTM_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQTM

```
typedef struct tagMQTM MQTM;
struct tagMQTM {
  MQCHAR4    StrucId;      /* Structure identifier */
  MQLONG     Version;      /* Structure version number */
  MQCHAR48   QName;        /* Name of triggered queue */
  MQCHAR48   ProcessName;  /* Name of process object */
  MQCHAR64   TriggerData;  /* Trigger data */
  MQLONG     ApplType;     /* Application type */
  MQCHAR256  ApplId;       /* Application identifier */
  MQCHAR128  EnvData;      /* Environment data */
  MQCHAR128  UserData;     /* User data */
};
```

Dichiarazione COBOL per MQTM

```
**   MQTM structure
  10 MQTM.
**    Structure identifier
   15 MQTM-STRUCID     PIC X(4).
**    Structure version number
   15 MQTM-VERSION     PIC S9(9) BINARY.
```

```
**    Name of triggered queue
   15 MQTM-QNAME       PIC X(48).
**    Name of process object
   15 MQTM-PROCESSNAME PIC X(48).
**    Trigger data
   15 MQTM-TRIGGERDATA PIC X(64).
**    Application type
   15 MQTM-APPLTYPE    PIC S9(9) BINARY.
**    Application identifier
   15 MQTM-APPLID      PIC X(256).
**    Environment data
   15 MQTM-ENVDATA     PIC X(128).
**    User data
   15 MQTM-USERDATA    PIC X(128).
```

Dichiarazione PL/I per MQTM

```
dcl
 1 MQTM based,
   3 StrucId     char(4),       /* Structure identifier */
   3 Version     fixed bin(31), /* Structure version number */
   3 QName       char(48),      /* Name of triggered queue */
   3 ProcessName char(48),      /* Name of process object */
   3 TriggerData char(64),      /* Trigger data */
   3 ApplType    fixed bin(31), /* Application type */
   3 ApplId      char(256),     /* Application identifier */
   3 EnvData     char(128),     /* Environment data */
   3 UserData    char(128);     /* User data */
```

Dichiarazione High Level Assembler per MQTM

```
MQTM              DSECT
MQTM_STRUCID      DS   CL4    Structure identifier
MQTM_VERSION      DS   F      Structure version number
MQTM_QNAME        DS   CL48   Name of triggered queue
MQTM_PROCESSNAME  DS   CL48   Name of process object
MQTM_TRIGGERDATA  DS   CL64   Trigger data
MQTM_APPLTYPE     DS   F      Application type
MQTM_APPLID       DS   CL256  Application identifier
MQTM_ENVDATA      DS   CL128  Environment data
MQTM_USERDATA     DS   CL128  User data
*
MQTM_LENGTH       EQU  *-MQTM
                  ORG  MQTM
MQTM_AREA         DS   CL(MQTM_LENGTH)
```

Dichiarazione Visual Basic per MQTM

```
Type MQTM
  StrucId     As String*4   'Structure identifier'
  Version     As Long       'Structure version number'
  QName       As String*48  'Name of triggered queue'
  ProcessName As String*48  'Name of process object'
  TriggerData As String*64  'Trigger data'
  ApplType    As Long       'Application type'
  ApplId      As String*256 'Application identifier'
  EnvData     As String*128 'Environment data'
  UserData    As String*128 'User data'
End Type
```

MQMD per un messaggio trigger

Tabella 534. Impostazioni per i campi in MQMD di un messaggio trigger generato dal gestore code

Campo in MQMD	Valore utilizzato
StrucId	ID_STRUC_MQMD
Version	MQMD_VERSION_1
Report	MQRO_NONE

Tabella 534. Impostazioni per i campi in MQMD di un messaggio trigger generato dal gestore code (Continua)

Campo in MQMD	Valore utilizzato
MsgType	MQMT_DATAGRAM
Expiry	MQEI_UNLIMITED
Feedback	MQFB_NONE
Encoding	MQEN_NATIVE
CodedCharSetId	Attributo **CodedCharSetId** del Gestore code
Format	MQFMT_TRIGGER
Priority	Attributo **DefPriority** della coda di avvio
Persistence	MQPER_NOT_PERSISTENT
MsgId	Un valore univoco
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	Spazi
ReplyToQMgr	Nome del gestore code
UserIdentifier	Spazi
AccountingToken	MQACT_NONE
ApplIdentityData	Spazi
PutApplType	MQAT_QMGR o come appropriato per l'agent del canale dei messaggi
PutApplName	I primi 28 byte del nome del gestore code
PutDate	Data in cui viene inviato il messaggio trigger
PutTime	Ora in cui viene inviato il messaggio trigger
ApplOriginData	Spazi

Un'applicazione che genera un messaggio trigger è consigliata per impostare valori simili, ad eccezione dei seguenti:

- Il campo *Priority* può essere impostato su MQPRI_PRIORITY_AS_Q_DEF (il gestore code modificherà la priorità predefinita per la coda di iniziazione quando il messaggio viene inserito).
- Il campo *ReplyToQMgr* può essere impostato su spazi vuoti (il gestore code lo modificherà con il nome del gestore code locale quando viene inserito il messaggio).
- Impostare i campi di contesto come appropriato per l'applicazione.

StrucId (MQCHAR4) per MQTM

Questo è l'identificativo della struttura del messaggio trigger. È sempre un campo di immissione. Il valore è MQTM_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQTM
Identificativo per la struttura del messaggio trigger.

Per il linguaggio di programmazione C, viene definita anche la costante MQTM_STRUC_ID_ARRAY. Ha lo stesso valore di MQTM_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQTM

Questo è il numero di versione della struttura. Il valore deve essere:

MQTM_VERSION_1
Numero di versione per la struttura del messaggio trigger.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQTM_CURRENT_
La versione corrente della struttura del messaggio trigger.

Il valore iniziale di questo campo è MQTM_VERSION_1.

QName (MQCHAR48) per MQTM

Questo è il nome della coda per cui si è verificato un evento trigger e viene utilizzato dall'applicazione avviata dall'applicazione trigger - monitor. Il gestore code inizializza questo campo con il valore dell'attributo **QName** della coda attivata; consultare“Attributi per le code” a pagina 858per i dettagli di questo attributo.

I nomi che sono più brevi della lunghezza definita del campo vengono riempiti a destra con spazi; non vengono terminati prematuramente con un carattere null.

La lunghezza di questo campo è fornita da MQ_Q_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

ProcessName (MQCHAR48) per MQTM

Questo è il nome dell'oggetto processo del gestore code specificato per la coda attivata e può essere utilizzato dall'applicazione di controllo trigger che riceve il messaggio del trigger. Il gestore code inizializza questo campo con il valore dell'attributo **ProcessName** della coda identificata dal campo *QName* ; consultare“Attributi per le code” a pagina 858per i dettagli di questo attributo.

I nomi più brevi della lunghezza definita del campo vengono sempre riempiti a destra con spazi vuoti; non vengono terminati prematuramente con un carattere null.

La lunghezza di questo campo viene fornita da MQ_PROCESS_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

TriggerData (MQCHAR64) per MQTM

Si tratta di dati in formato libero per l'utilizzo da parte dell'applicazione di controllo trigger che riceve il messaggio trigger. Il gestore code inizializza questo campo con il valore dell'attributo **TriggerData** della coda identificata dal campo *QName* ; consultare “Attributi per le code” a pagina 858 per i dettagli di questo attributo. Il contenuto di questi dati non è significativo per il gestore code.

Su z/OS, per un'applicazione CICS avviata utilizzando la transazione CKTI, queste informazioni non vengono utilizzate.

La lunghezza di questo campo è fornita da MQ_TRIGGER_DATA_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 64 caratteri vuoti in altri linguaggi di programmazione.

ApplType (MQLONG) per MQTM

Ciò identifica la natura del programma da avviare e viene utilizzato dall'applicazione di controllo trigger che riceve il messaggio trigger. Il gestore code inizializza questo campo con il valore dell'attributo **ApplType** dell'oggetto processo identificato dal campo *ProcessName* ; consultare “Attributi per le definizioni di processi” a pagina 895 per i dettagli di questo attributo. Il contenuto di questi dati non è significativo per il gestore code.

ApplType può avere uno dei seguenti valori standard. È possibile utilizzare anche i tipi definiti dall'utente, ma devono essere limitati ai valori compresi nell'intervallo tra MQAT_USER_FIRST e MQAT_USER_LAST:

AIX MQAT
Applicazione AIX (stesso valore di MQAT_UNIX).

MQAT_BATCH
Applicazione batch

MEDIA_MQAT_BROKER
Applicazione broker

MQAT_CICS
Transazione CICS .

BRIDGE - MQAT_CICS_BRIDGE
Applicazione CICS bridge .

CICS_VSE MQAT
Transazione CICS/VSE .

DOS MQAT
Applicazione IBM MQ MQI client su PC DOS.

IMS MQAT
Applicazione IMS .

MQAT_IM_Bridge
Applicazione bridge IMS .

JAVA MQAT
Applicazione Java .

MVS MQAT
Applicazione MVS o TSO (stesso valore di MQAT_ZOS).

MQAT_NOTES_AGENT
Lotus Notes Applicazione agent.

MQAT_OS390
Applicazione OS/390 (stesso valore di MQAT_ZOS).

MQAT_OS400
Applicazione IBM i .

BATCH_RRS_MQAT
Applicazione batch RRS.

UNIX MQAT
Applicazione UNIX .

MQAT_SCONOSCIUTO
Applicazione di tipo sconosciuto.

UTENTE MQAT
Tipo di applicazione definito dall'utente.

VOS MQAT
Applicazione VOS Stratus.

WINDOWS MQAT
Applicazione Windows a 16 bit.

MQAT_WINDOWS_NT
Applicazione Windows a 32 bit.

WLM MQAT
Applicazione z/OS workload manager.

XCF MQAT
XCF.

ZOS MQAT
Applicazione z/OS .

MQAT_USER_FIRST
Il valore più basso per il tipo di applicazione definito dall'utente.

MQAT_USER_LAST
Il valore più alto per il tipo di applicazione definito dall'utente.

Il valore iniziale di questo campo è 0.

ApplId (MQCHAR256) per MQTM

Si tratta di una stringa di caratteri che identifica l'applicazione da avviare e viene utilizzata dall'applicazione di controllo trigger che riceve il messaggio di trigger. Il gestore code inizializza questo campo con il valore dell'attributo **ApplId** dell'oggetto processo identificato dal campo *ProcessName* ; consultare “Attributi per le definizioni di processi” a pagina 895 per i dettagli di questo attributo. Il contenuto di questi dati non è significativo per il gestore code.

Il significato di *ApplId* è determinato dall'applicazione trigger - monitor. Il controllo dei trigger fornito da IBM MQ richiede *ApplId* come nome di un programma eseguibile. Le seguenti note si applicano agli ambienti indicati:

- Su z/OS, *ApplId* è:
  - Un identificativo di transazione CICS , per le applicazioni avviate utilizzando la transazione CKTI di controllo trigger CICS
  - Un identificativo della transazione IMS , per le applicazioni avviate utilizzando il controllo trigger IMS CSQQTRMN
- Su sistemi Windows , il nome del programma può essere preceduto da un percorso di unità e di directory.
- Su IBM i, il nome del programma può essere preceduto da un nome libreria e / o carattere.
- Su AIX and Linux, il nome programma può essere preceduto da un percorso di directory.

La lunghezza di questo campo viene fornita da MQ_PROCESS_APPL_ID_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 256 caratteri vuoti in altri linguaggi di programmazione.

EnvData (MQCHAR128) per MQTM

Si tratta di una stringa di caratteri che contiene informazioni relative all'ambiente relative all'applicazione da avviare e viene utilizzata dall'applicazione di controllo trigger che riceve il messaggio del trigger. Il gestore code inizializza questo campo con il valore dell'attributo **EnvData** dell'oggetto processo identificato dal campo *ProcessName* ; consultare “Attributi per le definizioni di processi” a pagina 895 per i dettagli di questo attributo. Il contenuto di questi dati non è significativo per il gestore code.

In z/OS, per un'applicazione CICS avviata utilizzando la transazioni CKTI o un'applicazione IMS da avviare utilizzando la transazione CSQQTRMN, queste informazioni non vengono utilizzate.

La lunghezza di questo campo è fornita da MQ_PROCESS_ENV_DATA_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 128 caratteri vuoti in altri linguaggi di programmazione.

UserData (MQCHAR128) per MQTM

Si tratta di una stringa di carattere che contiene le informazioni utente relative all'applicazione da avviare e viene utilizzata dall'applicazione di controllo trigger che riceve il messaggio del trigger. Il gestore code inizializza questo campo con il valore dell'attributo **UserData** dell'oggetto processo identificato dal campo *ProcessName* ; consultare “Attributi per le definizioni di processi” a pagina 895 per i dettagli di questo attributo. Il contenuto di questi dati non è significativo per il gestore code.

Per Microsoft Windows, la stringa di caratteri non deve contenere virgolette doppie se la definizione del processo sta per essere inoltrata a **runmqtrm**.

La lunghezza di questo campo è fornita da MQ_PROCESS_USER_DATA_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 128 caratteri vuoti in altri linguaggi di programmazione.

MQTMC2 - Messaggio di attivazione 2 (formato carattere)

Quando un'applicazione di controllo trigger richiama un messaggio trigger (MQTM) da una coda di iniziazione, il controllo trigger potrebbe dover trasmettere alcune o tutte le informazioni nel messaggio trigger all'applicazione avviata dal controllo trigger.

Le informazioni di cui l'applicazione avviata potrebbe aver bisogno includono *QName*, *TriggerData* e *UserData*. L'applicazione di controllo dei trigger può passare la struttura MQTM direttamente all'applicazione avviata oppure passare una struttura MQTMC2 , in base a quanto consentito dall'ambiente e conveniente per l'applicazione avviata.

Questa struttura fa parte di IBM MQ Trigger Monitor Interface (TMI), che è una delle interfacce del framework IBM MQ .

Serie di caratteri e codifica

I dati carattere in MQTMC2 si trovano nella serie di caratteri del gestore code locale; ciò viene fornito dall'attributo del gestore code **CodedCharSetId** .

Utilizzo

La struttura MQTMC2 è molto simile al formato della struttura MQTM. La differenza è che i campi non di caratteri in MQTM vengono modificati in MQTMC2 in campi di caratteri della stessa lunghezza e il nome del gestore code viene aggiunto alla fine della struttura.

- z/OS Su z/OS, per un'applicazione MQAT_IMS avviata utilizzando l'applicazione CSQQTRMN, viene resa disponibile all'applicazione avviata una struttura MQTMC2 .
- IBM i Su IBM i, l'applicazione di controllo dei trigger fornita con IBM MQ passa una struttura MQTMC2 all'applicazione avviata.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 535. Campi in MQTMC2

Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
StrucId (identificativo della struttura)	ID_STRUC_MQTMC	'TMC¬'
Versione (numero versione struttura)	MQTMC_VERSION_2	'¬¬¬2'
QName (nome della coda attivata)	Nessuna	Stringa null o spazi vuoti
ProcessName (nome dell'oggetto processo)	Nessuna	Stringa null o spazi vuoti
TriggerData (dati trigger)	Nessuna	Stringa null o spazi vuoti
ApplType (tipo di applicazione)	Nessuna	Spazi
ApplId (identificativo applicazione)	Nessuna	Stringa null o spazi vuoti
EnvData (dati di ambiente)	Nessuna	Stringa null o spazi vuoti
UserData (dati utente)	Nessuna	Stringa null o spazi vuoti
QMgrName (nome gestore code)	Nessuna	Stringa null o spazi vuoti

Tabella 535. Campi in MQTMC2 (Continua)

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.
3. Nel linguaggio di programmazione C, la variabile macroMQTMC2_DEFAULT contiene i seguenti valori. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQTMC2 MyTMC = {MQTMC2_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQTMC2

```
typedef struct tagMQTMC2 MQTMC2;
struct tagMQTMC2 {
  MQCHAR4    StrucId;      /* Structure identifier */
  MQCHAR4    Version;      /* Structure version number */
  MQCHAR48   QName;        /* Name of triggered queue */
  MQCHAR48   ProcessName;  /* Name of process object */
  MQCHAR64   TriggerData;  /* Trigger data */
  MQCHAR4    ApplType;     /* Application type */
  MQCHAR256  ApplId;       /* Application identifier */
  MQCHAR128  EnvData;      /* Environment data */
  MQCHAR128  UserData;     /* User data */
  MQCHAR48   QMgrName;     /* Queue manager name */
};
```

Dichiarazione COBOL per MQTMC2

```
**   MQTMC2 structure
  10 MQTMC2.
**    Structure identifier
   15 MQTMC2-STRUCID     PIC X(4).
**    Structure version number
   15 MQTMC2-VERSION     PIC X(4).
**    Name of triggered queue
   15 MQTMC2-QNAME       PIC X(48).
**    Name of process object
   15 MQTMC2-PROCESSNAME PIC X(48).
**    Trigger data
   15 MQTMC2-TRIGGERDATA PIC X(64).
**    Application type
   15 MQTMC2-APPLTYPE    PIC X(4).
**    Application identifier
   15 MQTMC2-APPLID      PIC X(256).
**    Environment data
   15 MQTMC2-ENVDATA     PIC X(128).
**    User data
   15 MQTMC2-USERDATA    PIC X(128).
**    Queue manager name
   15 MQTMC2-QMGRNAME    PIC X(48).
```

Dichiarazione PL/I per MQTMC2

```
dcl
 1 MQTMC2 based,
  3 StrucId     char(4),   /* Structure identifier */
  3 Version     char(4),   /* Structure version number */
  3 QName       char(48),  /* Name of triggered queue */
```

```
  3 ProcessName char(48),  /* Name of process object */
  3 TriggerData char(64),  /* Trigger data */
  3 ApplType    char(4),   /* Application type */
  3 ApplId      char(256), /* Application identifier */
  3 EnvData     char(128), /* Environment data */
  3 UserData    char(128), /* User data */
  3 QMgrName    char(48);  /* Queue manager name */
```

Dichiarazione High Level Assembler per MQTMC2

```
MQTMC2              DSECT
MQTMC2_STRUCID      DS   CL4    Structure identifier
MQTMC2_VERSION      DS   CL4    Structure version number
MQTMC2_QNAME        DS   CL48   Name of triggered queue
MQTMC2_PROCESSNAME  DS   CL48   Name of process object
MQTMC2_TRIGGERDATA  DS   CL64   Trigger data
MQTMC2_APPLTYPE     DS   CL4    Application type
MQTMC2_APPLID       DS   CL256  Application identifier
MQTMC2_ENVDATA      DS   CL128  Environment data
MQTMC2_USERDATA     DS   CL128  User data
MQTMC2_QMGRNAME     DS   CL48   Queue manager name
*
MQTMC2_LENGTH       EQU  *-MQTMC2
                    ORG  MQTMC2
MQTMC2_AREA         DS   CL(MQTMC2_LENGTH)
```

Dichiarazione Visual Basic per MQTMC2

```
Type MQTMC2
  StrucId     As String*4   'Structure identifier'
  Version     As String*4   'Structure version number'
  QName       As String*48  'Name of triggered queue'
  ProcessName As String*48  'Name of process object'
  TriggerData As String*64  'Trigger data'
  ApplType    As String*4   'Application type'
  ApplId      As String*256 'Application identifier'
  EnvData     As String*128 'Environment data'
  UserData    As String*128 'User data'
  QMgrName    As String*48  'Queue manager name'
End Type
```

StrucId (MQCHAR4) per MQTMC2

Questo è l'identificativo della struttura del messaggio trigger 2 (formato carattere). È sempre un campo di immissione. Il valore è MQTMC2_STRUC_ID.

Il valore deve essere:

MQTMC2_STRUC_ID
Identificativo per la struttura del messaggio trigger (formato carattere).

Per il linguaggio di programmazione C, viene definita anche la costante MQTMC2_STRUC_ID_ARRAY . Ha lo stesso valore di MQTMC2_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQCHAR4) per MQTMC2

Numero di versione della struttura.

Il valore deve essere:

MQTMC_VERSION_2
La versione 2 attiva la struttura del messaggio (formato carattere).

Per il linguaggio di programmazione C, viene definita anche la costante MQTMC_VERSION_2_ARRAY ; ha lo stesso valore di MQTMC_VERSION_2, ma è un array di caratteri invece di una stringa.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQTMC_CURRENT_
La versione corrente della struttura del messaggio trigger (formato carattere).

QName (MQCHAR48) per MQTMC2
Nome della coda attivata.

Consultare il campo *QName* nella struttura MQTM.

ProcessName (MQCHAR48) per MQTMC2
Nome dell'oggetto processo.

Consultare il campo *ProcessName* nella struttura MQTM.

TriggerData (MQCHAR64) per MQTMC2
I dati del trigger.

Consultare il campo *TriggerData* nella struttura MQTM.

ApplType (MQCHAR4) per MQTMC2
Il tipo di applicazione.

Questo campo contiene sempre spazi vuoti, qualunque sia il valore nel campo *ApplType* nella struttura MQTM del messaggio trigger originale.

ApplId (MQCHAR256) per MQTMC2

Identificativo applicazione.

Consultare il campo *ApplId* nella struttura MQTM.

EnvData (MQCHAR128) per MQTMC2
Dati di ambiente.

Consultare il campo *EnvData* nella struttura MQTM.

UserData (MQCHAR128) per MQTMC2
Dati utente.

Consultare il campo *UserData* nella struttura MQTM.

QMgrName (MQCHAR48) per MQTMC2
È il nome del gestore code.

Questo è il nome del gestore code in cui si è verificato l'evento trigger.

MQWIH - Intestazione informazioni di lavoro

Se un messaggio deve essere elaborato da WLM (workload manager) z/OS , il messaggio deve iniziare con una struttura MQWIH. Questa struttura descrive le informazioni che devono trovarsi all'inizio di un messaggio che deve essere gestito da WLM.

Disponibilità

Tutti i sistemi IBM MQ , più i client IBM MQ connessi a questi sistemi.

Nome formato

MQFMT_WORK_INFO_HEADER.

Serie di caratteri e codifica

I campi nella struttura MQWIH si trovano nella serie di caratteri e nella codifica forniti dai campi *CodedCharSetId* e *Encoding* nella struttura dell'intestazione che precede MQWIH o da quei campi nella struttura MQMD se MQWIH si trova all'inizio dei dati del messaggio dell'applicazione.

La serie di caratteri deve essere una serie di caratteri a byte singolo per i caratteri validi nei nomi coda.

Utilizzo

Per qualsiasi piattaforma supportata da IBM MQ è possibile creare e trasmettere un messaggio che include la struttura MQWIH, ma solo un gestore code IBM MQ for z/OS può interagire con WLM. Pertanto, per consentire al messaggio di raggiungere WLM da un gestore code nonz/OS , la rete del gestore code deve includere almeno un gestore code z/OS attraverso il quale il messaggio può essere instradato.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 536. Campi in MQWIH

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQWIH	'WIH¬'
Versione (numero versione struttura)	MQWIH_VERSION_1	1
StrucLength (lunghezza della struttura MQWIH)	MQWIH_LENGTH_1	120
Codifica (codifica numerica dei dati che seguono MQWIH)	Nessuna	0
CodedCharSetId (identificativo della serie di caratteri dei dati che seguono MQWIH)	MQCCSI_UNDEFINED	0
Formato (nome formato dei dati che seguono MQWIH)	MQFMT_NONE	Spazi
Indicatori (indicatori)	MQWIH_NONE	0
ServiceName (nome servizio)	Nessuna	Spazi
ServiceStep (nome passo del servizio)	Nessuna	Spazi
MsgToken (token messaggio)	MQMTOK_NONE	Valori null
Riservato (Riservato)	Nessuna	Spazi

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macroMQWIH_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQWIH MyWIH = {MQWIH_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQWIH

```
typedef struct tagMQWIH MQWIH;
struct tagMQWIH {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    StrucLength;     /* Length of MQWIH structure */
  MQLONG    Encoding;        /* Numeric encoding of data that follows
                                MQWIH */
  MQLONG    CodedCharSetId;  /* Character-set identifier of data that
                                follows MQWIH */
  MQCHAR8   Format;          /* Format name of data that follows
                                MQWIH */
  MQLONG    Flags;           /* Flags */
  MQCHAR32  ServiceName;     /* Service name */
  MQCHAR8   ServiceStep;     /* Service step name */
  MQBYTE16  MsgToken;        /* Message token */
  MQCHAR32  Reserved;        /* Reserved */
};
```

Dichiarazione COBOL per MQWIH

```
**   MQWIH structure
  10 MQWIH.
**    Structure identifier
   15 MQWIH-STRUCID        PIC X(4).
**    Structure version number
   15 MQWIH-VERSION        PIC S9(9) BINARY.
**    Length of MQWIH structure
   15 MQWIH-STRUCLENGTH    PIC S9(9) BINARY.
**    Numeric encoding of data that follows MQWIH
   15 MQWIH-ENCODING       PIC S9(9) BINARY.
**    Character-set identifier of data that follows MQWIH
   15 MQWIH-CODEDCHARSETID PIC S9(9) BINARY.
**    Format name of data that follows MQWIH
   15 MQWIH-FORMAT         PIC X(8).
**    Flags
   15 MQWIH-FLAGS          PIC S9(9) BINARY.
**    Service name
   15 MQWIH-SERVICENAME    PIC X(32).
**    Service step name
   15 MQWIH-SERVICESTEP    PIC X(8).
**    Message token
   15 MQWIH-MSGTOKEN       PIC X(16).
**    Reserved
   15 MQWIH-RESERVED       PIC X(32).
```

Dichiarazione PL/I per MQWIH

```
dcl
 1 MQWIH based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 StrucLength    fixed bin(31), /* Length of MQWIH structure */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
                                     follows MQWIH */
  3 CodedCharSetId fixed bin(31), /* Character-set identifier of data
                                     that follows MQWIH */
  3 Format         char(8),       /* Format name of data that follows
                                     MQWIH */
  3 Flags          fixed bin(31), /* Flags */
  3 ServiceName    char(32),      /* Service name */
  3 ServiceStep    char(8),       /* Service step name */
  3 MsgToken       char(16),      /* Message token */
  3 Reserved       char(32);      /* Reserved */
```

Dichiarazione High Level Assembler per MQWIH

```
MQWIH                DSECT
MQWIH_STRUCID        DS   CL4   Structure identifier
MQWIH_VERSION        DS   F     Structure version number
```

```
MQWIH_STRUCLENGTH     DS   F     Length of MQWIH structure
MQWIH_ENCODING        DS   F     Numeric encoding of data that follows
*                                MQWIH
MQWIH_CODEDCHARSETID  DS   F     Character-set identifier of data that
*                                follows MQWIH
MQWIH_FORMAT          DS   CL8   Format name of data that follows MQWIH
MQWIH_FLAGS           DS   F     Flags
MQWIH_SERVICENAME     DS   CL32  Service name
MQWIH_SERVICESTEP     DS   CL8   Service step name
MQWIH_MSGTOKEN        DS   XL16  Message token
MQWIH_RESERVED        DS   CL32  Reserved
*
MQWIH_LENGTH          EQU  *-MQWIH
                      ORG  MQWIH
MQWIH_AREA            DS   CL(MQWIH_LENGTH)
```

Dichiarazione Visual Basic per MQWIH

```
Type MQWIH
  StrucId        As String*4  'Structure identifier'
  Version        As Long      'Structure version number'
  StrucLength    As Long      'Length of MQWIH structure'
  Encoding       As Long      'Numeric encoding of data that follows'
                              'MQWIH'
  CodedCharSetId As Long      'Character-set identifier of data that'
                              'follows MQWIH'
  Format         As String*8  'Format name of data that follows MQWIH'
  Flags          As Long      'Flags'
  ServiceName    As String*32 'Service name'
  ServiceStep    As String*8  'Service step name'
  MsgToken       As MQBYTE16  'Message token'
  Reserved       As String*32 'Reserved'
End Type
```

StrucId (MQCHAR4) per MQWIH

Questo è l'identificatore della struttura dell'intestazione delle informazioni di lavoro. È sempre un campo di immissione. Il suo valore è MQWIH_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQWIH
Identificatore per la struttura di intestazione delle informazioni di lavoro.

Per il linguaggio di programmazione C, viene anche definita la costante MQWIH_STRUC_ID_ARRAY. Ha lo stesso valore di MQWIH_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQWIH

Questo è il numero di versione della struttura. Il valore deve essere:

MQWIH_VERSION_1
Struttura dell'intestazione delle informazioni di lavoro Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQWIH_CURRENT_
La versione corrente della struttura dell'intestazione delle informazioni di lavoro.

Il valore iniziale di questo campo è MQWIH_VERSION_1.

StrucLength (MQLONG) per MQWIH

Questa è la lunghezza della struttura MQWIH. Il valore deve essere:

MQWIH_LENGTH_1
Lunghezza della struttura dell'intestazione delle informazioni di lavoro version-1 .

La seguente costante specifica la lunghezza della versione corrente:

MQWIH_CURRENT_LENGTH
Lunghezza della versione corrente della struttura dell'intestazione delle informazioni di lavoro.

Il valore iniziale di questo campo è MQWIH_LENGTH_1.

Codifica (MQLONG) per MQWIH

Specifica la codifica numerica dei dati che seguono la struttura MQWIH; non si applica ai dati numerici nella struttura MQWIH stessa.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati.

Il valore iniziale di questo campo è 0.

CodedCharSetId (MQLONG) per MQWIH

Specifica l'identificatore della serie di caratteri dei dati che seguono la struttura MQWIH; non si applica ai dati di caratteri nella stessa struttura MQWIH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. È possibile utilizzare il seguente valore speciale:

MQCCSI_INHERIT
I dati carattere nei dati *che seguono* questa struttura si trovano nella stessa serie di caratteri di questa struttura.

Il gestore code modifica questo valore nella struttura inviata nel messaggio nell'effettivo identificativo della serie di caratteri della struttura. Se non si verifica alcun errore, il valore MQCCSI_INHERIT non viene restituito dalla chiamata MQGET.

MQCCSI_INHERIT non può essere utilizzato se il valore del campo *PutApplType* in MQMD è MQAT_BROKER.

Il valore iniziale di questo campo è MQCCSI_UNDEFINED.

Formato (MQCHAR8) per MQWIH

Specifica il formato dei dati che seguono la struttura MQWIH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono le stesse del campo *Format* in MQMD.

La lunghezza di questo campo è fornita da MQ_FORMAT_LENGTH. Il valore iniziale di questo campo è MQFMT_NONE.

Indicatori (MQLONG) per MQWIH

Il valore deve essere:

MQWIH_NONE
Nessun indicatore.

Il valore iniziale di questo campo è MQWIH_NONE.

ServiceName (MQCHAR32) per MQWIH

È il nome del servizio che deve elaborare il messaggio.

La lunghezza di questo campo è fornita da MQ_SERVICE_NAME_LENGTH. Il valore iniziale di questo campo è di 32 caratteri vuoti.

ServiceStep (MQCHAR8) per MQWIH

Questo è il nome del passo di *ServiceName* a cui è correlato il messaggio.

La lunghezza di questo campo è fornita da MQ_SERVICE_STEP_LENGTH. Il valore iniziale di questo campo è di 8 caratteri vuoti.

MsgToken (MQBYTE16) per MQWIH

Questo è un token di messaggio che identifica in modo univoco il messaggio.

Per le chiamate MQPUT e MQPUT1 , questo campo viene ignorato. La lunghezza di questo campo viene fornita da MQ_MSG_TOKEN_LENGTH. Il valore iniziale di questo campo è MQMTOK_NONE.

Riservato (MQCHAR32) per MQWIH

Questo è un campo riservato; deve essere vuoto.

MQXP - Blocco parametri di uscita

La struttura MQXP viene utilizzata come parametro di input / output per l'uscita incrociata API.Per ulteriori informazioni su questa uscita, vedi L'uscita di attraversamento API.

Serie di caratteri e codifica

I dati carattere in MQXP si trovano nella serie di caratteri del gestore code locale; ciò viene fornito dall'attributo del gestore code **CodedCharSetId** . I dati numerici in MQXP sono nella codifica della macchina nativa; ciò è fornito da MQENC_NATIVE.

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 537. Campi in MQXP

Nome e descrizione del campo	Nome della costante
StrucId (identificativo della struttura)	ID_STRUC_MQXP
Versione (numero versione struttura)	MQXP_VERSION_1
ExitId (identificativo uscita)	MQXT_API_CROSSING_EXIT
ExitReason (motivo del richiamo dell'uscita)	Nessuna
ExitResponse (risposta dall'uscita)	Nessuna
ExitCommand (codice chiamata API)	Nessuna
ExitParmCount (conteggio parametri)	Nessuna
Riservato (Riservato)	Nessuna
Area ExitUser (area utente)	Nessuna

Dichiarazioni di lingua

Dichiarazione C per MQXP

```
typedef struct tagMQXP MQXP;
struct tagMQXP {
  MQCHAR4   StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQLONG    ExitId;         /* Exit identifier */
  MQLONG    ExitReason;     /* Reason for invocation of exit */
  MQLONG    ExitResponse;   /* Response from exit */
  MQLONG    ExitCommand;    /* API call code */
  MQLONG    ExitParmCount;  /* Parameter count */
  MQLONG    Reserved;       /* Reserved */
  MQBYTE16  ExitUserArea;   /* User area */
};
```

Dichiarazione COBOL per MQXP

```
**   MQXP structure
  10 MQXP.
**    Structure identifier
   15 MQXP-STRUCID       PIC X(4).
**    Structure version number
   15 MQXP-VERSION       PIC S9(9) BINARY.
**    Exit identifier
   15 MQXP-EXITID        PIC S9(9) BINARY.
**    Reason for invocation of exit
   15 MQXP-EXITREASON    PIC S9(9) BINARY.
**    Response from exit
   15 MQXP-EXITRESPONSE  PIC S9(9) BINARY.
**    API call code
   15 MQXP-EXITCOMMAND   PIC S9(9) BINARY.
**    Parameter count
   15 MQXP-EXITPARMCOUNT PIC S9(9) BINARY.
**    Reserved
   15 MQXP-RESERVED      PIC S9(9) BINARY.
**    User area
   15 MQXP-EXITUSERAREA  PIC X(16).
```

Dichiarazione PL/I per MQXP

```
dcl
 1 MQXP based,
  3 StrucId       char(4),       /* Structure identifier */
  3 Version       fixed bin(31), /* Structure version number */
  3 ExitId        fixed bin(31), /* Exit identifier */
  3 ExitReason    fixed bin(31), /* Reason for invocation of exit */
  3 ExitResponse  fixed bin(31), /* Response from exit */
  3 ExitCommand   fixed bin(31), /* API call code */
  3 ExitParmCount fixed bin(31), /* Parameter count */
  3 Reserved      fixed bin(31), /* Reserved */
  3 ExitUserArea  char(16);      /* User area */
```

Dichiarazione High Level Assembler per MQXP

```
MQXP                DSECT
MQXP_STRUCID        DS   CL4   Structure identifier
MQXP_VERSION        DS   F     Structure version number
MQXP_EXITID         DS   F     Exit identifier
MQXP_EXITREASON     DS   F     Reason for invocation of exit
MQXP_EXITRESPONSE   DS   F     Response from exit
MQXP_EXITCOMMAND    DS   F     API call code
MQXP_EXITPARMCOUNT  DS   F     Parameter count
MQXP_RESERVED       DS   F     Reserved
MQXP_EXITUSERAREA   DS   XL16  User area
*
MQXP_LENGTH         EQU  *-MQXP
                    ORG  MQXP
MQXP_AREA           DS   CL(MQXP_LENGTH)
```

StrucId (MQCHAR4) per MQXP

Questo è l'identificativo della struttura del parametro di uscita. È sempre un campo di immissione. Il suo valore è MQXP_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQXP
Identificativo per la struttura del parametro di uscita.

Per il linguaggio di programmazione C, viene definita anche la costante MQXP_STRUC_ID_ARRAY. Ha lo stesso valore di MQXP_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQXP

Questo è il numero di versione della struttura. Il valore deve essere:

MQXP_VERSION_1
Numero di versione per il parametro di uscita - struttura blocco.

Nota: Quando viene introdotta una nuova versione di questa struttura, il layout della parte esistente non viene modificato. L'uscita deve quindi controllare che il numero di versione sia uguale o superiore alla versione più bassa che contiene i campi che l'uscita deve utilizzare.

Questo è un campo di immissione per l'uscita.

ExitId (MQLONG) per MQXP

Viene impostato all'entrata della routine di uscita e indica il tipo di uscita:

MQXT_API_CROSSING_EXIT
API - crossing exit per CICS.

Questo è un campo di immissione per l'uscita.

ExitReason (MQLONG) per MQXP

Viene impostato all'entrata della routine di uscita. Per l'uscita di attraversamento API indica se la routine viene richiamata prima o dopo l'esecuzione della chiamata API:

MQXR_BEFORE
Prima dell'esecuzione dell'API.

MQXR_XX_ENCODE_CASE_ONE dopo
Dopo l'esecuzione dell'API.

Questo è un campo di immissione per l'uscita.

ExitResponse (MQLONG) per MQXP

Il valore viene impostato dall'uscita per comunicare con il chiamante. Vengono definiti i seguenti valori:

MQXCC_OK
Uscita completata correttamente.

MQXCC_SUPPRESS_FUNZIONE
Sopprimere la funzione.

Quando questo valore è impostato da un'uscita incrociata API denominata *prima* della chiamata API, la chiamata API non viene eseguita. Il *CompCode* per la chiamata è impostato su MQCC_FAILED, il *Reason* è impostato su MQRC_SUPPRESSED_BY_EXIT e tutti gli altri parametri rimangono come l'uscita li ha lasciati.

Quando questo valore viene impostato da un'uscita incrociata API denominata *dopo* la chiamata API, viene ignorato dal gestore code.

FUNZIONE SKIP_MQXCC
Ignora funzione.

Quando questo valore viene impostato da un'uscita incrociata API denominata *prima* della chiamata API, la chiamata API non viene eseguita; *CompCode* e *Reason* e tutti gli altri parametri rimangono come l'uscita li ha lasciati.

Quando questo valore viene impostato da un'uscita incrociata API denominata *dopo* la chiamata API, viene ignorato dal gestore code.

Questo è un campo di output dall'uscita.

ExitCommand (MQLONG) per MQXP

Questo campo è impostato all'entrata della routine di uscita. Identifica la chiamata API che ha causato il richiamo dell'uscita:

MQXC_CALLBACK
La chiamata CALLBACK.

MQX_MQBACK
La chiamata MQBACK.

MQXC_MQCB
La chiamata MQCB.

MQX_MQCLOSE
La chiamata MQCLOSE.

MQX_MQCMIT
La chiamata MQCMIT.

MQX_MQCTL
La chiamata MQCTL.

MQX_MQGET
La chiamata MQGET.

MQX_MQINQ
La chiamata MQINQ.

MQX_MQOPEN
La chiamata MQOPEN.

MQXC_MQPUT
La chiamata MQPUT.

MQXC_MQPUT1
La chiamata MQPUT1 .

MQXC_MQSET
La chiamata MQSET.

MQX_MQSTAT
La chiamata MQSTAT.

MQX_MQSUB
La chiamata MQSUB.

MQXC_MQSUBRQ
La chiamata MQSUBRQ.

Questo è un campo di immissione per l'uscita.

Conteggio ExitParm(MQLONG) per MQXP

Questo campo è impostato all'entrata della routine di uscita. Contiene il numero di parametri utilizzati dalla chiamata MQ .

Tabella 538. Numero di parametri per ogni chiamata MQ

Nome chiamata	Numero di parametri
MQBACK	3
MQCLOSE	5
MQCMIT	3
MQGET	9
MQINQ	10
MQOPEN	6
MQPUT	8
MQPUT1	8
MQSET	10

Questo è un campo di immissione per l'uscita.

Riservato (MQLONG) per MQXP

Questo è un campo riservato. Il suo valore non è significativo per l'uscita.

Area ExitUser(MQBYTE16) per MQXP

Questo è un campo disponibile per l'uscita da utilizzare. Viene inizializzato su zero binario per la lunghezza del campo prima del primo richiamo dell'uscita per il task, e in seguito tutte le modifiche apportate a questo campo dall'uscita vengono conservate tra i richiami dell'uscita. Viene definito il seguente valore:

MQXUA_NONE
Nessuna informazione utente.

Il valore è zero binario per la lunghezza del campo.

Per il linguaggio di programmazione C, viene definita anche la costante MQXUA_NONE_ARRAY; ha lo stesso valore di MQXUA_NONE, ma è un array di caratteri invece di una stringa.

La lunghezza di questo campo è fornita da MQ_EXIT_USER_AREA_LENGTH. Questo è un campo di immissione / emissione per l'uscita.

MQXQH - Intestazione coda di trasmissione

La struttura MQXQH descrive le informazioni prefissate ai dati dei messaggi dell'applicazione quando si trovano nelle code di trasmissione.Una coda di trasmissione è un tipo speciale di coda locale che contiene temporaneamente i messaggi destinati alle code remote (ossia, destinati alle code che non appartengono al gestore code locale). Una coda di trasmissione è denotata dall'attributo coda **Usage** con il valore MQUS_TRANSMISSION.

Nome formato

MQFMT_XMIT_Q_HEADER

Serie di caratteri e codifica

I dati in MQXQH devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da MQENC_NATIVE.

Impostare la serie di caratteri e la codifica di MQXQH nei campi *CodedCharSetId* e *Encoding* in:

- MQMD separato (se la struttura MQXQH si trova all'inizio dei dati del messaggio) oppure
- La struttura dell'intestazione che precede la struttura MQXQH (tutti gli altri casi).

Campi

Nota: Nella seguente tabella, i campi sono raggruppati per utilizzo piuttosto che in ordine alfabetico. Gli argomenti secondari seguono la stessa sequenza.

Tabella 539. Campi in MQXQH per MQXQH

Nome e descrizione del campo	Nome della costante	Valore iniziale (se presente) della costante
StrucId (identificativo della struttura)	ID_STRUC_MQXQH	'XQH¬'
Versione (numero versione struttura)	MQXQH_VERSION_1	1
RemoteQName (nome della coda di destinazione)	Nessuna	Stringa null o spazi vuoti
RemoteQMgrNome (nome del gestore code di destinazione)	Nessuna	Stringa null o spazi vuoti

Tabella 539. Campi in MQXQH per MQXQH (Continua)		
Nome e descrizione del campo	**Nome della costante**	**Valore iniziale (se presente) della costante**
MsgDesc (descrittore messaggio originale)	Stessi nomi e valori di MQMD; consultare Tabella 500 a pagina 433	-

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.
3. Nel linguaggio di programmazione C, la variabile macroMQXQH_DEFAULT contiene i valori elencati nella tabella. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQXQH MyXQH = {MQXQH_DEFAULT};
```

Dichiarazioni di lingua

Dichiarazione C per MQXQH

```
typedef struct tagMQXQH MQXQH;
struct tagMQXQH {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQCHAR48  RemoteQName;     /* Name of destination queue */
  MQCHAR48  RemoteQMgrName;  /* Name of destination queue manager */
  MQMD1     MsgDesc;         /* Original message descriptor */
};
```

Dichiarazione COBOL per MQXQH

```
**   MQXQH structure
  10 MQXQH.
**    Structure identifier
   15 MQXQH-STRUCID                  PIC X(4).
**    Structure version number
   15 MQXQH-VERSION                  PIC S9(9) BINARY.
**    Name of destination queue
   15 MQXQH-REMOTEQNAME              PIC X(48).
**    Name of destination queue manager
   15 MQXQH-REMOTEQMGRNAME           PIC X(48).
**    Original message descriptor
   15 MQXQH-MSGDESC.
**     Structure identifier
    20 MQXQH-MSGDESC-STRUCID          PIC X(4).
**     Structure version number
    20 MQXQH-MSGDESC-VERSION          PIC S9(9) BINARY.
**     Report options
    20 MQXQH-MSGDESC-REPORT           PIC S9(9) BINARY.
**     Message type
    20 MQXQH-MSGDESC-MSGTYPE          PIC S9(9) BINARY.
**     Expiry time
    20 MQXQH-MSGDESC-EXPIRY           PIC S9(9) BINARY.
**     Feedback or reason code
    20 MQXQH-MSGDESC-FEEDBACK         PIC S9(9) BINARY.
**     Numeric encoding of message data
    20 MQXQH-MSGDESC-ENCODING         PIC S9(9) BINARY.
**     Character set identifier of message data
    20 MQXQH-MSGDESC-CODEDCHARSETID   PIC S9(9) BINARY.
**     Format name of message data
```

```
    20 MQXQH-MSGDESC-FORMAT           PIC X(8).
**     Message priority
    20 MQXQH-MSGDESC-PRIORITY         PIC S9(9) BINARY.
**     Message persistence
    20 MQXQH-MSGDESC-PERSISTENCE      PIC S9(9) BINARY.
**     Message identifier
    20 MQXQH-MSGDESC-MSGID            PIC X(24).
**     Correlation identifier
    20 MQXQH-MSGDESC-CORRELID         PIC X(24).
**     Backout counter
    20 MQXQH-MSGDESC-BACKOUTCOUNT     PIC S9(9) BINARY.
**     Name of reply-to queue
    20 MQXQH-MSGDESC-REPLYTOQ         PIC X(48).
**     Name of reply queue manager
    20 MQXQH-MSGDESC-REPLYTOQMGR      PIC X(48).
**     User identifier
    20 MQXQH-MSGDESC-USERIDENTIFIER   PIC X(12).
**     Accounting token
    20 MQXQH-MSGDESC-ACCOUNTINGTOKEN  PIC X(32).
**     Application data relating to identity
    20 MQXQH-MSGDESC-APPLIDENTITYDATA PIC X(32).
**     Type of application that put the message
    20 MQXQH-MSGDESC-PUTAPPLTYPE      PIC S9(9) BINARY.
**     Name of application that put the message
    20 MQXQH-MSGDESC-PUTAPPLNAME      PIC X(28).
**     Date when message was put
    20 MQXQH-MSGDESC-PUTDATE          PIC X(8).
**     Time when message was put
    20 MQXQH-MSGDESC-PUTTIME          PIC X(8).
**     Application data relating to origin
    20 MQXQH-MSGDESC-APPLORIGINDATA   PIC X(4).
```

Dichiarazione PL/I per MQXQH

```
dcl
 1 MQXQH based,
  3 StrucId           char(4),       /* Structure identifier */
  3 Version           fixed bin(31), /* Structure version number */
  3 RemoteQName       char(48),      /* Name of destination queue */
  3 RemoteQMgrName    char(48),      /* Name of destination queue
                                        manager */
  3 MsgDesc,                         /* Original message descriptor */
   5 StrucId          char(4),       /* Structure identifier */
   5 Version          fixed bin(31), /* Structure version number */
   5 Report           fixed bin(31), /* Report options */
   5 MsgType          fixed bin(31), /* Message type */
   5 Expiry           fixed bin(31), /* Expiry time */
   5 Feedback         fixed bin(31), /* Feedback or reason code */
   5 Encoding         fixed bin(31), /* Numeric encoding of message
                                        data */
   5 CodedCharSetId   fixed bin(31), /* Character set identifier of
                                        message data */
   5 Format           char(8),       /* Format name of message data */
   5 Priority         fixed bin(31), /* Message priority */
   5 Persistence      fixed bin(31), /* Message persistence */
   5 MsgId            char(24),      /* Message identifier */
   5 CorrelId         char(24),      /* Correlation identifier */
   5 BackoutCount     fixed bin(31), /* Backout counter */
   5 ReplyToQ         char(48),      /* Name of reply-to queue */
   5 ReplyToQMgr      char(48),      /* Name of reply queue manager */
   5 UserIdentifier   char(12),      /* User identifier */
   5 AccountingToken  char(32),      /* Accounting token */
   5 ApplIdentityData char(32),      /* Application data relating to
                                        identity */
   5 PutApplType      fixed bin(31), /* Type of application that put the
                                        message */
   5 PutApplName      char(28),      /* Name of application that put the
                                        message */
   5 PutDate          char(8),       /* Date when message was put */
   5 PutTime          char(8),       /* Time when message was put */
   5 ApplOriginData   char(4);       /* Application data relating to
                                        origin */
```

Dichiarazione High Level Assembler per MQXQH

```
MQXQH                           DSECT
MQXQH_STRUCID                   DS   CL4   Structure identifier
```

```
MQXQH_VERSION                   DS   F     Structure version number
MQXQH_REMOTEQNAME               DS   CL48  Name of destination queue
MQXQH_REMOTEQMGRNAME            DS   CL48  Name of destination queue
*                                          manager
MQXQH_MSGDESC                   DS   0F    Force fullword alignment
MQXQH_MSGDESC_STRUCID           DS   CL4   Structure identifier
MQXQH_MSGDESC_VERSION           DS   F     Structure version number
MQXQH_MSGDESC_REPORT            DS   F     Report options
MQXQH_MSGDESC_MSGTYPE           DS   F     Message type
MQXQH_MSGDESC_EXPIRY            DS   F     Expiry time
MQXQH_MSGDESC_FEEDBACK          DS   F     Feedback or reason code
MQXQH_MSGDESC_ENCODING          DS   F     Numeric encoding of message
*                                          data
MQXQH_MSGDESC_CODEDCHARSETID    DS   F     Character set identifier of
*                                          message data
MQXQH_MSGDESC_FORMAT            DS   CL8   Format name of message data
MQXQH_MSGDESC_PRIORITY          DS   F     Message priority
MQXQH_MSGDESC_PERSISTENCE       DS   F     Message persistence
MQXQH_MSGDESC_MSGID             DS   XL24  Message identifier
MQXQH_MSGDESC_CORRELID          DS   XL24  Correlation identifier
MQXQH_MSGDESC_BACKOUTCOUNT      DS   F     Backout counter
MQXQH_MSGDESC_REPLYTOQ          DS   CL48  Name of reply-to queue
MQXQH_MSGDESC_REPLYTOQMGR       DS   CL48  Name of reply queue manager
MQXQH_MSGDESC_USERIDENTIFIER    DS   CL12  User identifier
MQXQH_MSGDESC_ACCOUNTINGTOKEN   DS   XL32  Accounting token
MQXQH_MSGDESC_APPLIDENTITYDATA  DS   CL32  Application data relating to
*                                          identity
MQXQH_MSGDESC_PUTAPPLTYPE       DS   F     Type of application that put
*                                          the message
MQXQH_MSGDESC_PUTAPPLNAME       DS   CL28  Name of application that put
*                                          the message
MQXQH_MSGDESC_PUTDATE           DS   CL8   Date when message was put
MQXQH_MSGDESC_PUTTIME           DS   CL8   Time when message was put
MQXQH_MSGDESC_APPLORIGINDATA    DS   CL4   Application data relating to
*                                          origin
MQXQH_MSGDESC_LENGTH            EQU  *-MQXQH_MSGDESC
                                ORG  MQXQH_MSGDESC
MQXQH_MSGDESC_AREA              DS   CL(MQXQH_MSGDESC_LENGTH)
*
MQXQH_LENGTH                    EQU  *-MQXQH
                                ORG  MQXQH
MQXQH_AREA                      DS   CL(MQXQH_LENGTH)
```

Dichiarazione Visual Basic per MQXQH

```
Type MQXQH
  StrucId        As String*4  'Structure identifier'
  Version        As Long      'Structure version number'
  RemoteQName    As String*48 'Name of destination queue'
  RemoteQMgrName As String*48 'Name of destination queue manager'
  MsgDesc        As MQMD1     'Original message descriptor'
End Type
```

Campi nel descrittore del messaggio separato

Un messaggio che si trova su una coda di trasmissione ha *due* descrittori di messaggi:

- Un descrittore di messaggi viene archiviato separatamente dai dati del messaggio; questo viene denominato *descrittore di messaggi separato*e viene generato dal gestore code quando il messaggio viene inserito nella coda di trasmissione. Alcuni dei campi nel descrittore del messaggio separato vengono copiati dal descrittore del messaggio fornito dall'applicazione nella chiamata MQPUT o MQPUT1 .

  Il descrittore del messaggio separato è quello restituito all'applicazione nel parametro **MsgDesc** della chiamata MQGET quando il messaggio viene rimosso dalla coda di trasmissione.
- Un secondo descrittore del messaggio viene memorizzato all'interno della struttura MQXQH come parte dei dati del messaggio; questo è denominato *descrittore del messaggio integrato*ed è una copia del descrittore del messaggio fornito dall'applicazione nella chiamata MQPUT o MQPUT1 (con variazioni minori).

Il descrittore del messaggio incorporato è sempre un MQMD version-1 . Se il messaggio immesso dall'applicazione ha valori non predefiniti per uno o più campi version-2 in MQMD, una struttura MQMDE segue MQXQH ed è a sua volta seguita dai dati del messaggio dell'applicazione (se presenti). MQMDE è:

- Generato dal gestore code (se l'applicazione utilizza un MQMD version-2 per inserire il messaggio) oppure
- Già presente all'inizio dei dati del messaggio dell'applicazione (se l'applicazione utilizza un MQMD version-1 per inserire il messaggio).

Il descrittore del messaggio incorporato è quello restituito all'applicazione nel parametro **MsgDesc** della chiamata MQGET quando il messaggio viene rimosso dalla coda di destinazione finale.

I campi nel descrittore del messaggio separato vengono impostati dal gestore code come mostrato. Se il gestore code non supporta l'MQMD version-2 , viene utilizzato un MQMD version-1 senza perdita di funzione.

Tabella 540. Valori utilizzati per i campi in MQMD separato

Campo in MQMD separato	Valore utilizzato
StrucId	ID_STRUC_MQMD
Version	MQMD_VERSION_2
Report	Copiato dal descrittore del messaggio incorporato, ma con i bit identificati da MQRO_ACCEPT_UNSUP_IF_XMIT_MASK impostato su zero. Ciò impedisce che un messaggio di report COA o COD venga generato quando un messaggio viene inserito o rimosso da una coda di trasmissione.
MsgType	Copiato dal descrittore del messaggio incorporato.
Expiry	Copiato dal descrittore del messaggio incorporato.
Feedback	Copiato dal descrittore del messaggio incorporato.
Encoding	MQENC_NATIVE (vedere nota)
CodedCharSetId	Attributo **CodedCharSetId** del gestore code.
Format	MQFMT_XMIT_Q_HEADER
Priority	Copiato dal descrittore del messaggio incorporato.
Persistence	Copiato dal descrittore del messaggio incorporato.
MsgId	Il gestore code genera un nuovo valore. Questo identificativo del messaggio è diverso dal *MsgId* che il gestore code potrebbe aver generato per il descrittore del messaggio integrato descritto in precedenza.
CorrelId	Il *MsgId* dal descrittore del messaggio incorporato. Per i messaggi inseriti nel SISTEMA SYSTEM.CLUSTER.TRANSMIT.QUEUE, *CorrelId* è riservato per uso interno.
BackoutCount	0
ReplyToQ	Copiato dal descrittore del messaggio incorporato.
ReplyToQMgr	Copiato dal descrittore del messaggio incorporato.
UserIdentifier	Copiato dal descrittore del messaggio incorporato.
AccountingToken	Copiato dal descrittore del messaggio incorporato. Per i messaggi inseriti nel SISTEMA SYSTEM.CLUSTER.TRANSMIT.QUEUE, *AccountingToken* è riservato per uso interno.
ApplIdentityData	Copiato dal descrittore del messaggio incorporato.

Tabella 540. Valori utilizzati per i campi in MQMD separato (Continua)

Campo in MQMD separato	Valore utilizzato
PutApplType	Gestore code MQAT
PutApplName	Primi 28 byte del nome del gestore code.
PutDate	Data in cui il messaggio è stato inserito nella coda di trasmissione.
PutTime	L'ora in cui il messaggio è stato inserito nella coda di trasmissione.
ApplOriginData	Spazi
GroupId	MQGI_NONE
MsgSeqNumber	1
Offset	0
MsgFlags	MQMF_NONE
OriginalLength	MQOL_NON DEFINITO

- Su Windows, il valore di MQENC_NATIVE per Micro Focus COBOL è diverso dal valore per C. Il valore nel campo *Encoding* nel descrittore del messaggio separato è sempre il valore per C in questi ambienti; questo valore è 546 in decimale. Inoltre, i campi interi nella struttura MQXQH sono nella codifica che corrisponde a questo valore (la codifica Intel nativa).

Campi nel descrittore del messaggio incorporato

I campi nel descrittore del messaggio integrato hanno gli stessi valori di quelli del parametro **MsgDesc** della chiamata MQPUT o MQPUT1 , ad eccezione dei seguenti:

- Il campo *Version* ha sempre il valore MQMD_VERSION_1.
- Se il campo *Priority* ha il valore MQPRI_PRIORITY_AS_Q_DEF, viene sostituito dal valore dell'attributo **DefPriority** della coda.
- Se il campo *Persistence* ha il valore MQPER_PERSISTENCE_AS_Q_DEF, viene sostituito dal valore dell'attributo **DefPersistence** della coda.
- Se il campo *MsgId* ha il valore MQMI_NONE o se è stata specificata l'opzione MQPMO_NEW_MSG_ID oppure se il messaggio è un messaggio dell'elenco di distribuzione, *MsgId* viene sostituito da un nuovo identificativo del messaggio generato dal gestore code.

  Quando un messaggio dell'elenco di distribuzione viene suddiviso in messaggi dell'elenco di distribuzione più piccoli posizionati su code di trasmissione differenti, il campo *MsgId* in ciascuno dei nuovi descrittori di messaggi incorporati è uguale a quello del messaggio dell'elenco di distribuzione originale.
- Se è stata specificata l'opzione MQPMO_NEW_CORREL_ID, *CorrelId* viene sostituito da un nuovo identificativo di correlazione generato dal gestore code.
- I campi di contesto sono impostati come indicato dalle opzioni di MQPMO_ * _CONTEXT specificate nel parametro **PutMsgOpts** ; i campi di contesto sono:
  - *AccountingToken*
  - *ApplIdentityData*
  - *ApplOriginData*
  - *PutApplName*
  - *PutApplType*
  - *PutDate*
  - *PutTime*
  - *UserIdentifier*

- I campi version-2 (se presenti) vengono rimossi da MQMD e spostati in una struttura MQMDE, se uno o più campi version-2 hanno un valore non predefinito.

Inserimento di messaggi nelle code remote

Quando un'applicazione inserisce un messaggio su una coda remota (specificando direttamente il nome della coda remota o utilizzando una definizione locale della coda remota), il gestore code locale:

- Crea una struttura MQXQH contenente il descrittore del messaggio incorporato
- Accoda un MQMDE se ne è necessario uno e non è già presente
- Accoda i dati del messaggio dell'applicazione
- Inserisce il messaggio in una coda di trasmissione appropriata

Inserimento diretto dei messaggi nelle code di trasmissione

Un'applicazione può anche inserire un messaggio direttamente su una coda di trasmissione. In questo caso l'applicazione deve aggiungere ai dati del messaggio dell'applicazione una struttura MQXQH e inizializzare i campi con valori appropriati. Inoltre, il campo *Format* nel parametro **MsgDesc** della chiamata MQPUT o MQPUT1 deve avere il valore MQFMT_XMIT_Q_HEADER.

I dati carattere nella struttura MQXQH creati dall'applicazione devono trovarsi nella serie di caratteri del gestore code locale (definito dall'attributo gestore code **CodedCharSetId**) e i dati interi devono essere nella codifica della macchina nativa. Inoltre, i dati dei caratteri nella struttura MQXQH devono essere riempiti con spazi vuoti fino alla lunghezza definita del campo; i dati non devono essere terminati prematuramente utilizzando un carattere null, perché il gestore code non converte i caratteri null e successivi in spazi vuoti nella struttura MQXQH.

Tuttavia, il gestore code non controlla che sia presente una struttura MQXQH o che siano stati specificati valori validi per i campi.

Le applicazioni non devono inserire i relativi messaggi direttamente nel SISTEMA SYSTEM.CLUSTER.TRANSMIT.QUEUE.

Richiamo dei messaggi dalle code di trasmissione

Le applicazioni che ricevono messaggi da una coda di trasmissione devono elaborare le informazioni nella struttura MQXQH in modo appropriato. La presenza della struttura di MQXQH all'inizio dei dati del messaggio dell'applicazione è indicata dal valore MQFMT_XMIT_Q_HEADER restituito nel campo *Format* nel parametro **MsgDesc** della chiamata MQGET. I valori restituiti nei campi *CodedCharSetId* e *Encoding* nel parametro **MsgDesc** indicano la serie di caratteri e la codifica dei dati carattere e numero intero nella struttura MQXQH. La serie di caratteri e la codifica dei dati del messaggio dell'applicazione sono definiti dai campi *CodedCharSetId* e *Encoding* nel descrittore del messaggio incorporato.

StrucId (MQCHAR4) per MQXQH

Questo è l'identificativo della struttura dell'intestazione della coda di trasmissione. È sempre un campo di immissione. Il valore è MQXQH_STRUC_ID.

Il valore deve essere:

ID_STRUC_MQXQH
Identificativo per la struttura dell'intestazione della coda di trasmissione.

Per il linguaggio di programmazione C, è definita anche la costante MQXQH_STRUC_ID_ARRAY. Ha lo stesso valore di MQXQH_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione (MQLONG) per MQXQH

Questo è il numero di versione della struttura. Il valore deve essere:

MQXQH_VERSION_1
Numero di versione per la struttura di intestazione della coda di trasmissione.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQXQH_CURRENT_
Versione corrente della struttura dell'intestazione della coda di trasmissione.

Il valore iniziale di questo campo è MQXQH_VERSION_1.

RemoteQName (MQCHAR48) per MQXQH

Questo è il nome della coda messaggi che è la destinazione finale apparente per il messaggio (questa potrebbe non essere la destinazione finale se, ad esempio, questa coda è definita in *RemoteQMgrName* per essere una definizione locale di un'altra coda remota).

Se il messaggio è un messaggio dell'elenco di distribuzione (ovvero, il campo *Format* nel descrittore del messaggio incorporato è MQFMT_DIST_HEADER), *RemoteQName* è vuoto.

La lunghezza di questo campo è fornita da MQ_Q_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

Nome RemoteQMgr(MQCHAR48) per MQXQH

Questo è il nome del gestore code o del gruppo di condivisione code che possiede la coda che è la destinazione finale apparente per il messaggio.

Se il messaggio è un messaggio dell'elenco di distribuzione, *RemoteQMgrName* è vuoto.

La lunghezza di questo campo viene fornita da MQ_Q_MGR_NAME_LENGTH. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione.

MsgDesc (MQMD1) per MQXQH

Questo è il descrittore del messaggio incorporato ed è una copia di chiusura del descrittore del messaggio MQMD specificato come parametro **MsgDesc** nella chiamata MQPUT o MQPUT1 quando il messaggio è stato originariamente inserito sulla coda remota.

Nota: Questo è un MQMD version-1 .

I valori iniziali dei campi in questa struttura sono gli stessi della struttura MQMD.

Chiamate della funzione

Questa sezione fornisce informazioni su tutte le chiamate MQI possibili. Le descrizioni, la sintassi, le informazioni sui parametri, le note sull'utilizzo e i richiami della lingua per ogni lingua possibile vengono forniti per ciascuna delle diverse chiamate.

Riferimenti correlati

z/OS Esempi di output CEDF da chiamate MQI

Descrizioni chiamate

Questa sezione descrive le chiamate MQI.

- “MQCRTMH - Creazione handle messaggio” a pagina 692
- “MQCTL - Controllo callback” a pagina 696
- “MQDISC - Disconnetti gestore code” a pagina 702
- “MQDLTMH - Elimina gestione messaggi” a pagina 706
- “MQDLTMP - Proprietà Elimina messaggio” a pagina 708
- “MQGET - Richiama messaggio” a pagina 711
- “MQINQ - Richiedi attributi oggetto” a pagina 723
- “MQINQMP - Proprietà del messaggio di interrogazione” a pagina 741
- “MQMHBUF - Conversione dell'handle del messaggio in buffer” a pagina 746
- “MQOPEN - Apri oggetto” a pagina 751
- “MQPUT - Inserisci messaggio” a pagina 768
- “MQPUT1 - Inserire un messaggio” a pagina 782
- “MQSET - Imposta attributi oggetto” a pagina 793
- “MQSETMP - Imposta proprietà messaggio” a pagina 799
- “MQSTAT - Richiamo delle informazioni di stato” a pagina 803
- “MQMHBUF - Conversione dell'handle del messaggio in buffer” a pagina 746
- “MQSUB - Registrazione sottoscrizione” a pagina 807
- “MQSUBRQ - Richiesta di sottoscrizione” a pagina 815

La guida in linea sulle piatteforme UNIX , sotto forma di pagine *man* , è disponibile per queste chiamate.

Nota: Le chiamate associate alla conversione dati, MQXCNVC e MQ_DATA_CONV_EXIT, si trovano in “Uscita conversione dati” a pagina 932.

Convenzioni utilizzate nelle descrizioni delle chiamate

Per ogni chiamata, questa raccolta di argomenti fornisce una descrizione dei parametri e dell'utilizzo della chiamata in un formato indipendente dal linguaggio di programmazione. Ciò è seguito da richiami tipici della chiamata e dichiarazioni tipiche dei parametri, in ciascuno dei linguaggi di programmazione supportati.

Importante: Durante la codifica delle chiamate API IBM MQ è necessario assicurarsi che siano forniti tutti i parametri rilevanti (come descritto nelle seguenti sezioni). In caso contrario, si potrebbero ottenere risultati imprevedibili.

La descrizione di ogni chiamata contiene le sezioni seguenti:

Nome chiamata
Il nome della chiamata, seguito da una breve descrizione dello scopo della chiamata.

Parametri
Per ogni parametro, il nome è seguito dal relativo tipo di dati tra parentesi () e uno dei seguenti:

input
Fornire le informazioni nel parametro quando si effettua la chiamata.

output
Il gestore code restituisce le informazioni nel parametro quando la chiamata viene completata o non riesce.

I/O
Si forniscono le informazioni nel parametro quando si effettua la chiamata e il gestore code modifica le informazioni quando la chiamata viene completata o non riesce.

Ad esempio:

Compcode (MQLONG) - output

In alcuni casi, il tipo di dati è una struttura. In tutti i casi, sono disponibili ulteriori informazioni sul tipo di dati o sulla struttura in “Tipi di dati elementari” a pagina 236.

Gli ultimi due parametri in ogni chiamata sono un codice di completamento e un codice motivo. Il codice di completamento indica se la chiamata è stata completata correttamente, parzialmente o per nulla. Ulteriori informazioni sull'esito positivo parziale o sull'esito negativo della chiamata sono fornite nel codice di errore. Per ulteriori informazioni su ciascun codice motivo e di completamento, consultare “Codici di ritorno” a pagina 898.

Note d'utilizzo
Ulteriori informazioni sulla chiamata, che descrivono come utilizzarla e le eventuali limitazioni al suo utilizzo.

Richiamo linguaggio Assembler
Richiamo tipico della chiamata e dichiarazione dei suoi parametri, in linguaggio assembler.

Richiamo C
Richiamo tipico della chiamata e dichiarazione dei relativi parametri, in C.

Richiamo COBOL
Chiamata tipica della chiamata e dichiarazione dei relativi parametri in COBOL.

Chiamata PL/I
Richiamo tipico della chiamata, e dichiarazione dei suoi parametri, in PL/I.

Tutti i parametri vengono passati per riferimento.

Richiamo Visual Basic
Richiamo tipico della chiamata e dichiarazione dei relativi parametri in Visual Basic.

Altre convenzioni di notazione sono:

Costanti
I nomi delle costanti vengono visualizzati in maiuscolo; ad esempio, MQOO_OUTPUT. Una serie di costanti con lo stesso prefisso viene mostrata come segue: MQIA_ *. Consultare “Costanti” a pagina 61 per il valore di una costante.

Array
In alcune chiamate, i parametri sono schiere di stringhe di caratteri che non hanno dimensioni fisse. Nelle descrizioni di questi parametri, una n minuscola rappresenta una costante numerica. Quando si codifica la dichiarazione per tale parametro, sostituire n con il valore numerico richiesto.

Utilizzo delle chiamate in linguaggio C

I parametri che sono *solo input* e di tipo MQHCONN, MQHOBJ, MQHMSG o MQLONG vengono passati per valore. Per tutti gli altri parametri, l' *indirizzo* del parametro viene passato per valore.

Non è necessario specificare tutti i parametri passati per indirizzo ogni volta che si richiama una funzione. Quando non è necessario un particolare parametro, specificare un puntatore null come parametro sul richiamo della funzione, al posto dell'indirizzo dei dati del parametro. I parametri per i quali ciò è possibile sono identificati nelle descrizioni delle chiamate.

Nessun parametro viene restituito come valore della chiamata; nella terminologia C, ciò indica che tutte le chiamate restituiscono `void`.

Dichiarazione del parametro Buffer

Le chiamate **MQGET**, **MQPUT**e **MQPUT1** hanno ciascuna un parametro con un tipo di dati non definito: il parametro *Buffer* . Utilizzare questo parametro per inviare e ricevere i dati del messaggio dell'applicazione.

I parametri di questo tipo vengono mostrati negli esempi C come array di MQBYTE. È possibile dichiarare i parametri in questo modo, ma di solito è più conveniente dichiararli come la struttura particolare che descrive il layout dei dati nel messaggio. Il prototipo della funzione dichiara il parametro come un puntatore a void, in modo che sia possibile specificare l'indirizzo di qualsiasi tipo di dati come parametro sul richiamo della chiamata.

Puntatore a vuoto è un puntatore ai dati di formato non definito. È definito come:

```
typedef void *PMQVOID;
```

MQBACK - Modifiche di back-out

La chiamata MQBACK indica al gestore code che è necessario eseguire il backout di tutti i messaggi richiamati e inseriti verificatisi dall'ultimo punto di sincronizzazione.

I messaggi inseriti come parte di un'unità di lavoro vengono eliminati; i messaggi richiamati come parte di un'unità di lavoro vengono reintegrati nella coda.

- Su z/OS, questa chiamata viene usata solo da programmi batch (inclusi IMS programmi DL/I batch).

Sintassi

MQBACK (*Hconn*, *Compcode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

MQRC_OUTCOME_PENDING
(2124, X'84C') Il risultato dell'operazione di backout è in sospeso.

Se *CompCode* è MQCC_FAILED:

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

MQRC_CF_STRUC_IN_USO
(2346, X'92A') Struttura CF in uso.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

ERRORE MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Chiamata non valida nell'ambiente.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

MQRC_OBJECT_DAMAGED
(2101, X'835 ') Oggetto danneggiato.

MQRC_OUTCOME_MIXED
(2123, X'84B') Il risultato dell'operazione di commit o di backout è misto.

MQRC_Q_MGR_STOPPING
(2162, X'872 ') Chiusura del gestore code.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890 ') Il supporto di memoria esterna è pieno.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici motivo

Note d'utilizzo

1. È possibile utilizzare questa chiamata solo quando il gestore code stesso coordina l'unità di lavoro. Questo può essere:
   - Un'unità di lavoro locale, in cui le modifiche influiscono solo sulle risorse MQ .
   - Un'unità di lavoro globale, in cui le modifiche possono influire sulle risorse appartenenti ad altri gestori risorse, nonché sulle risorse MQ .

   Per ulteriori dettagli sulle unità di lavoro locali e globali, consultare “MQBEGIN - Inizio unità di lavoro” a pagina 649.
2. In ambienti in cui il gestore code non coordina l'unità di lavoro, utilizzare la chiamata di backout appropriata invece di MQBACK. L'ambiente potrebbe anche supportare un backout implicito causato dalla chiusura anomala dell'applicazione.
   - Su z/OS, utilizzare le seguenti chiamate:
     - I programmi batch (inclusi i programmi IMS batch DL/I) possono utilizzare la chiamata MQBACK se l'unità di lavoro interessa solo le risorse MQ . Tuttavia, se l'unità di lavoro influenza sia le risorse MQ che le risorse appartenenti ad altri gestori risorse (ad esempio, Db2), utilizzare la chiamata SRRBACK fornita da RRS (Recoverable Resource Service) z/OS . La chiamata SRRBACK ripristina le modifiche alle risorse appartenenti ai gestori risorse che sono state abilitate per il coordinamento RRS.
     - Le applicazioni CICS devono utilizzare il comando `EXEC CICS SYNCPOINT ROLLBACK` per eseguire il backout dell'unità di lavoro. Non utilizzare la chiamata MQBACK per applicazioni CICS .
     - Le applicazioni IMS (diverse dai programmi DL/I batch) devono utilizzare chiamate IMS come `ROLB` per eseguire il backout dell'unità di lavoro. Non utilizzare la chiamata MQBACK per applicazioni IMS (diverse dai programmi DL/I batch).

- Su IBM i, utilizzare questa chiamata per le unità di lavoro locali coordinate dal gestore code. Ciò significa che una definizione di commit non deve esistere a livello di lavoro, ovvero il comando STRCMTCTL con il parametro **CMTSCOPE(*JOB)** non deve essere stato immesso per il lavoro.

3. Se un'applicazione termina con modifiche non sottoposte a commit in un'unità di lavoro, la disposizione di tali modifiche dipende dal fatto che l'applicazione termini normalmente o in modo anomalo. Consultare le note sull'utilizzo in "MQDISC - Disconnetti gestore code" a pagina 702 per ulteriori dettagli.
4. Quando un'applicazione inserisce o richiama i messaggi in gruppi o segmenti di messaggi logici, il gestore code conserva le informazioni relative al gruppo di messaggi e al messaggio logico per le ultime chiamate MQPUT e MQGET riuscite. Queste informazioni sono associate all'handle della coda e includono:
   - I valore dei campi *GroupId*, *MsgSeqNumber*, *Offset*e *MsgFlags* in MQMD.
   - Se il messaggio fa parte di un'unità di lavoro.
   - Per la chiamata MQPUT: se il messaggio è persistente o non persistente.

   Il gestore code conserva *tre* serie di informazioni sui gruppi e sui segmenti, una serie per ciascuno dei seguenti:
   - L'ultima chiamata MQPUT riuscita (può far parte di un'unità di lavoro).
   - L'ultima chiamata MQGET riuscita che ha rimosso un messaggio dalla coda (può far parte di un'unità di lavoro).
   - L'ultima chiamata MQGET riuscita che ha visualizzato un messaggio sulla coda (non può far parte di un'unità di lavoro).
5. Le informazioni associate alla chiamata MQGET vengono ripristinate al valore che aveva prima della prima chiamata MQGET riuscita per tale handle di coda nell'unità di lavoro corrente.

   Le code che sono state aggiornate dall'applicazione dopo l'avvio dell'unità di lavoro, ma al di fuori dell'ambito dell'unità di lavoro, non hanno le relative informazioni sul gruppo e sul segmento ripristinate se viene eseguito il backout dell'unità di lavoro.

   Il ripristino delle informazioni sul gruppo e sul segmento al suo valore precedente quando viene eseguito il backout di un'unità di lavoro consente all'applicazione di distribuire un gruppo di messaggi di grandi dimensioni o un messaggio logico di grandi dimensioni costituito da molti segmenti in diverse unità di lavoro e di riavviare nel punto corretto nel gruppo di messaggi o nel messaggio logico in caso di errore di una delle unità di lavoro.

   L'utilizzo di diverse unità di lavoro potrebbe essere vantaggioso se il gestore code locale ha solo una memoria di coda limitata. Tuttavia, l'applicazione deve conservare informazioni sufficienti per essere in grado di riavviare l'inserimento o il richiamo dei messaggi nel punto corretto se si verifica un errore di sistema.

   Per i dettagli su come riavviare il sistema nel punto corretto dopo un malfunzionamento del sistema, consultare l'opzione MQPMO_LOGICAL_ORDER descritta in "MQPMO - Opzioni inserimento messaggio" a pagina 512e l'opzione MQGMO_LOGICAL_ORDER descritta in "MQGMO - Opzioni Get - message" a pagina 376.

   Le note di utilizzo rimanenti si applicano solo quando il Gestore code coordina le unità di lavoro.
6. Un'unità di lavoro ha lo stesso ambito di un handle di connessione. Tutte le chiamate MQ che interessano una particolare unità di lavoro devono essere eseguite utilizzando lo stesso handle di connessione. Le chiamate emesse utilizzando un handle di collegamento differente (ad esempio, le chiamate emesse da un'altra applicazione) influenzano un'unità di lavoro diversa. Per informazioni sull'ambito degli handle di connessione, consultare il parametro **Hconn** descritto in "MQCONN - Connetti gestore code" a pagina 679 .
7. Solo i messaggi inseriti o richiamati come parte dell'unità di lavoro corrente vengono influenzati da questa chiamata.
8. Un'applicazione di lunga durata che emette chiamate MQGET, MQPUT o MQPUT1 all'interno di un'unità di lavoro, ma che non emette mai una chiamata di commit o di backout, può riempire le

code con messaggi che non sono disponibili per altre applicazioni. Per evitare questa possibilità, l'amministratore deve impostare l'attributo del gestore code **MaxUncommittedMsgs** su un valore sufficientemente basso per evitare che le applicazioni runaway riempiano le code, ma abbastanza alto per consentire il corretto funzionamento delle applicazioni di messaggistica previste.

Richiamo C

```
MQBACK (Hconn, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQBACK' USING HCONN, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQBACK (Hconn, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQBACK,(HCONN,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Richiamo Visual Basic

```
MQBACK Hconn, CompCode, Reason
```

Dichiarare i parametri come segue:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQBEGIN - Inizio unità di lavoro

La chiamata MQBEGIN avvia un'unità di lavoro coordinata dal gestore code e che può coinvolgere gestori risorse esterni.

Sintassi

MQBEGIN (*Hconn*, *BeginOptions*, *Compcode*, *Reason*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

Hconn deve essere un handle di connessione non condiviso. Se viene specificato un handle di connessione condiviso, la chiamata ha esito negativo con codice motivo MQRC_HCONN_ERROR. Consultare la descrizione delle opzioni MQCNO_HANDLE_SHARE_ * in “MQCNO - Opzioni di connessione” a pagina 321 per ulteriori informazioni sugli handle condivisi e non condivisi.

BeginOptions
Tipo: MQBO - input/output

Si tratta di opzioni che controllano l'azione di MQBEGIN, come descritto in “MQBO - Opzioni di inizio” a pagina 283.

Se non è richiesta alcuna opzione, i programmi scritti nell'assembler C o S/390 possono specificare un indirizzo di parametro null, invece di specificare l'indirizzo di una struttura MQBO.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

MQRC_NO_EXTERNAL_PARTECIPANTI
(2121, X'849 ') Nessun gestore risorse partecipante registrato.

MQRC_PARTICIPANT_NON_DISPONIBILE
(2122, X'84A') Il gestore risorse partecipante non è disponibile.

Se *CompCode* è MQCC_FAILED:

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita.

ERRORE BO_MQRC
(2134, X'856 ') Struttura opzioni di inizio non valida.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

ERRORE MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Chiamata non valida nell'ambiente.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

ERRORE MQRC_OPTIONS_
(2046, X'7FE') Opzioni non valide o non congruenti.

MQRC_Q_MGR_STOPPING
(2162, X'872 ') Chiusura del gestore code.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

MQRC_UOW_IN_PROVERDE
(2128, X'850 ') Unità di lavoro già avviata.

Per ulteriori informazioni su questi codici, consultare Messaggi e codici di errore.

Note d'utilizzo

1. Utilizzare la chiamata MQBEGIN per avviare un'unità di lavoro coordinata dal gestore code e che potrebbe comportare modifiche alle risorse di proprietà di altri gestori risorse. Il gestore code supporta tre tipi di unità di lavoro:
   - **Unità di lavoro locale coordinata dal gestore code**: un'unità di lavoro in cui il gestore code è l'unico gestore risorse partecipante e quindi il gestore code agisce come coordinatore dell'unità di lavoro.
     - Per avviare questo tipo di unità di lavoro, specificare l'opzione MQPMO_SYNCPOINT o MQGMO_SYNCPOINT nella prima chiamata MQPUT, MQPUT1o MQGET nell'unità di lavoro.
     - Per eseguire il commit o il backout di questo tipo di unità di lavoro, utilizzare la chiamata MQCMIT o MQBACK.
   - **Unità di lavoro globale coordinata dal gestore code**: un'unità di lavoro in cui il gestore code funge da coordinatore dell'unità di lavoro, sia per le risorse MQ *che* per le risorse appartenenti ad altri gestori risorse. Questi gestori risorse cooperano con il gestore code per garantire che tutte le modifiche alle risorse nell'unità di lavoro siano sottoposte a commit o a backout insieme.
     - Per avviare questo tipo di unità di lavoro, utilizzare la chiamata MQBEGIN.
     - Per eseguire il commit o il backout di questo tipo di unità di lavoro, utilizzare le chiamate MQCMIT e MQBACK.
   - **Unità di lavoro globale coordinata esternamente**: un'unità di lavoro in cui il gestore code è un partecipante, ma il gestore code non agisce come coordinatore dell'unità di lavoro. Esiste invece un coordinatore dell'unità di lavoro esterna con cui collabora il gestore code.

– Per avviare questo tipo di unità di lavoro, utilizzare la chiamata pertinente fornita dal coordinatore unità di lavoro esterno.

  Se la chiamata MQBEGIN viene utilizzata per tentare di avviare l'unità di lavoro, la chiamata ha esito negativo con il codice motivo MQRC_ENVIRONMENT_ERROR.

– Per eseguire il commit o il backout di questo tipo di unità di lavoro, utilizzare le chiamate di commit e backout fornite dal coordinatore dell'unità di lavoro esterno.

  Se si utilizza la chiamata MQCMIT o MQBACK per eseguire il commit o il backout dell'unità di lavoro, la chiamata ha esito negativo con codice motivo MQRC_ENVIRONMENT_ERROR.

2. Se l'applicazione termina con modifiche non sottoposte a commit in un'unità di lavoro, la disposizione di tali modifiche dipende dal fatto che l'applicazione termini normalmente o in modo anomalo. Consultare le note sull'utilizzo in "MQDISC - Disconnetti gestore code" a pagina 702 per ulteriori dettagli.
3. Un'applicazione può partecipare a una sola unità di lavoro alla volta. La chiamata MQBEGIN ha esito negativo con codice motivo MQRC_UOW_IN_PROGRESS se esiste già un'unità di lavoro per l'applicazione, indipendentemente dal tipo di unità di lavoro.
4. La chiamata MQBEGIN non è valida in un ambiente client MQI MQ . Un tentativo di utilizzare la chiamata ha esito negativo con codice motivo MQRC_ENVIRONMENT_ERROR.
5. Quando il gestore code agisce come coordinatore dell'unità di lavoro per le unità di lavoro globali, i gestori risorse che possono partecipare all'unità di lavoro sono definiti nel file di configurazione del gestore code.
6. Su IBM i, i tre tipi di unità di lavoro sono supportati come segue:
   - L' **unità di lavoro locale coordinata dal gestore code** può essere utilizzata solo quando non è presente una definizione di commit a livello di lavoro, ovvero il comando STRCMTCTL con il parametro **CMTSCOPE(*JOB)** non deve essere stato immesso per il lavoro.
   - **Unità di lavoro globale coordinata dal gestore code** non è supportata.
   - **Unità di lavoro globale coordinata esternamente** può essere utilizzata solo quando esiste una definizione di commit a livello di lavoro, vale a dire, il comando STRCMTCTL con il parametro **CMTSCOPE(*JOB)** deve essere stato immesso per il lavoro. Se questa operazione è stata eseguita, le operazioni di IBM i COMMIT e ROLLBACK si applicano alle risorse MQ e alle risorse appartenenti ad altri gestori risorse partecipanti.

Richiamo C

```
MQBEGIN (Hconn, &BeginOptions, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;         /* Connection handle */
MQBO     BeginOptions;  /* Options that control the action of MQBEGIN */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQBEGIN' USING HCONN, BEGINOPTIONS, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Options that control the action of MQBEGIN
 01  BEGINOPTIONS.
     COPY CMQBOV.
```

```
**   Completion code
 01  COMPCODE      PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQBEGIN (Hconn, BeginOptions, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn         fixed bin(31);  /* Connection handle */
dcl BeginOptions  like MQBO;      /* Options that control the action of
                                     MQBEGIN */
dcl CompCode      fixed bin(31);  /* Completion code */
dcl Reason        fixed bin(31);  /* Reason code qualifying CompCode */
```

Richiamo Visual Basic

```
MQBEGIN Hconn, BeginOptions, CompCode, Reason
```

Dichiarare i parametri come segue:

```
Dim Hconn        As Long 'Connection handle'
Dim BeginOptions As MQBO 'Options that control the action of MQBEGIN'
Dim CompCode     As Long 'Completion code'
Dim Reason       As Long 'Reason code qualifying CompCode'
```

MQBUFMH - Converti buffer in handle del messaggio

La chiamata alla funzione MQBUFMH converte un buffer in un handle del messaggio ed è l'inverso della chiamata MQMHBUF.

Questa chiamata acquisisce un descrittore di messaggi e le proprietà MQRFH2 nel buffer e li rende disponibili tramite un handle del messaggio. Le proprietà MQRFH2 nei dati del messaggio vengono, facoltativamente, rimosse. I campi *Encoding*, *CodedCharSetId*e *Format* del descrittore del messaggio vengono aggiornati, se necessario, per descrivere correttamente il contenuto del buffer dopo la rimozione delle proprietà.

Sintassi

MQBUFMH (*Hconn*, *Hmsg*, *BufMsgHOpts*, *MsgDesc*, *BufferLength*, *Buffer*, *DataLength*, *Compcode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di **Hconn** deve corrispondere all'handle di connessione utilizzato per creare l'handle del messaggio specificato nel parametro **Hmsg** .

Se l'handle del messaggio è stato creato utilizzando MQHC_UNASSOCIATED_HCONN, è necessario stabilire una connessione valida sul thread convertendo un buffer in un handle del messaggio. Se non viene stabilita una connessione valida, la chiamata ha esito negativo con MQRC_CONNECTION_BROKEN.

Msg
Tipo: MQHMQSG - input

Questo è l'handle del messaggio per cui è richiesto un buffer. Il valore è stato restituito da una precedente chiamata MQCRTMH.

BufMsgHOpts
Tipo: MQBMHO - input

La struttura MQBMHO consente alle applicazioni di specificare le opzioni che controllano il modo in cui i gestori dei messaggi vengono prodotti dai buffer.

Vedi "MQBMHO - Buffer per le opzioni di gestione dei messaggi" a pagina 278 per i dettagli.

MsgDesc
Tipo: MQMD - input/output

La struttura *MsgDesc* contiene le proprietà descrittore del messaggio e descrive il contenuto dell'area di buffer.

In fase di output dalla chiamata, le proprietà vengono facoltativamente rimosse dall'area del buffer e, in questo caso, il descrittore del messaggio viene aggiornato per descrivere correttamente l'area del buffer.

I dati in questa struttura devono essere nella serie di caratteri e nella codifica dell'applicazione.

BufferLength
Tipo: MQLONG - input

BufferLength è la lunghezza dell'area Buffer, in byte.

Un *BufferLength* di zero byte è valido e indica che l'area buffer non contiene dati.

Memorizza nel buffer
Tipo: MQBYTExBufferLength - input/output

Si tratta di opzioni che controllano l'azione di MQBEGIN, come descritto in "MQBEGIN - Inizio unità di lavoro" a pagina 649.

Buffer definisce l'area contenente il buffer di messaggi. Per la maggior parte dei dati, è necessario allineare il buffer su un limite di 4 byte.

Se **Buffer** contiene dati numerici o di caratteri, impostare i campi *CodedCharSetId* e *Encoding* nel parametro **MsgDesc** sui valori appropriati per i dati; ciò consente la conversione dei dati, se necessario.

Se le proprietà vengono trovate nel buffer del messaggio, vengono facoltativamente rimosse; in seguito diventano disponibili dall'handle del messaggio al ritorno dalla chiamata.

Nel linguaggio di programmazione C, il parametro viene dichiarato come un puntatore a void, il che significa che l'indirizzo di qualsiasi tipo di dati può essere specificato come parametro.

Se il parametro **BufferLength** è zero, **Buffer** non viene indicato; in questo caso, l'indirizzo del parametro passato dai programmi scritti nell'assembler C o System/390 può essere null.

DataLength
Tipo: MQLONG - output

La lunghezza, in byte, del buffer per cui potrebbero essere rimosse le proprietà.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'089C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

ERRORE MQRC_BMHO_
(2489, X'09B9') Buffer per la struttura delle opzioni di gestione messaggi non valida.

ERRORE MQRC_BUFFER_
Parametro buffer (2004, X'07D4') non valido.

ERRORE MQRC_BUFFER_LENGTH
Parametro di lunghezza buffer (2005, X'07D5') non valido.

MQRC_CALL_IN_PROVERDE
(2219, X'08AB') Chiamata MQI immessa prima del completamento della chiamata precedente.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Connessione al gestore code persa.

ERRORE MQRC_HMSG_
(2460, X'099C') Gestione messaggio non valida.

ERRORE MQRC_MD
(2026, X'07EA') Descrittore messaggio non valido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') handle del messaggio già in uso.

ERRORE MQRC_OPTIONS_
(2046, X'07FE') Opzioni non valide o non congruenti.

ERRORE MQRC_RFH
(2334, X'091E') Struttura MQRFH2 non valida.

ERRORE MQRC_RFH_FORMATO
(2421, X'0975 ') Impossibile analizzare una cartella MQRFH2 contenente le proprietà.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note d'utilizzo

Le chiamate MQBUFMH non possono essere intercettate dalle uscite API - un buffer viene convertito in un handle del messaggio nello spazio dell'applicazione; la chiamata non raggiunge il gestore code.

Richiamo C

```
MQBUFMH (Hconn, Hmsg, &BufMsgHOpts, &MsgDesc, BufferLength, Buffer,
         &DataLength, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN Hconn;        /* Connection handle */
MQHMSG  Hmsg;         /* Message handle */
MQBMHO  BufMsgHOpts;  /* Options that control the action of MQBUFMH */
```

```
MQMD    MsgDesc;      /* Message descriptor */
MQLONG  BufferLength; /* Length in bytes of the Buffer area */
MQBYTE  Buffer[n];    /* Area to contain the message buffer */
MQLONG  DataLength;   /* Length of the output buffer */
MQLONG  CompCode;     /* Completion code */
MQLONG  Reason;       /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQBUFMH' USING HCONN, HMSG, BUFMSGHOPTS, MSGDESC, BUFFERLENGTH,
                           BUFFER, DATALENGTH, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Message handle
 01  HMSG          PIC S9(18) BINARY.
**   Options that control the action of MQBUFMH
 01  BUFMSGHOPTS.
     COPY CMQBMHOV.
**   Message descriptor
 01  MSGDESC.
     COPY CMQMD.
**   Length in bytes of the Buffer area
 01  BUFFERLENGTH  PIC S9(9) BINARY.
**   Area to contain the message buffer
 01  BUFFER        PIC X(n).
**   Length of the output buffer
 01  DATALENGTH    PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE      PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQBUFMH (Hconn, Hmsg, BufMsgHOpts, MsgDesc, BufferLength, Buffer,
DataLength, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn        fixed bin(31); /* Connection handle */
dcl Hmsg         fixed bin(63); /* Message handle */
dcl BufMsgHOpts  like MQBMHO;   /* Options that control the action of
                                   MQBUFMH */
dcl MsgDesc      like MQMD;     /* Message descriptor */
dcl BufferLength fixed bin(31); /* Length in bytes of the Buffer area */
dcl Buffer       char(n);       /* Area to contain the message buffer */
dcl DataLength   fixed bin(31); /* Length of the output buffer */
dcl CompCode     fixed bin(31); /* Completion code */
dcl Reason       fixed bin(31); /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQBUFMH,(HCONN,HMSG,BUFMSGHOPTS,MSGDESC,BUFFERLENGTH,BUFFER,
               DATALENGTH,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN        DS       F      Connection handle
HMSG         DS       D      Message handle
BUFMSGHOPTS  CMQBMHOA ,      Options that control the action of MQBUFMH
MSGDESC      CMQMDA   ,      Message descriptor
```

```
BUFFERLENGTH DS       F      Length in bytes of the BUFFER area
BUFFER       DS       CL(n)  Area to contain the properties
DATALENGTH   DS       F      Length of the output buffer
COMPCODE     DS       F      Completion code
REASON       DS       F      Reason code qualifying COMPCODE
```

MQCB - Gestisci callback

La chiamata MQCB registra un callback per l'handle di oggetto specificato e controlla l'attivazione e le modifiche al callback.

Un callback è una parte di codice (specificato come nome di una funzione che può essere collegata dinamicamente o come puntatore di funzione) richiamata da IBM MQ quando si verificano determinati eventi.

Per utilizzare MQCB e MQCTL su un client, è necessario essere connessi a un server in cui il parametro **SHARECNV** negoziato del canale ha accettato un valore diverso da zero.

I tipi di callback che possono essere definiti sono:

consumatore di messaggi
Una funzione di callback del destinatario del messaggio viene chiamata quando un messaggio, che soddisfa i criteri di selezione specificati, è disponibile su un handle dell'oggetto.

È possibile registrare una sola funzione di callback per ciascun handle di oggetto. Se una singola coda deve essere letta con più criteri di selezione, la coda deve essere aperta più volte e una funzione consumer deve essere registrata su ciascun handle.

Gestore eventi
Il gestore eventi viene richiamato per condizioni che influenzano l'intero ambiente di callback.

La funzione viene richiamata quando si verifica una condizione di evento, ad esempio, un gestore code o una connessione in fase di arresto o di sospensione.

La funzione non viene richiamata per le condizioni specifiche di un singolo utente di messaggi, ad esempio MQRC_GET_INHIBITED; viene richiamata tuttavia se una funzione di callback non termina normalmente.

Sintassi

MQCB (*Hconn*, *Operazione*, *CallbackDesc*, *Hobj*, *MsgDesc*, *GetMsgOpzioni*, *CompCode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

Su applicazioni z/OS per CICS è possibile specificare il seguente valore speciale per *MQHC_DEF_HCONN* per utilizzare l'handle di connessione associato a questa unità di esecuzione.

Operazione
Tipo: MQLONG - input

L'operazione in fase di elaborazione sul callback definito per l'handle oggetto specificato. È necessario specificare una delle opzioni riportate di seguito. Per specificare più di un'opzione, aggiungere i valori insieme (non aggiungere la stessa costante più di una volta) o combinare i valori utilizzando l'operazione OR bit per bit (se il linguaggio di programmazione supporta le operazioni bit).

REGISTER MQOP
Definire la funzione di callback per l'handle oggetto specificato. Questa operazione definisce la funzione da chiamare e i criteri di selezione da utilizzare.

Se una funzione di callback è già definita per la gestione dell'oggetto, la definizione viene sostituita. Se viene rilevato un errore durante la sostituzione del callback, la registrazione della funzione viene annullata.

Se un callback è registrato nella stessa funzione di callback in cui è stata precedentemente annullata la registrazione, viene considerato come un'operazione di sostituzione; tutte le chiamate iniziali o finali non vengono richiamate.

È possibile utilizzare MQOP_REGISTER con MQOP_SUSPEND o MQOP_RESUME.

MQOP_DEREGISTER
Arresta l'utilizzo dei messaggi per l'handle dell'oggetto e rimuove l'handle da quelli idonei per una richiamata.

La registrazione di un callback viene annullata automaticamente se l'handle associato è chiuso.

Se MQOP_DEREGISTER viene richiamato dall'interno di un consumer e il callback ha una chiamata di arresto definita, viene richiamato al ritorno dal consumer.

Se questa operazione viene eseguita su un *Hobj* senza un consumer registrato, la chiamata viene restituita con MQRC_CALLBACK_NOT_REGISTERED.

MQOP_SOSPENSIONE
Sospende l'utilizzo dei messaggi per l'handle dell'oggetto.

Se questa operazione viene applicata a un gestore eventi, il gestore eventi non riceve gli eventi mentre è sospeso e gli eventi mancati mentre si trovano nello stato sospeso non vengono forniti all'operazione quando viene ripresa.

Mentre è sospesa, la funzionalità consumer continua a richiamare i callback di tipo controllo.

MQOP_RESUME
Riprendere l'utilizzo dei messaggi per l'handle dell'oggetto.

Se questa operazione viene applicata a un gestore eventi, il gestore eventi non riceve gli eventi mentre è sospeso e gli eventi mancati mentre si trovano nello stato sospeso non vengono forniti all'operazione quando viene ripresa.

CallbackDesc
Tipo: MQCBD - input

Questa è una struttura che identifica la funzione di callback che viene registrata dall'applicazione e le opzioni utilizzate durante la registrazione.

Consultare MQCBD per dettagli sulla struttura.

Il descrittore di callback è richiesto solo per l'opzione MQOP_REGISTER; se il descrittore non è richiesto, l'indirizzo del parametro inoltrato può essere null.

HOBJ
Tipo: MQHOBJ - input

Questo handle rappresenta l'accesso stabilito all'oggetto da cui deve essere utilizzato un messaggio. Questo è un handle restituito da una chiamata MQOPEN o MQSUB precedente (nel parametro **Hobj**).

Hobj non è richiesto quando si definisce una routine del gestore eventi (MQCBT_EVENT_HANDLER) e deve essere specificato come MQHO_NONE.

Se *Hobj* è stato restituito da una chiamata MQOPEN, la coda deve essere stata aperta con una o più delle seguenti opzioni:

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_SFOGLIA

MsgDesc
Tipo: MQMD - input

Questa struttura descrive gli attributi del messaggio richiesto e gli attributi del messaggio richiamato.

Il parametro **MsgDesc** definisce gli attributi dei messaggi richiesti dal consumer e la versione di MQMD da trasmettere al consumer del messaggio.

MsgId, *CorrelId*, *GroupId*, *MsgSeqNumbere* *Offset* in MQMD vengono utilizzati per selezionare i messaggi, in base alle opzioni specificate nel parametro **GetMsgOpts** .

Encoding e *CodedCharSetId* vengono utilizzati per la conversione del messaggio se si specifica l'opzione MQGMO_CONVERT.

Per i dettagli, consultare MQMD .

MsgDesc viene utilizzato per MQOP_REGISTER e se si richiedono valori diversi da quelli predefiniti per qualsiasi campo. *MsgDesc* non viene utilizzato per un gestore eventi.

Se il descrittore non è richiesto, l'indirizzo del parametro passato può essere null.

Notare che se più consumer sono registrati nella stessa coda con selettori che si sovrappongono, il consumer scelto per ogni messaggio non è definito.

GetMsgOpzioni
Tipo: MQGMO - input

Il parametro **GetMsgOpts** controlla il modo in cui il consumatore di messaggi riceve i messaggi. Tutte le opzioni di questo parametro hanno un significato come descritto in "MQGMO - Opzioni Get - message" a pagina 376, quando vengono utilizzate su una chiamata MQGET, tranne:

MQGMO_SET_SIGNAL
Questa opzione non è consentita.

MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT, MQGMO_MARK_ *
L'ordine dei messaggi consegnati a un utente di navigazione è dettato dalle combinazioni di queste opzioni. Le combinazioni significative sono:

MQGMO_BROWSE_FIRST
Il primo messaggio sulla coda viene consegnato ripetutamente al consumer. Ciò è utile quando il consumer utilizza in modo distruttivo il messaggio nel callback. Utilizzare questa opzione con attenzione.

MQGMO_BROWSE_SUCCESSIVO
Al consumer viene assegnato ogni messaggio sulla coda, dalla posizione corrente del cursore fino al raggiungimento della fine della coda.

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT
Il cursore viene reimpostato all'inizio della coda. Al consumer viene quindi fornito ogni messaggio fino a quando il cursore non raggiunge la fine della coda.

MQGMO_BROWSE_FIRST + MQGMO_MARK_ *
A partire dall'inizio della coda, il consumer riceve il primo messaggio non contrassegnato sulla coda, che viene quindi contrassegnato per questo consumer. Questa combinazione garantisce che il consumer possa ricevere nuovi messaggi aggiunti dietro il punto cursore corrente.

MQGMO_BROWSE_NEXT + MQGMO_MARK_ *
A partire dalla posizione del cursore, il consumer riceve il successivo messaggio non contrassegnato sulla coda, che viene quindi contrassegnato per questo consumer. Utilizzare questa combinazione con attenzione perché i messaggi possono essere aggiunti alla coda dietro la posizione del cursore corrente.

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT + MQGMO_MARK_ *
Questa combinazione non è consentita. Se utilizzata, la chiamata restituisce MQRC_OPTIONS_ERROR.

MQGMO_NO_WAIT, MQGMO_WAIT e WaitInterval
Queste opzioni controllano la modalità di richiamo del consumer.

MQGMO_NO_WAIT
Il consumer non viene mai richiamato con MQRC_NO_MSG_AVAILABLE. Il consumer viene richiamato solo per messaggi ed eventi.

MQGMO_WAIT con uno zero WaitInterval
Il codice MQRC_NO_MSG_AVAILABLE viene passato al consumer quando non ci sono messaggi disponibili e il consumer è stato avviato o è stato consegnato almeno un messaggio dall'ultimo codice di errore "nessun messaggio".

Ciò impedisce al consumer di eseguire il polling in un loop occupato quando viene specificato un intervallo di attesa zero.

MQGMO_WAIT e un WaitInterval positivo
Il consumer viene richiamato dopo l'intervallo di attesa specificato con codice motivo MQRC_NO_MSG_AVAILABLE. Questa chiamata viene effettuata indipendentemente dal fatto che i messaggi siano stati consegnati al consumer. Ciò consente all'utente di eseguire l'elaborazione di tipo heartbeat o batch.

MQGMO_WAIT e WaitInterval di MQWI_UNLIMITED
Specifica un'attesa infinita prima di restituire MQRC_NO_MSG_AVAILABLE. Il consumer non viene mai richiamato con MQRC_NO_MSG_AVAILABLE.

GetMsgOpts viene utilizzato solo per MQOP_REGISTER e se si richiedono valori diversi da quelli predefiniti per qualsiasi campo. *GetMsgOpts* non viene utilizzato per un gestore eventi.

Se *GetMsgOpts* non è richiesto, l'indirizzo del parametro passato può essere null. L'uso di questo parametro equivale a specificare MQGMO_DEFAULT insieme a MQGMO_FAIL_IF_QUIESCING.

Se nella struttura MQGMO viene fornito un handle delle proprietà del messaggio, viene fornita una copia nella struttura MQGMO che viene passata al callback del consumer. Al ritorno dalla chiamata MQCB, è possibile che l'applicazione elimini l'handle delle proprietà del messaggio.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

I codici di errore nel seguente elenco sono quelli che il gestore code può restituire per il parametro **Reason** .

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'89C') Adattatore non disponibile.

ERRORE CARICAMENTO MQRC_ADAPTER_CONV
(2133, X'855 ') Impossibile caricare i moduli dei servizi di conversione dati.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita.

ERRORE USCITA MQRC_API
(2183, X'887 ') Impossibile caricare l'uscita API.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

ERRORE MQRC_BUFFER_LENGTH
(2005, X'7D5') Parametro di lunghezza del buffer non valido.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

ERRORE MQRC_CALLBACK_LINK
(2487, X'9B7') Campo di tipo callback non corretto.

MQRC_CALLBACK_NON_REGISTRATO
(2448, X' 990 ') Impossibile annullare la registrazione, sospendere o riprendere poiché non è presente alcun callback registrato.

ERRORE DI MQRC_CALLBACK_ROUTINE_ERROR
(2486, X'9B6') È necessario specificare *CallbackFunction* o *CallbackName* , ma non entrambi.

ERRORE MQRC_CALLBACK_TIPO
(2483, X'9B3') Campo di tipo callback non corretto.

ERRORE MQRC_CBD_OPTIONS_
(2484, X'9B4') Campo di opzioni MQCBD non corretto.

MQRC_CICS_WAIT_NON RIUSCITO
(2140, X'85C') Richiesta di attesa rifiutata da CICS.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Non autorizzato per la connessione.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Connessione in fase di sospensione.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Chiusura della connessione.

ERRORE ID CORREL_MQR_
(2207, X'89F') Errore identificativo di correlazione.

ERRORE MQRC_DATA_LENGTH
(2010, X'7DA') Parametro di lunghezza dati non valido.

ERRORE MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Chiamata non valida nell'ambiente.

MQRC_FUNZIONE_NON_SUPPORTATA
(2298, X'8FA') La funzione richiesta non è disponibile nell'ambiente corrente.

MQRC_GET_INHIBITED
(2016, X'7E0') Ottiene inibiti per la coda.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Unità globali di conflitto di lavoro.

ERRORE MQRC_GMO
(2186, X'88A') Struttura delle opzioni Get - message non valida.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X' 931 ') Handle in uso per l'unità di lavoro globale.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

ERRORE MQRC_HOBJ_R
(2019, X'7E3') Handle oggetto non valido.

BROWSE INCONSIST_MQRC
(2259, X'8D3') Specifica di ricerca incongruente.

UOW MQRC_INCONSISTENT_
(2245, X'8C5') Specifica dell'unità di lavoro non congruente.

MQRC_INVALID_MSG_UNDER_CURSOR
(2246, X'8C6') Messaggio sotto il cursore non valido per il recupero.

MQRC_LOCAL_UOW_CONFLICT
(2352, X' 930 ') L'unità di lavoro globale è in conflitto con l'unità di lavoro locale.

ERRORE MQRC_MATCH_OPTIONS_
(2247, X'8C7') Opzioni di corrispondenza non valide.

ERRORE MQRC_MAX_MSG_LENGTH
(2485, X'9B4') Campo *MaxMsgLength* non corretto.

ERRORE MQRC_MD
(2026, X'7EA') Descrittore messaggio non valido.

MQRC_MODULE_ENTRY_NOT_FOUND
(2497, X'9C1') Il punto di ingresso della funzione specificato non è stato trovato nel modulo.

ID_NON_VALIDO MQRC_MODULE
(2496, X'9C0') È stato trovato un modulo, tuttavia è del tipo errato; non a 32 bit, 64 bit o una DLL (dynamic link library) valida.

MQRC_MODULE_NOT_FOUND
(2495, X'9BF') Modulo non trovato nel percorso di ricerca o non autorizzato al caricamento.

ERRORE MQRC_MSG_SEQ_NUMBER_
(2250, X'8CA') Numero di sequenza messaggio non valido.

ERRORE MQRC_MSG_TOKEN_
(2331, X'91B') L'utilizzo del token del messaggio non è valido.

MQRC_NO_MSG_AVAILABLE
(2033, X'7F1') Nessun messaggio disponibile.

MQRC_NO_MSG_UNDER_CURSOR
(2034, X'7F2') Il cursore di ricerca non è posizionato sul messaggio.

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') Coda non aperta per la ricerca.

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') Coda non aperta per l'input.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Definizione oggetto modificata dall'apertura.

MQRC_OBJECT_DAMAGED
(2101, X'835 ') Oggetto danneggiato.

ERRORE OPERAZIONE MQRC
(2206, X'89E') Codice di operazione non corretto nella chiamata API.

ERRORE MQRC_OPTIONS_
(2046, X'7FE') Opzioni non valide o non congruenti.

ERRORE MQRC_PAGESET_
(2193, X'891 ') Errore durante l'accesso al dataset della serie di pagine.

MQRC_Q_XX_ENCODE_CASE_ONE eliminato
(2052, X'804 ') La coda è stata eliminata.

MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') La coda ha un tipo di indice errato.

ERRORE MQRC_Q_MGR_NAME_
(2058, X'80A') Nome gestore code non valido o sconosciuto.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Gestore code non disponibile per la connessione.

MQRC_Q_MGR_QUIESCING
(2161, X'871 ') Gestore code in fase di sospensione.

MQRC_Q_MGR_STOPPING
(2162, X'872 ') Chiusura del gestore code.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

MQRC_SIGNAL_OUTSTANDING
(2069, X'815 ') Segnale eccezionale per questa maniglia.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Chiamata eliminata dal programma di uscita.

MQRC_SYNCPOINT_LIMITE_RAGGIUNTO
(2024, X'7E8') Non è possibile gestire ulteriori messaggi all'interno dell'unità di lavoro corrente.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818 ') Supporto punto di sincronizzazione non disponibile.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

ERRORE MQRC_UOW_ENLISTMENT_ERROR
(2354, X' 932 ') L'inserimento nell'unità di lavoro globale non è riuscito.

MQRC_UOW_MIX_NON_SUPPORTATO
(2355, X' 933 ') La miscelazione delle chiamate UOW non è supportata.

MQRC_UOW_NON_DISPONIBILE
(2255, X'8CF') Unità di lavoro non disponibile per il gestore code.

ERRORE INTERVAL_WAIT_MQRC
(2090, X'82A') Intervallo di attesa in MQGMO non valido.

MQRC_WRONG_GMO_VERSIONE
(2256, X'8D0') Versione errata di MQGMO fornita.

MQRC_WRONG_MD_VERSIONE
(2257, X'8D1') Versione non corretta di MQMD fornita.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note d'utilizzo

1. MQCB viene utilizzato per definire l'azione da richiamare per ciascun messaggio, corrispondente ai criteri specificati, disponibile sulla coda. Quando l'azione viene elaborata, il messaggio viene rimosso dalla coda e inoltrato al destinatario del messaggio definito oppure viene fornito un token del messaggio, che viene utilizzato per richiamare il messaggio.
2. MQCB può essere utilizzato per definire le routine di callback prima di iniziare l'utilizzo con MQCTL oppure può essere utilizzato dall'interno di una routine di callback.
3. Per utilizzare MQCB dall'esterno di una routine di callback, è necessario prima sospendere il consumo dei messaggi utilizzando MQCTL e riprendere il consumo in seguito.
4. MQCB non è supportato all'interno dell'adattatore IMS .

Sequenza di callback del destinatario del messaggio

È possibile configurare un consumer per richiamare il callback in punti chiave durante il ciclo di vita del consumer. Ad esempio:

- quando il consumatore è registrato per la prima volta,

- quando viene avviata la connessione,
- quando la connessione viene arrestata e
- quando la registrazione del consumer viene annullata, esplicitamente o implicitamente da un MQCLOSE.

Tabella 541. Definizioni di verbo MQCTL

Verbo	**Significato**
MQCTL (START)	Chiamata MQCTL utilizzando l'operazione MQOP_START
MQCTL (STOP)	Chiamata MQCTL che utilizza l'operazione MQOP_STOP
MQCTL (ATTESA)	Chiamata MQCTL che utilizza l'operazione MQOP_START_WAIT

Ciò consente al consumer di mantenere lo stato associato al consumer. Quando un'applicazione richiede un callback, le regole per il richiamo del consumer sono le seguenti:

REGISTRAZIONE
È sempre il primo tipo di richiamo del callback.

Viene sempre richiamato sullo stesso thread, come la chiamata MQCB (REGISTER).

INIZIO
Viene sempre richiamato in modo sincrono con il verbo MQCTL (START).

- Tutte le richiamate START vengono completate prima della restituzione del verbo MQCTL (START).

Si trova sullo stesso thread della consegna del messaggio se è richiesto THREAD_AFFINITY.

La chiamata con avvio non è garantita se, ad esempio, un callback precedente emette MQCTL (STOP) durante MQCTL (START).

ARRESTA
Non vengono consegnati ulteriori messaggi o eventi dopo questa chiamata fino a quando la connessione non viene riavviata.

Un STOP è garantito se l'applicazione è stata precedentemente richiamata per START, un messaggio o un evento.

DERISTER
È sempre l'ultimo tipo di richiamo del callback.

Verificare che l'applicazione esegua l'inizializzazione e la ripulitura basata sui thread nei callback START e STOP. È possibile eseguire l'inizializzazione e la ripulitura non basate su thread con callback REGISTER e DEREGISTER.

Non fare alcuna supposizione sulla vita e la disponibilità del thread diverso da quello che viene dichiarato. Ad esempio, non fare affidamento su un thread che rimane attivo oltre l'ultima chiamata a DEREGISTER. Allo stesso modo, quando si sceglie di non utilizzare THREAD_AFFINITY, non presupporre che il thread esista ogni volta che viene avviata la connessione.

Se l'applicazione ha particolari requisiti per le caratteristiche del thread, può sempre creare un thread di conseguenza, quindi utilizzare MQCTL (WAIT). Ciò ha l'effetto di 'donare ' il thread a IBM MQ per la consegna asincrona dei messaggi.

Utilizzo della connessione dell'utente del messaggio

È possibile configurare un consumer per richiamare il callback in punti chiave durante il ciclo di vita del consumer. Ad esempio:

- quando il consumatore è registrato per la prima volta,
- quando viene avviata la connessione,

- quando la connessione viene arrestata e
- quando la registrazione del consumer viene annullata, esplicitamente o implicitamente da un MQCLOSE.

Tabella 542. Definizioni di verbo MQCTL

Verbo	**Significato**
MQCTL (START)	Chiamata MQCTL utilizzando l'operazione MQOP_START
MQCTL (STOP)	Chiamata MQCTL che utilizza l'operazione MQOP_STOP
MQCTL (ATTESA)	Chiamata MQCTL che utilizza l'operazione MQOP_START_WAIT

Ciò consente al consumer di mantenere lo stato associato al consumer. Quando un'applicazione richiede un callback, le regole per il richiamo del consumer sono le seguenti:

REGISTRAZIONE
È sempre il primo tipo di richiamo del callback.

Viene sempre richiamato sullo stesso thread, come la chiamata MQCB (REGISTER).

INIZIO
Viene sempre richiamato in modo sincrono con il verbo MQCTL (START).

- Tutte le richiamate START vengono completate prima della restituzione del verbo MQCTL (START).

Si trova sullo stesso thread della consegna del messaggio se è richiesto THREAD_AFFINITY.

La chiamata con avvio non è garantita se, ad esempio, un callback precedente emette MQCTL (STOP) durante MQCTL (START).

ARRESTA
Non vengono consegnati ulteriori messaggi o eventi dopo questa chiamata fino a quando la connessione non viene riavviata.

Un STOP è garantito se l'applicazione è stata precedentemente richiamata per START, un messaggio o un evento.

DERISTER
È sempre l'ultimo tipo di richiamo del callback.

Verificare che l'applicazione esegua l'inizializzazione e la ripulitura basata sui thread nei callback START e STOP. È possibile eseguire l'inizializzazione e la ripulitura non basate su thread con callback REGISTER e DEREGISTER.

Non fare alcuna supposizione sulla vita e la disponibilità del thread diverso da quello che viene dichiarato. Ad esempio, non fare affidamento su un thread che rimane attivo oltre l'ultima chiamata a DEREGISTER. Allo stesso modo, quando si sceglie di non utilizzare THREAD_AFFINITY, non presupporre che il thread esista ogni volta che viene avviata la connessione.

Se l'applicazione ha particolari requisiti per le caratteristiche del thread, può sempre creare un thread di conseguenza, quindi utilizzare MQCTL (WAIT). Ciò ha l'effetto di 'donare ' il thread a IBM MQ per la consegna asincrona dei messaggi.

Richiamo C

```
MQCB (Hconn, Operation, CallbackDesc, Hobj, MsgDesc,
GetMsgOpts, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;         /* Connection handle */
MQLONG   Operation;     /* Operation being processed */
MQCBD    CallbackDesc;  /* Callback descriptor */
MQHOBJ   HObj           /* Object handle */
MQMD     MsgDesc        /* Message descriptor attributes */
MQGMO    GetMsgOpts     /* Message options */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQCB' USING HCONN, OPERATION, CBDESC, HOBJ, MSGDESC,
           GETMSGOPTS, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN    PIC S9(9) BINARY.
**   Operation
 01  OPERATION PIC S9(9) BINARY.
**   Callback Descriptior
 01  CBDESC.
     COPY CMQCBDV.
01  HOBJ     PIC S9(9) BINARY.
**   Message Descriptior
 01  MSGDESC.
     COPY CMQMDV.
**   Get Message Options
 01  GETMSGOPTS.
     COPY CMQGMOV.
**   Completion code
 01  COMPCODE PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON   PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQCB(Hconn, Operation, CallbackDesc, Hobj, MsgDesc, GetMsgOpts,
          CompCode, Reason)
```

Dichiarare i parametri come segue:

```
dcl Hconn        fixed bin(31); /* Connection handle */
dcl Operation    fixed bin(31); /* Operation */
dcl CallbackDesc like MQCBD;    /* Callback Descriptor */
dcl Hobj         fixed bin(31); /* Object Handle */
dcl MsgDesc      like MQMD;     /* Message Descriptor */
dcl GetMsgOpts   like MQGMO;    /* Get Message Options */
dcl CompCode     fixed bin(31); /* Completion code */
dcl Reason       fixed bin(31); /* Reason code qualifying CompCode */
```

MQCB_FUNCTION - Funzione di callback

La chiamata alla funzione MQCB_FUNCTION è la funzione di callback per la gestione degli eventi e il consumo asincrono dei messaggi.

La definizione della chiamata MQCB_FUNCTION viene fornita esclusivamente per descrivere i parametri passati alla funzione di callback. Il gestore code non fornisce alcun punto di ingresso denominato MQCB_FUNCTION.

La specifica della funzione effettiva da chiamare è un input per la chiamata MQCB e viene inoltrata attraverso la struttura MQCBD .

Sintassi

MQCB_FUNCTION (*Hconn*, *MsgDesc*, *GetMsgOpts*, *Buffer*, *Contesto*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente. In applicazioni z/OS per CICS la chiamata MQCONN può essere omessa e il seguente valore specificato per Hconn:

CONN MQHC_DEF

Handle di connessione predefinito.

MsgDesc
Tipo: MQMD - input

Questa struttura descrive gli attributi del messaggio richiamato.

Vedi "MQMD - Descrittore messaggi" a pagina 431 per i dettagli.

La versione di MQMD inoltrata è la stessa della chiamata MQCB che ha definito la funzione consumer.

L'indirizzo di MQMD viene passato come caratteri null se è stato utilizzato un MQGMO versione 4 per richiedere che venga restituito un handle del messaggio invece di un MQMD.

Questo è un campo di input per la funzione del consumer del messaggio; non è rilevante per una funzione del gestore eventi.

Opzioni GetMsg
Tipo: MQGMO - input

Opzioni utilizzate per controllare le azioni dell'utente del messaggio. Questo parametro contiene anche ulteriori informazioni sul messaggio restituito.

Consultare MQGMO per i dettagli.

La versione di MQGMO passata è l'ultima versione supportata.

Questo è un campo di input per la funzione del consumer del messaggio; non è rilevante per una funzione del gestore eventi.

Memorizza nel buffer
Tipo: MQBYTExBufferLength - input

Questa è l'area contenente i dati del messaggio.

Se non è disponibile alcun messaggio per questa chiamata o se il messaggio non contiene dati del messaggio, l'indirizzo del *Buffer* viene passato come null.

Questo è un campo di input per la funzione del consumer del messaggio; non è rilevante per una funzione del gestore eventi.

Contesto
Tipo: MQCBC - input/output

Questa struttura fornisce le informazioni di contesto alle funzioni di callback. Vedi "MQCBC - Contesto callback" a pagina 285 per i dettagli.

Note d'utilizzo

1. Tenere presente che se le routine di callback utilizzano servizi che potrebbero ritardare o bloccare il thread, ad esempio MQGET con attesa, potrebbe ritardare l'invio di altri callback.
2. Un'unità di lavoro separata non viene stabilita automaticamente per ogni richiamo di una routine di callback, quindi le routine possono emettere una chiamata di commit o differire il commit, fino a

quando non viene elaborato un batch logico di lavoro. Quando viene eseguito il commit del batch di lavoro, viene eseguito il commit dei messaggi per tutte le funzioni di callback richiamate dall'ultimo punto di sincronizzazione.

3. I programmi richiamati da CICS LINK o CICS START richiamano i parametri utilizzando i servizi CICS tramite oggetti denominati noti come contenitori del canale. I nomi dei contenitori sono uguali ai nomi dei parametri. Per ulteriori informazioni, consultare la documentazione CICS .
4. Le routine di callback possono emettere una chiamata MQDISC, ma non per la propria connessione. Ad esempio, se una routine di callback ha creato una connessione, può anche disconnettere la connessione.
5. Una routine di callback non deve, in generale, basarsi sul fatto di essere richiamata ogni volta dallo stesso thread. Se necessario, utilizzare MQCTLO_THREAD_AFFINITY quando viene avviata la connessione.
6. Quando una routine di callback riceve un codice di errore diverso da zero, deve intraprendere l'azione appropriata.
7. MQCB_FUNCTION non è supportato nell'adattatore IMS .

MQCLOSE - Chiudi oggetto

La chiamata MQCLOSE rinuncia all'accesso a un oggetto ed è l'inverso delle chiamate MQOPEN e MQSUB.

Sintassi

MQCLOSE (*Hconn*, *Hobj*, *Opzioni*, *CompCode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

Su z/OS per applicazioni CICS è possibile omettere la chiamata MQCONN e specificare il valore seguente per *Hconn*:

DEF_MQH_HCONN
Handle di connessione predefinito.

HOBJ
Tipo: MQHOBJ - input/output

Questo handle rappresenta l'oggetto in fase di chiusura. L'oggetto può essere di qualsiasi tipo. Il valore di *Hobj* è stato restituito da una chiamata MQOPEN precedente.

Al corretto completamento della chiamata, il gestore code imposta questo parametro su un valore che non è un handle valido per l'ambiente. Questo valore è:

MQHO_UNUSABLE_HOBJ
Gestione oggetto non utilizzabile.

Su z/OS, *Hobj* è impostato su un valore non definito.

Opzioni
Tipo: MQLONG - input

Questo parametro controlla la modalità di chiusura dell'oggetto.

Solo le sottoscrizioni e le code dinamiche permanenti possono essere chiuse in più di un modo, perché devono essere conservate o eliminate; si tratta di code con l'attributo **DefinitionType** che ha il valore MQQDT_PERMANENT_DYNAMIC (vedere l'attributo **DefinitionType** descritto in “Attributi per le code” a pagina 858). Le opzioni di chiusura sono riepiloga in questo argomento.

Le sottoscrizioni durature possono essere conservate o rimosse; queste vengono create utilizzando la chiamata MQSUB con l'opzione MQSO_DURABLE.

Quando si chiude l'handle ad una destinazione gestita (ovvero il parametro **Hobj** restituito in una chiamata MQSUB che ha utilizzato l'opzione MQSO_MANAGED), il gestore code ripulisce tutte le pubblicazioni che non sono state richiamate quando anche la sottoscrizione associata è stata rimossa. La sottoscrizione viene rimossa utilizzando l'opzione MQCO_REMOVE_SUB sul parametro **Hsub** restituito su una chiamata MQSUB. MQCO_REMOVE_SUB è il comportamento predefinito di MQCLOSE per una sottoscrizione non durevole.

Quando si chiude un handle per una destinazione non gestita, l'utente è responsabile della ripulitura della coda in cui vengono inviate le pubblicazioni. Chiudere prima la sottoscrizione utilizzando MQCO_REMOVE_SUB ed elaborare quindi i messaggi fuori dalla coda fino a quando non ne rimangono.

È necessario specificare una sola opzione tra le seguenti:

Opzioni della coda dinamica: queste opzioni controllano la modalità di chiusura delle code dinamiche permanenti.

MQCO_DELETE
La coda viene eliminata se si verifica una delle seguenti condizioni:

- Si tratta di una coda dinamica permanente, creata da una precedente chiamata MQOPEN, e non ci sono messaggi sulla coda e non ci sono richieste get o put non sottoposte a commit in attesa per la coda (sia per l'attività corrente che per qualsiasi altra attività).
- È la coda dinamica temporanea creata dalla chiamata MQOPEN che ha restituito *Hobj*. In questo caso, tutti i messaggi sulla coda vengono eliminati.

In tutti gli altri casi, incluso il caso in cui il *Hobj* è stato restituito su una chiamata MQSUB, la chiamata ha esito negativo con codice motivo MQRC_OPTION_NOT_VALID_FOR_TYPE e l'oggetto non viene eliminato.

Su z/OS, se la coda è una coda dinamica che è stata eliminata logicamente e questo è l'ultimo handle per essa, la coda viene eliminata fisicamente. Consultare “Note d'utilizzo” a pagina 673 per ulteriori dettagli.

MQCO_DELETE_PURGE
La coda viene eliminata e tutti i messaggi su di essa vengono eliminati, se si verifica una delle seguenti condizioni:

- Si tratta di una coda dinamica permanente, creata da una precedente chiamata MQOPEN, e non vi sono richieste get o put non sottoposte a commit in sospeso per la coda (per l'attività corrente o per qualsiasi altra attività).
- È la coda dinamica temporanea creata dalla chiamata MQOPEN che ha restituito *Hobj*.

In tutti gli altri casi, incluso il caso in cui il *Hobj* è stato restituito su una chiamata MQSUB, la chiamata ha esito negativo con codice motivo MQRC_OPTION_NOT_VALID_FOR_TYPE e l'oggetto non viene eliminato.

Tabella 543. Opzioni di chiusura per diversi tipi di oggetto

Tipo di oggetto o coda	MQCO_NONE	MQCO_DELETE	MQCO_DELETE_PURGE
Oggetto diverso da una coda	Conservato	Non valido	Non valido
Coda predefinita	Conservato	Non valido	Non valido
coda dinamica permanente	Conservato	Eliminato se vuoto e nessun aggiornamento in sospeso	Messaggi eliminati; coda eliminata se non vi sono aggiornamenti in sospeso

Tabella 543. Opzioni di chiusura per diversi tipi di oggetto (Continua)			
Tipo di oggetto o coda	**MQCO_NONE**	**MQCO_DELETE**	**MQCO_DELETE_PURGE**
Coda dinamica temporanea (chiamata emessa dal creatore della coda)	Eliminato	Eliminato	Eliminato
Coda dinamica temporanea (chiamata non emessa dal creatore della coda)	Conservato	Non valido	Non valido
Elenco di distribuzione	Conservato	Non valido	Non valido
Destinazione sottoscrizione gestita	Conservato	Non valido	Non valido
Elenco di distribuzione (la sottoscrizione è stata rimossa)	Messaggi eliminati; coda eliminata	Non valido	Non valido

Opzioni di chiusura della sottoscrizione: queste opzioni controllano se le sottoscrizioni durevoli vengono rimosse quando l'handle viene chiuso e se le pubblicazioni ancora in attesa di essere lette dall'applicazione vengono ripulite. Queste opzioni sono valide solo per l'utilizzo con un handle dell'oggetto restituito nel parametro **Hsub** di una chiamata MQSUB.

SUB MQCO_KEEP_
L'handle per la sottoscrizione è chiuso ma la sottoscrizione effettuata viene mantenuta. Le pubblicazioni continuano ad essere inviate alla destinazione specificata nella sottoscrizione. Questa opzione è valida solo se la sottoscrizione è stata effettuata con l'opzione MQSO_DURABLE.

MQCO_KEEP_SUB è il valore predefinito se la sottoscrizione è durevole

MQCO_REMOVE_SUB
La sottoscrizione viene rimossa e l'handle della sottoscrizione viene chiuso.

Il parametro **Hobj** della chiamata MQSUB non viene invalidato dalla chiusura del parametro **Hsub** e potrebbe continuare ad essere utilizzato per MQGET o MQCB per ricevere le pubblicazioni rimanenti. Quando viene chiuso anche il parametro **Hobj** della chiamata MQSUB, se era una destinazione gestita, tutte le pubblicazioni non richiamate vengono rimosse.

MQCO_REMOVE_SUB è il valore predefinito se la sottoscrizione non è durevole.

Il corretto completamento di MQCO_REMOVE_SUB non significa che l'azione sia stata completata. Per controllare che questa chiamata sia stata completata, consultare il passo DELETE SUB in Verifica del completamento dei comandi asincroni per le reti distribuite.

Queste opzioni di chiusura della sottoscrizione vengono riepilogate nelle seguenti tabelle.

Tabella 544. Opzioni per chiudere un handle di sottoscrizione durevole ma conservare la sottoscrizione	
Attività	**Opzione di chiusura sottoscrizione**
Conserva pubblicazioni su un handle MQOPENed	SUB MQCO_KEEP_
Rimuovi pubblicazioni su un handle MQOPENed	Azione non consentita
Conserva pubblicazioni su un handle MQSO_MANAGED	SUB MQCO_KEEP_
Rimuovi pubblicazioni su un handle MQSO_MANAGED	Azione non consentita

Per annullare la sottoscrizione, chiudendo un handle di sottoscrizione durevole e annullando la sottoscrizione o chiudendo un handle di sottoscrizione non durevole, utilizzare le seguenti opzioni di chiusura della sottoscrizione:

Tabella 545. Opzioni per annullare la sottoscrizione	
Attività	**Opzione di chiusura sottoscrizione**
Conserva pubblicazioni su un handle MQOPENed	MQCO_REMOVE_SUB
Rimuovi pubblicazioni su un handle MQOPENed	Azione non consentita
Conserva pubblicazioni su un handle MQSO_MANAGED	MQCO_REMOVE_SUB

Opzioni di lettura anticipata: le seguenti opzioni controllano cosa accade ai messaggi non persistenti che sono stati inviati al client prima che un'applicazione li richiedesse e che non sono ancora stati utilizzati dall'applicazione. Questi messaggi vengono memorizzati nel buffer di lettura anticipata del client in attesa di essere richiesti dall'applicazione e possono essere eliminati o consumati dalla coda prima del completamento di MQCLOSE.

MQCO_PRIMO
L'oggetto viene chiuso immediatamente e tutti i messaggi che sono stati inviati al client prima che un'applicazione li richiedesse vengono eliminati e non sono disponibili per essere utilizzati da alcuna applicazione. Questo è il valore predefinito.

MQCO_QUIESCE
Viene effettuata una richiesta di chiusura dell'oggetto, ma se i messaggi che sono stati inviati al client prima che un'applicazione li richiedesse, risiedono ancora nel buffer di lettura anticipata del client, la chiamata MQCLOSE restituisce un'avvertenza di MQRC_READ_AHEAD_MSGS e l'handle dell'oggetto rimane valido.

L'applicazione può quindi continuare a utilizzare l'handle dell'oggetto per richiamare i messaggi fino a quando non sono più disponibili e quindi chiudere nuovamente l'oggetto. Non vengono inviati ulteriori messaggi al client prima di un'applicazione che li richiede, la lettura anticipata è ora disattivata.

Si consiglia alle applicazioni di utilizzare MQCO_QUIESCE piuttosto che tentare di raggiungere un punto in cui non ci sono più messaggi nel buffer di lettura anticipata del client, perché potrebbe arrivare un messaggio tra l'ultima chiamata MQGET e il seguente MQCLOSE che sarebbe stato eliminato se fosse stato utilizzato MQCO_IMMEDIATE.

Se un MQCLOSE con MQCO_QUIESCE viene emesso dall'interno di una funzione di callback asincrona, si applica lo stesso comportamento dei messaggi di lettura anticipata. Se viene restituita l'avvertenza MQRC_READ_AHEAD_MSGS, la funzione di callback viene richiamata almeno un'altra volta. Quando l'ultimo messaggio rimanente che era stato letto in anticipo è stato passato alla funzione di callback, il campo ConsumerFlags di MQCBC è impostato su MQCBCF_READA_BUFFER_EMPTY.

Opzione predefinita: se non è richiesta nessuna delle opzioni descritte precedentemente, è possibile utilizzare la seguente opzione:

MQCO_NONE
Non è richiesta alcuna elaborazione di chiusura facoltativa.

Deve essere specificato per:

- Oggetti diversi dalle code
- Code predefinite
- Code dinamiche temporanee (ma solo nei casi in cui *Hobj* non è l'handle restituito dalla chiamata MQOPEN che ha creato la coda).
- Liste di distribuzione

In tutti i casi precedenti, l'oggetto viene conservato e non eliminato.

Se questa opzione viene specificata per una coda dinamica temporanea:

- La coda viene eliminata, se è stata creata dalla chiamata MQOPEN che ha restituito *Hobj* ; tutti i messaggi presenti nella coda vengono eliminati.
- In tutti gli altri casi, la coda (e gli eventuali messaggi su di essa) vengono conservati.

Se questa opzione viene specificata per una coda dinamica permanente, la coda viene conservata e non eliminata.

Su z/OS, se la coda è una coda dinamica che è stata eliminata logicamente e questo è l'ultimo handle per essa, la coda viene eliminata fisicamente. Consultare "Note d'utilizzo" a pagina 673 per ulteriori dettagli.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

I codici di errore elencati sono quelli che il gestore code può restituire per il parametro **Reason** .

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

GRUPPO_INCOMPLE_MQRC
(2241, X'8C1') Gruppo di messaggi non completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Messaggio logico non completo.

MQRC_READ_AHEAD_MSGS
(nnnn, X'xxx ') Il client ha messaggi di lettura anticipata che non sono ancora stati utilizzati dall'applicazione.

Se *CompCode* è MQCC_FAILED:

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'89C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita.

ERRORE USCITA MQRC_API
(2183, X'887 ') Impossibile caricare l'uscita API.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

MQRC_CF_NOT_AVAILABLE
(2345, X' 929 ') La funzione di accoppiamento non è disponibile.

MQRC_CF_STRUC_NON RIUSCITO
(2373, X' 945 ') La struttura della funzione di accoppiamento non è riuscita.

MQRC_CF_STRUC_IN_USO
(2346, X'92A') Struttura CF in uso.

MQRC_CICS_WAIT_NON RIUSCITO
(2140, X'85C') Richiesta di attesa rifiutata da CICS.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Non autorizzato per la connessione.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Chiusura della connessione.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') Db2 sottosistema non disponibile.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

ERRORE MQRC_HOBJ_R
(2019, X'7E3') Handle oggetto non valido.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

MQRC_OBJECT_DAMAGED
(2101, X'835 ') Oggetto danneggiato.

MQRC_OPTION_NOT_VALID_FOR_TYPE
(2045, X'7FD') Su una chiamata MQOPEN o MQCLOSE: opzione non valida per il tipo di oggetto.

ERRORE MQRC_OPTIONS_
(2046, X'7FE') Opzioni non valide o non congruenti.

ERRORE MQRC_PAGESET_
(2193, X'891 ') Errore durante l'accesso al dataset della serie di pagine.

ERRORE MQRC_Q_MGR_NAME_
(2058, X'80A') Nome gestore code non valido o sconosciuto.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Gestore code non disponibile per la connessione.

MQRC_Q_MGR_STOPPING
(2162, X'872 ') Chiusura del gestore code.

MQR_Q_NO_EMPTY
(2055, X'807 ') La coda contiene uno o più messaggi o richieste put o get senza commit.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

ERRORE MQRC_SECURITY_ERROR
(2063, X'80F') Si è verificato un errore di sicurezza.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Chiamata eliminata dal programma di uscita.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note d'utilizzo

1. Quando un'applicazione emette la chiamata MQDISC o termina normalmente o in modo anomalo, tutti gli oggetti aperti dall'applicazione e ancora aperti vengono chiusi automaticamente con l'opzione MQCO_NONE.
2. I seguenti punti si applicano se l'oggetto da chiudere è una *coda*:
   - Se le operazioni sulla coda vengono eseguite come parte di un'unità di lavoro, la coda può essere chiusa prima o dopo che si verifichi il punto di sincronizzazione senza influenzare il risultato del punto di sincronizzazione. Se la coda viene attivata, l'esecuzione di un rollback prima della chiusura della coda può causare l'emissione di un messaggio trigger. Per ulteriori informazioni sui messaggi trigger, consultare Proprietà dei messaggi trigger.
   - Se la coda è stata aperta con l'opzione MQOO_BROWSE, il cursore di ricerca viene eliminato. Se la coda viene quindi riaperta con l'opzione MQOO_BROWSE, viene creato un nuovo cursore di ricerca (vedere MQOO_BROWSE).
   - Se un messaggio è attualmente bloccato per questo handle al momento della chiamata MQCLOSE, il blocco viene rilasciato (consultare MQGMO_LOCK).
   - Su z/OS, se è presente una richiesta MQGET con l'opzione MQGMO_SET_SIGNAL in sospeso rispetto all'handle della coda in fase di chiusura, la richiesta viene annullata (consultare MQGMO_SET_SIGNAL). Le richieste di segnale per la stessa coda ma depositate su handle differenti (*Hobj*) non vengono influenzate (a meno che non venga eliminata una coda dinamica, nel qual caso vengono annullate).
3. I seguenti punti si applicano se l'oggetto che si sta chiudendo è una *coda dinamica* (permanente o temporanea):
   - Per una coda dinamica, è possibile specificare le opzioni MQCO_DELETE e MQCO_DELETE_PURGE indipendentemente dalle opzioni specificate nella chiamata MQOPEN corrispondente.
   - Quando una coda dinamica viene eliminata, tutte le chiamate MQGET con l'opzione MQGMO_WAIT in sospeso rispetto alla coda vengono annullate e viene restituito il codice motivo MQRC_Q_DELETED. Vedere MQGMO_WAIT.

     Anche se le applicazioni non possono accedere a una coda eliminata, la coda non viene rimossa dal sistema e le risorse associate non vengono liberate, fino a quando tutti gli handle che fanno riferimento alla coda non sono stati chiusi e tutte le unità di lavoro che influiscono sulla coda non sono state sottoposte a commit o a backout.

     Su z/OS, una coda che è stata eliminata logicamente ma non ancora rimossa dal sistema impedisce la creazione di una nuova coda con lo stesso nome della coda eliminata; in questo caso, la chiamata MQOPEN ha esito negativo con codice motivo MQRC_NAME_IN_USE. Inoltre, una coda di questo tipo può ancora essere visualizzata utilizzando i comandi MQSC, anche se non è possibile accedervi dalle applicazioni.
   - Quando una coda dinamica permanente viene eliminata, se l'handle *Hobj* specificato nella chiamata MQCLOSE non è quello restituito dalla chiamata MQOPEN che ha creato la coda, viene eseguito un verifica che l'identificativo utente utilizzato per convalidare la chiamata MQOPEN sia autorizzato a eliminare la coda. Se l'opzione MQOO_ALTERNATE_USER_AUTHORITY è stata specificata nella chiamata MQOPEN, l'identificativo utente controllato è *AlternateUserId*.

     Questo controllo non viene eseguito se:

     – L'handle specificato è quello restituito dalla chiamata MQOPEN che ha creato la coda.

     – La coda che si sta eliminando è una coda dinamica temporanea.
   - Quando una coda dinamica temporanea viene chiusa, se l'handle *Hobj* specificato nella chiamata MQCLOSE è quello restituito dalla chiamata MQOPEN che ha creato la coda, la coda viene eliminata. Ciò si verifica indipendentemente dalle opzioni di chiusura specificate nella chiamata MQCLOSE. Se ci sono messaggi nella coda, vengono eliminati; non viene generato alcun messaggio di report.

     Se ci sono unità di lavoro non sottoposte a commit che influiscono sulla coda, la coda e i relativi messaggi vengono ancora eliminati, ma le unità di lavoro non hanno esito negativo. Tuttavia, come

descritto in precedenza, le risorse associate alle unità di lavoro non vengono liberate fino a quando non viene eseguito il commit o il backout di ciascuna unità di lavoro.

4. I seguenti punti si applicano se l'oggetto da chiudere è un *elenco di distribuzione*:

   - L'unica opzione di chiusura valida per un elenco di distribuzione è MQCO_NONE; la chiamata ha esito negativo con codice motivo MQRC_OPTIONS_ERROR o MQRC_OPTION_NOT_VALID_FOR_TYPE se sono specificate altre opzioni.
   - Quando un elenco di distribuzione viene chiuso, i singoli codici di completamento e i codici di errore non vengono restituiti per le code nell'elenco; solo i parametri **CompCode** e **Reason** della chiamata sono disponibili per scopi diagnostici.

     Se si verifica un errore durante la chiusura di una delle code, il gestore code continua l'elaborazione e tenta di chiudere le code rimanenti nell'elenco di distribuzione. I parametri **CompCode** e **Reason** della chiamata sono impostati per restituire informazioni che descrivono l'errore. È possibile che il codice di completamento sia MQCC_FAILED, anche se la maggior parte delle code è stata chiusa correttamente. La coda che ha rilevato l'errore non è stata identificata.

     Se si verifica un errore su più di una coda, non è definito quale errore viene riportato nei parametri **CompCode** e **Reason** .

Richiamo C

```
MQCLOSE (Hconn, &Hobj, Options, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;     /* Connection handle */
MQHOBJ   Hobj;      /* Object handle */
MQLONG   Options;   /* Options that control the action of MQCLOSE */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
     CALL 'MQCLOSE' USING HCONN, HOBJ, OPTIONS, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Object handle
 01  HOBJ      PIC S9(9) BINARY.
**   Options that control the action of MQCLOSE
 01  OPTIONS   PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQCLOSE (Hconn, Hobj, Options, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl Hobj      fixed bin(31);  /* Object handle */
dcl Options   fixed bin(31);  /* Options that control the action of
                                 MQCLOSE */
```

```
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQCLOSE,(HCONN,HOBJ,OPTIONS,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN     DS  F  Connection handle
HOBJ      DS  F  Object handle
OPTIONS   DS  F  Options that control the action of MQCLOSE
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Richiamo Visual Basic

```
MQCLOSE Hconn, Hobj, Options, CompCode, Reason
```

Dichiarare i parametri come segue:

```
Dim Hconn    As Long 'Connection handle'
Dim Hobj     As Long 'Object handle'
Dim Options  As Long 'Options that control the action of MQCLOSE'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQCMIT - Commit modifiche

La chiamata MQCMIT indica al gestore code che l'applicazione ha raggiunto un punto di sincronizzazione e che tutti i richiami e gli inserimenti di messaggi che si sono verificati dall'ultimo punto di sincronizzazione devono essere resi permanenti.

I messaggi inseriti come parte di un'unità di lavoro vengono resi disponibili ad altre applicazioni; i messaggi richiamati come parte di un'unità di lavoro vengono eliminati.

- z/OS Su z/OS, la chiamata viene utilizzata solo dai programmi batch (inclusi i programmi DL/I batch IMS).

Sintassi

MQCMIT (*Hconn*, *CompCode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

I codici di errore elencati sono quelli che il gestore code può restituire per il parametro **Reason** .

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

MQRC_BACK_OUT
(2003, X'7D3') Unità di lavoro ripristinata.

MQRC_OUTCOME_PENDING
(2124, X'84C') Il risultato dell'operazione di commit è in sospeso.

Se *CompCode* è MQCC_FAILED:

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

MQRC_CALL_INTERROTTO
(2549, X'9F5') MQPUT o MQCMIT è stato interrotto e l'elaborazione della riconnessione non può ristabilire un risultato definito.

MQRC_CF_STRUC_IN_USO
(2346, X'92A') Struttura CF in uso.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

ERRORE MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Chiamata non valida nell'ambiente.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

MQRC_OBJECT_DAMAGED
(2101, X'835 ') Oggetto danneggiato.

MQRC_OUTCOME_MIXED
(2123, X'84B') Il risultato dell'operazione di commit o di backout è misto.

MQRC_Q_MGR_STOPPING
(2162, X'872 ') Chiusura del gestore code.

MQRC_RECONNECT_NON RIUSCITO
(2548, X'9F4') Dopo la riconnessione, si è verificato un errore durante la reintegrazione delle maniglie per una connessione ricollegabile.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890 ') Il supporto di memoria esterna è pieno.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note d'utilizzo

1. Utilizzare questa chiamata solo quando il gestore code stesso coordina l'unità di lavoro. Questo può essere:
   - Un'unità di lavoro locale, in cui le modifiche influenzano solo le risorse IBM MQ .
   - Un'unità di lavoro globale, in cui le modifiche possono influire sulle risorse appartenenti ad altri gestori risorse, nonché sulle risorse IBM MQ .

   Per ulteriori dettagli sulle unità di lavoro locali e globali, consultare “MQBEGIN - Inizio unità di lavoro” a pagina 649.
2. In ambienti in cui il gestore code non coordina l'unità di lavoro, è necessario utilizzare la chiamata di commit appropriata al posto di MQCMIT. L'ambiente potrebbe anche supportare un commit implicito causato dalla chiusura normale dell'applicazione.
   - Su z/OS, utilizzare le seguenti chiamate:
     - I programmi batch (inclusi i programmi IMS batch DL/I) possono utilizzare la chiamata MQCMIT se l'unità di lavoro influisce solo sulle risorse IBM MQ . Tuttavia, se l'unità di lavoro influisce sia sulle risorse IBM MQ che sulle risorse appartenenti ad altri gestori risorse (ad esempio, Db2), utilizzare la chiamata SRRCMIT fornita da RRS (Recoverable Resource Service) z/OS . La chiamata SRRCMIT esegue il commit delle modifiche alle risorse appartenenti ai gestori risorse che sono state abilitate per il coordinamento RRS.
     - Le applicazioni CICS devono utilizzare il comando `EXEC CICS SYNCPOINT` per eseguire esplicitamente il commit dell'unità di lavoro. In alternativa, la chiusura della transazione comporta un commit implicito dell'unità di lavoro. Non è possibile utilizzare la chiamata MQCMIT per applicazioni CICS .
     - Le applicazioni IMS (diverse dai programmi DL/I batch) devono utilizzare le chiamate IMS come `GU` e `CHKP` per eseguire il commit dell'unità di lavoro. La chiamata MQCM non può essere utilizzata per applicazioni IMS (diverse dai programmi DL/I batch).
   - Su IBM i, utilizzare questa chiamata per le unità di lavoro locali coordinate dal gestore code. Ciò significa che una definizione di commit non deve esistere a livello di lavoro, ovvero il comando `STRCMTCTL` con il parametro **`CMTSCOPE(*JOB)`** non deve essere stato immesso per il lavoro.
3. Se un'applicazione termina con modifiche non sottoposte a commit in un'unità di lavoro, la disposizione di tali modifiche dipende dal fatto che l'applicazione termini normalmente o in modo anomalo. Consultare Note sull'utilizzo di MQDISC per ulteriori dettagli.
4. Quando un'applicazione inserisce o richiama i messaggi in gruppi o segmenti di messaggi logici, il gestore code conserva le informazioni relative al gruppo di messaggi e al messaggio logico per le ultime chiamate MQPUT e MQGET riuscite. Queste informazioni sono associate all'handle della coda e includono:
   - I valore dei campi *`GroupId`*, *`MsgSeqNumber`*, *`Offset`*e *`MsgFlags`* in MQMD.
   - Se il messaggio fa parte di un'unità di lavoro.
   - Per la chiamata MQPUT: se il messaggio è persistente o non persistente.

   Quando viene eseguito il commit di un'unità di lavoro, il gestore code conserva le informazioni sul gruppo e sul segmento e l'applicazione può continuare a inserire o richiamare i messaggi nel gruppo di messaggi corrente o nel messaggio logico.

   La conservazione delle informazioni sul gruppo e sul segmento quando viene eseguito il commit di un'unità di lavoro consente all'applicazione di distribuire un gruppo di messaggi di grandi dimensioni

o un messaggio logico di grandi dimensioni costituito da molti segmenti su diverse unità di lavoro. L'utilizzo di diverse unità di lavoro è vantaggioso se il gestore code locale ha solo una memoria di coda limitata. Tuttavia, l'applicazione deve conservare informazioni sufficienti per riavviare l'inserimento o il richiamo dei messaggi nel punto corretto se si verifica un malfunzionamento del sistema. Per i dettagli su come riavviare il sistema nel punto corretto dopo un errore di sistema, consultare MQPMO_LOGICAL_ORDER e MQGMO_LOGICAL_ORDER.

Le note di uso rimanenti si applicano solo quando il gestore code coordina le unità di lavoro:

5. Un'unità di lavoro ha lo stesso ambito di un handle di connessione; tutte le chiamate IBM MQ che interessano una particolare unità di lavoro devono essere eseguite utilizzando lo stesso handle di connessione. Le chiamate emesse utilizzando un handle di collegamento differente (ad esempio, le chiamate emesse da un'altra applicazione) influenzano un'unità di lavoro diversa. Consultare il parametro **Hconn** descritto in MQCONN per informazioni sull'ambito degli handle di connessione.
6. Solo i messaggi inseriti o richiamati come parte dell'unità di lavoro corrente vengono influenzati da questa chiamata.
7. Un'applicazione di lunga durata che emette chiamate MQGET, MQPUT o MQPUT1 all'interno di un'unità di lavoro, ma che non emette mai una chiamata di commit o di back-out, può riempire le code con messaggi che non sono disponibili per altre applicazioni. Per evitare ciò, l'amministratore deve impostare l'attributo del gestore code **MaxUncommittedMsgs** su un valore sufficientemente basso per evitare che le applicazioni runaway riempiano le code, ma sufficientemente alto per consentire alle applicazioni di messaggistica previste di funzionare correttamente.
8. ALW Su sistemi AIX, Linux, and Windows , se il parametro **Reason** è MQRC_CONNECTION_BROKEN (con un *CompCode* di MQCC_FAILED) o MQRC_UNEXPECTED_ERROR è possibile che il commit dell'unità di lavoro sia stato eseguito correttamente.

Richiamo C

```
MQCMIT (Hconn, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQCMIT' USING HCONN, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQCMIT (Hconn, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQCMIT,(HCONN,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Richiamo Visual Basic

```
MQCMIT Hconn, CompCode, Reason
```

Dichiarare i parametri come segue:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQCONN - Connetti gestore code

La chiamata MQCONN collega un programma applicativo a un gestore code.

Fornisce un handle di connessione del gestore code, che l'applicazione utilizza nelle successive chiamate di accodamento messaggi.

- Su z/OS, le applicazioni CICS non devono emettere questa chiamata. Queste applicazioni vengono connesse automaticamente al gestore code a cui è connesso il sistema CICS . Tuttavia, le chiamate MQCONN e MQDISC sono ancora accettate dalle applicazioni CICS .
- Su IBM i, le applicazioni devono utilizzare la chiamata MQCONN o MQCONNX per connettersi al gestore code e la chiamata MQDISC per disconnettersi dal gestore code.

Non è possibile stabilire una connessione client solo su un'installazione server e non è possibile stabilire una connessione locale solo su un'installazione client.

Sintassi

MQCONN (*QMgrName*, *Hconn*, *CompCode*, *Reason*)

Parametri

QMgrName

Tipo: MQCHAR48 - input

Questo è il nome del gestore code a cui l'applicazione desidera connettersi. Il nome può contenere i seguenti caratteri:

- Caratteri alfabetici maiuscoli (da A a Z)
- Caratteri alfabetici minuscoli (da a a z)
- Cifre numeriche (da 0 a 9)
- Punto (.), barra (/), sottolineatura (_), percentuale (%)

Il nome non deve contenere spazi iniziali o intermedi, ma può contenere spazi finali. Un carattere null può essere utilizzato per indicare la fine dei dati significativi nel nome; il valore null e i caratteri che lo seguono vengono trattati come spazi vuoti. Le seguenti limitazioni si applicano agli ambienti indicati:

- Sui sistemi che utilizzano EBCDIC Katakana, non è possibile utilizzare caratteri minuscoli.
- Su z/OS, i nomi che iniziano o terminano con un carattere di sottolineatura non possono essere elaborati dalle operazioni e dai pannelli di controllo. Per questo motivo, evitare tali nomi.
- Su IBM i, racchiudere i nomi contenenti caratteri minuscoli, barre o percentuale tra virgolette quando specificati nei comandi. non specificare queste virgolette nel parametro **QMgrName** .

Se il nome è composto interamente da spazi vuoti, viene utilizzato il nome del gestore code *predefinito* . Tuttavia, notare l'utilizzo di nomi di gestori code vuoti descritti nella sezione sulle applicazioni IBM MQ MQI client .

Il nome specificato per *QMgrName* deve essere il nome di un gestore code *collegabile* oppure, se si stanno utilizzando i gruppi di gestori code, il nome del gruppo di gestori code.

Su z/OS, i gestori code a cui è possibile connettersi sono determinati dall'ambiente:

- Per CICS, è possibile utilizzare solo il gestore code a cui è connesso il sistema CICS . Il parametro **QMgrName** deve essere ancora specificato, ma il suo valore viene ignorato; i caratteri vuoti sono un'opzione adatta.
- Per IMS, solo i gestori code elencati nella tabella di definizione del sottosistema (CSQQDEFV), *e* elencati nella tabella SSM in IMS, sono collegabili (consultare la nota sull'uso 6).
- Per z/OS batch e TSO, sono collegabili solo i gestori code che risiedono sullo stesso sistema dell'applicazione (vedere la nota di utilizzo 6).

Gruppi di condivisione code:Sui sistemi in cui esistono diversi gestori code e sono configurati per formare un gruppo di condivisione code, il nome del gruppo di condivisione code può essere specificato per *QMgrName* invece del nome di un gestore code. Ciò consente all'applicazione di connettersi a *qualsiasi* gestore code disponibile nel gruppo di condivisione code e che si trova sulla stessa immagine z/OS dell'applicazione. Il sistema può essere configurato anche in modo che l'uso di un *QMgrName* vuoto si connetta al gruppo di condivisione code invece che al gestore code predefinito.

Se *QMgrName* specifica il nome del gruppo di condivisione code, ma è presente anche un gestore code con tale nome sul sistema, la connessione viene effettuata al secondo in preferenza al primo. Solo se la connessione non riesce, viene tentata la connessione a uno dei gestori code nel gruppo di condivisione code.

Se la connessione è riuscita, è possibile utilizzare l'handle restituito dalla chiamata MQCONN o MQCONNX per accedere a *tutte* le risorse (condivise e non condivise) che appartengono al gestore code a cui è stata effettuata la connessione. L'accesso a queste risorse è soggetto ai tipici controlli di autorizzazione.

Se l'applicazione emette due chiamate MQCONN o MQCONNX per stabilire connessioni simultanee e una o entrambe le chiamate specifica il nome del gruppo di condivisione code, la seconda chiamata restituisce il codice di completamento MQCC_WARNING e il codice motivo MQRC_ALREADY_CONNECTED quando si connette allo stesso gestore code della prima chiamata.

I gruppi di condivisione code sono supportati solo su z/OS. La connessione ad un gruppo di condivisione code è supportata solo negli ambienti batch, RRS batch CICSe TSO. Per CICS, è possibile utilizzare solo il gruppo di condivisione code a cui è connesso il sistema CICS . È comunque necessario specificare il parametro **QMgrName** , ma il relativo valore viene ignorato; i caratteri vuoti sono un'opzione adatta.

Attenzione: IMS non è in grado di collegarsi a un gruppo di condivisione code.

IBM MQ MQI client applications: per le applicazioni IBM MQ MQI client , viene tentata una connessione per ogni definizione di canale di connessione client con il nome gestore code specificato, fino a quando non ne viene eseguita una. Il gestore code, tuttavia, deve avere lo stesso nome del nome specificato. Se viene specificato un nome completamente vuoto, ogni canale di connessione

client con un nome gestore code completamente vuoto viene tentato fino a quando non viene eseguito correttamente; in questo caso, non viene eseguito alcun controllo rispetto al nome effettivo del gestore code.

Le applicazioni client di IBM MQ non sono supportate in z/OS, ma z/OS può agire come un server IBM MQ , a cui le applicazioni client IBM MQ possono connettersi.

IBM MQ MQI client gruppi di gestori code: se il nome specificato inizia con un asterisco (*), il gestore code a cui viene effettuata la connessione potrebbe avere un nome diverso da quello specificato dall'applicazione. Il nome specificato (senza l'asterisco) definisce un *gruppo* di gestori code idonei per la connessione. L'implementazione ne seleziona una dal gruppo provando ciascuna a turno fino a quando non ne trova una a cui è possibile stabilire una connessione. L'ordine in cui vengono tentate le connessioni è influenzato dal peso del canale client e dai valori di affinità di connessione dei canali candidati. Se nessuno dei gestori code del gruppo è disponibile per la connessione, la chiamata non riesce. Ogni gestore code viene tentato una sola volta. Se per il nome viene specificato un asterisco, viene utilizzato un gruppo di gestori code predefinito definito dall'implementazione.

I gruppi di gestori code sono supportati solo per le applicazioni in esecuzione in un ambiente client MQ; la chiamata ha esito negativo se un'applicazione non client specifica un nome gestore code che inizia con un asterisco. Un gruppo viene definito fornendo diverse definizioni di canale di connessione client con lo stesso nome gestore code (il nome specificato senza l'asterisco), per comunicare con ciascuno dei gestori code del gruppo. Il gruppo predefinito viene definito fornendo una o più definizioni di canale di connessione client, ciascuna con un nome gestore code vuoto (specificare un nome tutto vuoto ha quindi lo stesso effetto di specificare un singolo asterisco per il nome di un'applicazione client).

Dopo la connessione a un gestore code di un gruppo, un'applicazione può specificare spazi vuoti nel modo tipico nei campi del nome del gestore code nei descrittori di messaggi e oggetti per indicare il nome del gestore code a cui l'applicazione si è connessa (il *gestore code locale*). Se l'applicazione deve conoscere questo nome, utilizzare la chiamata MQINQ per analizzare l'attributo del gestore code **QMgrName** .

Il prefisso di un asterisco al nome della connessione implica che l'applicazione non dipende dalla connessione a un determinato gestore code nel gruppo. Le applicazioni adatte sono:

- Applicazioni che immettono messaggi ma non li ricevono.
- Le applicazioni che immettono i messaggi di richiesta e quindi ricevono i messaggi di risposta da una coda *dinamica temporanea* .

Le applicazioni non adatte sono quelle che devono richiamare i messaggi da una coda particolare in un determinato gestore code; tali applicazioni non devono anteporre un asterisco al nome.

Se si specifica un asterisco, la lunghezza massima del resto del nome è 47 caratteri.

La lunghezza di questo parametro è fornita da MQ_Q_MGR_NAME_LENGTH.

Hconn

Tipo: MQHCONN - output

Questo handle rappresenta la connessione al gestore code. Specificarlo su tutte le chiamate di accodamento messaggi successive emesse dall'applicazione. Cessa di essere valida quando viene emessa la chiamata MQDISC o quando termina l'unità di elaborazione che definisce l'ambito dell'handle.

IBM MQ ora fornisce la libreria mqm con pacchetti client e pacchetti server. Ciò significa che quando viene effettuata una chiamata MQI che si trova nella libreria mqm, il tipo di connessione viene controllato per verificare se si tratta di una connessione client o server e quindi viene effettuata la chiamata sottostante corretta. Pertanto, un'uscita passata a *Hconn* può ora essere collegata alla libreria mqm, ma utilizzata su un'installazione client.

Gestisci ambito: L'ambito dell'handle restituito dipende dalla chiamata utilizzata per la connessione al gestore code (MQCONN o MQCONNX). Se la chiamata utilizzata è MQCONNX, l'ambito dell'handle dipende anche dall'opzione MQCNO_HANDLE_SHARE_ * specificata nel campo *Options* della struttura MQCNO.

- Se la chiamata è MQCONN o viene specificata l'opzione MQCNO_HANDLE_SHARE_NONE, l'handle restituito è un handle *non condiviso* .

  L'ambito di un handle non condiviso è l'unità di elaborazione parallela più piccola supportata dalla piattaforma su cui è in esecuzione l'applicazione (per i dettagli, consultare Tabella 546 a pagina 682); l'handle non è valido all'esterno dell'unità di elaborazione parallela da cui è stata emessa la chiamata.
- Se si specifica l'opzione MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK, l'handle restituito è un handle *condiviso* .

  L'ambito di un handle condiviso è il processo che possiede il thread da cui è stata emessa la chiamata; l'handle può essere utilizzato da qualsiasi thread che appartiene a tale processo. Non tutte le piattaforme supportano i thread.
- Se la chiamata MQCONN o MQCONNX non riesce con codice di completamento uguale a MQCC_FAILED, il valore Hconn non è definito.

Tabella 546. Ambito degli handle non condivisi su varie piattaforme

Piattaforma	**Ambito dell'handle non condiviso**
z/OS	• CICS: l'attività CICS • IMS: l'attività, fino al punto di sincronizzazione successivo (escludendo le attività secondarie dell'attività) • z/OS batch e TSO: l'attività (escludendo le attività secondarie dell'attività)
IBM i	Lavoro
AIX and Linux	Sottoprocesso
Applicazioni Windows a 32 bit	Sottoprocesso
Applicazioni Windows a 64 bit	Sottoprocesso

Su applicazioni z/OS per CICS , il valore restituito è:

DEF_MQH_HCONN
Handle di connessione predefinito.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

MQRC_ALREADY_CONNECTED
(2002, X'7D2') Applicazione già connessa.

ERRORE DI CARICAMENTO MQRC_CLUSTER_EXIT_LOAD
(2267, X'8DB') Impossibile caricare l'uscita del carico di lavoro del cluster.

MQRC_SSL_ALREADY_INITIALIZED
(2391, X' 957 ') SSL già inizializzato.

Se *CompCode* è MQCC_FAILED:

ERRORE CARICAMENTO MQRC_ADAPTER_CONN_LOAD
(2129, X'851 ') Impossibile caricare il modulo di collegamento dell'adattatore.

ERRORE DEFS MQRC_ADAPTER_
(2131, X'853 ') Modulo definizione sottosistema adattatore non valido.

ERRORE CARICAMENTO MQRC_ADAPTER_DEFS_
(2132, X'854 ') Impossibile caricare il modulo di definizione del sottosistema dell'adattatore.

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'89C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

SCARSE_ARCHIVIAZIONE_ADATTATORE_MQRC_
(2127, X'84F') Memoria insufficiente per l'adattatore.

MQRC_ANOTHER_Q_MGR_CONNECTED
(2103, X'837 ') Un altro gestore code è già connesso.

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita.

MQRC_API_EXIT_INIT_ERROR
(2375, X' 947 ') Inizializzazione dell'uscita API non riuscita.

ERRORE USCITA MQRC_API
(2376, X' 948 ') Terminazione uscita API non riuscita.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

ERRORE MQRC_BUFFER_LENGTH
(2005, X'7D5') Parametro di lunghezza del buffer non valido.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

MQRC_CONN_ID_IN_USE
(2160, X'870 ') Identificativo di collegamento già in uso.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

ERRORE MQRC_CONNECTION_
(2273, X'8E1') Errore durante l'elaborazione della chiamata MQCONN.

MQRC_CONNECTION_NON_DISPONIBILE
(2568, X'A08') Si verifica su una chiamata MQCONN o MQCONNX quando il gestore code non è in grado di fornire una connessione del tipo di connessione richiesto sull'installazione corrente. Non è possibile effettuare una connessione client solo su un'installazione server. Non è possibile stabilire una connessione locale solo su un client.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Connessione in fase di sospensione.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Chiusura della connessione.

MQRC_CRYPTO_HARDWARE_ERROR
(2382, X'94E') Errore di configurazione hardware crittografico.

MQRC_DUPLICATE_RECOV_COORD
(2163, X'873 ') Il coordinatore di recupero esiste.

ERRORE MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Chiamata non valida nell'ambiente.

Inoltre, sulla chiamata MQCONNX, passando il blocco di controllo “MQCSP - Parametri di sicurezza” a pagina 340 da un'applicazione CICS o IMS .

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

MQRC_HOST_NON_DISPONIBILE
(2538, X'9EA') È stata emessa una chiamata MQCONN da un client per connettersi a un gestore code, ma il tentativo di assegnare una conversazione al sistema remoto non è riuscito.

MQRC_INSTALLATION_MISMATCH
(2583, X'A17') Mancata corrispondenza tra l'installazione del gestore code e la libreria selezionata.

MQRC_KEY_REPOSITORY_ERROR
(2381, X'94D') Repository chiavi non valido.

MQRC_MAX_CONNS_LIMIT_REACHED
(2025, X'7E9') Numero massimo di connessioni raggiunto.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

MQRC_OPEN_NON RIUSCITO
(2137, X'859 ') Oggetto non aperto correttamente.

ERRORE MQRC_Q_MGR_NAME_
(2058, X'80A') Nome gestore code non valido o sconosciuto.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Gestore code non disponibile per la connessione.

MQRC_Q_MGR_QUIESCING
(2161, X'871 ') Gestore code in fase di sospensione.

MQRC_Q_MGR_STOPPING
(2162, X'872 ') Chiusura del gestore code.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

ERRORE MQRC_SECURITY_ERROR
(2063, X'80F') Si è verificato un errore di sicurezza.

ERRORE MQRC_SSL_INITIALIZATION_ERROR
(2393, X' 959 ') Errore di inizializzazione SSL.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note d'utilizzo

1. Il gestore code a cui viene effettuata una connessione utilizzando la chiamata MQCONN è denominato *gestore code locale*.
2. Le code di proprietà del gestore code locale vengono visualizzate dall'applicazione come code locali. È possibile inserire e richiamare messaggi da queste code.

   Le code condivise di proprietà del gruppo di condivisione code a cui appartiene il gestore code locale appaiono all'applicazione come code locali. È possibile inserire e richiamare messaggi da queste code.

Le code di proprietà di gestori code remoti vengono visualizzate come code remote. È possibile inserire messaggi in queste code, ma non ottenere messaggi da queste code.

3. Se il gestore code ha esito negativo mentre un'applicazione è in esecuzione, l'applicazione deve emettere nuovamente la chiamata MQCONN per ottenere un nuovo handle di connessione da utilizzare nelle chiamate IBM MQ successive. L'applicazione può emettere la chiamata MQCONN periodicamente fino a quando la chiamata ha esito positivo.

   Se un'applicazione non è sicura se è connessa al gestore code, può tranquillamente emettere una chiamata MQCONN per ottenere un handle di connessione. Se l'applicazione è già connessa, l'handle restituito è uguale a quello restituito dalla chiamata MQCONN precedente, ma con codice di completamento MQCC_WARNING e codice motivo MQRC_ALREADY_CONNECTED.
4. Quando l'applicazione ha finito di utilizzare le chiamate IBM MQ , deve utilizzare la chiamata MQDISC per disconnettersi dal gestore code.
5. Se la chiamata MQCONN ha esito negativo con codice di completamento uguale a MQCC_FAILED, il valore Hconn non è definito.
6. Su z/OS:

   - Le applicazioni batch, TSO e IMS devono emettere la chiamata MQCONN per utilizzare le altre chiamate IBM MQ . Queste applicazioni possono connettersi a più di un gestore code contemporaneamente.

     Se il gestore code ha esito negativo, l'applicazione deve emettere di nuovo la chiamata dopo che il gestore code è stato riavviato per ottenere un nuovo handle di connessione.

     Sebbene le applicazioni IMS possano emettere la chiamata MQCONN ripetutamente, anche quando sono già connesse, ciò non è consigliato per i programmi di elaborazione dei messaggi in linea (MPP).
   - Le applicazioni CICS non devono emettere la chiamata MQCONN per utilizzare le altre chiamate IBM MQ , ma possono farlo se lo desiderano; vengono accettate sia la chiamata MQCONN che la chiamata MQDISC. Tuttavia, non è possibile connettersi a più di un gestore code contemporaneamente.

     Se il gestore code ha esito negativo, queste applicazioni vengono riconnesse automaticamente quando il gestore code viene riavviato, quindi non è necessario emettere la chiamata MQCONN.
7. Su z/OS, per definire i gestori code disponibili:

   - Per le applicazioni batch, i programmatori di sistema possono utilizzare la macro CSQBDEF per creare un modulo (CSQBDEFV) che definisce il nome del gestore code predefinito o il nome del gruppo di condivisione code.
   - Per applicazioni IMS , i programmatori di sistema possono utilizzare la macro CSQQDEFX per creare un modulo (CSQQDEFV) che definisce i nomi dei gestori code disponibili e specifica il gestore code predefinito.

     Inoltre, ogni gestore code deve essere definito per la control region IMS e per ogni regione dipendente che accede a tale gestore code. A tale scopo, è necessario creare un membro del sottosistema in IMS.PROCLIB e identificare il membro del sottosistema per le regioni IMS applicabili. Se un'applicazione tenta di connettersi a un gestore code che non è stato definito nel membro del sottosistema per la regione IMS , l'applicazione viene terminata in modo anomalo.

   z/OS Per ulteriori informazioni sull'utilizzo di queste macro, consultare Macro destinate all'utilizzo da parte del cliente.
8. Su IBM i, i programmi che terminano in modo anomalo non vengono automaticamente disconnessi dal gestore code. Scrivere le applicazioni per consentire la possibilità che la chiamata MQCONN o MQCONNX restituisca il codice di completamento MQCC_WARNING e il codice motivo MQRC_ALREADY_CONNECTED. Utilizzare l'handle di connessione restituito in questa situazione come normale.

Richiamo C

```
MQCONN (QMgrName, &Hconn, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQCHAR48  QMgrName;  /* Name of queue manager */
MQHCONN   Hconn;     /* Connection handle */
MQLONG    CompCode;  /* Completion code */
MQLONG    Reason;    /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQCONN' USING QMGRNAME, HCONN, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Name of queue manager
 01  QMGRNAME  PIC X(48).
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQCONN (QMgrName, Hconn, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl QMgrName  char(48);       /* Name of queue manager */
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQCONN,(QMGRNAME,HCONN,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
QMGRNAME  DS  CL48  Name of queue manager
HCONN     DS  F     Connection handle
COMPCODE  DS  F     Completion code
REASON    DS  F     Reason code qualifying COMPCODE
```

Richiamo Visual Basic

```
MQCONN QMgrName, Hconn, CompCode, Reason
```

Dichiarare i parametri come segue:

```
Dim QMgrName As String*48 'Name of queue manager'
```

```
Dim Hconn    As Long      'Connection handle'
Dim CompCode As Long      'Completion code'
Dim Reason   As Long      'Reason code qualifying CompCode'
```

MQCONNX - Gestore code di connessione (esteso)

La chiamata MQCONNX connette un programma applicativo a un gestore code. Fornisce un handle di connessione del gestore code, utilizzato dall'applicazione nelle successive chiamate IBM MQ .

La chiamata MQCONNX è simile alla chiamata MQCONN, ma MQCONNX consente di specificare opzioni per controllare il funzionamento della chiamata.

- Questa chiamata viene supportata su tutti i sistemi IBM MQ e i client IBM MQ connessi a questi sistemi.

Non è possibile stabilire una connessione client solo su un'installazione server e non è possibile stabilire una connessione locale solo su un'installazione client.

Sintassi

MQCONNX (*QMgrName*, *ConnectOpts*, *Hconn*, *CompCode*, *Motivo*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

Per i dettagli, consultare il parametro **QMgrName** descritto in “MQCONN - Connetti gestore code” a pagina 679 .

ConnectOpts
Tipo: MQCNO - input/output

Vedi “MQCNO - Opzioni di connessione” a pagina 321 per i dettagli.

Hconn
Tipo: MQHCONN - output

Questo handle rappresenta la connessione al gestore code. Specificarlo su tutte le chiamate di accodamento messaggi successive emesse dall'applicazione. Cessa di essere valida quando viene emessa la chiamata MQDISC o quando termina l'unità di elaborazione che definisce l'ambito dell'handle.

IBM MQ ora fornisce la libreria mqm con pacchetti client e pacchetti server. Ciò significa che quando viene effettuata una chiamata MQI che si trova nella libreria mqm, il tipo di connessione viene controllato per verificare se si tratta di una connessione client o server e quindi viene effettuata la chiamata sottostante corretta. Pertanto, un'uscita passata a *Hconn* può ora essere collegata alla libreria mqm, ma utilizzata su un'installazione client.

Gestisci ambito: L'ambito dell'handle restituito dipende dalla chiamata utilizzata per la connessione al gestore code (MQCONN o MQCONNX). Se la chiamata utilizzata è MQCONNX, l'ambito dell'handle dipende anche dall'opzione MQCNO_HANDLE_SHARE_ * specificata nel campo *Options* della struttura MQCNO.

- Se la chiamata è MQCONN o viene specificata l'opzione MQCNO_HANDLE_SHARE_NONE, l'handle restituito è un handle *non condiviso* .

  L'ambito di un handle non condiviso è l'unità di elaborazione parallela più piccola supportata dalla piattaforma su cui è in esecuzione l'applicazione (per i dettagli, consultare Tabella 547 a pagina 688); l'handle non è valido all'esterno dell'unità di elaborazione parallela da cui è stata emessa la chiamata.
- Se si specifica l'opzione MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK, l'handle restituito è un handle *condiviso* .

L'ambito di un handle condiviso è il processo che possiede il thread da cui è stata emessa la chiamata; l'handle può essere utilizzato da qualsiasi thread che appartiene a tale processo. Non tutte le piattaforme supportano i thread.

- Se la chiamata MQCONN o MQCONNX non riesce con codice di completamento uguale a MQCC_FAILED, il valore Hconn non è definito.

Tabella 547. Ambito degli handle non condivisi su varie piattaforme

Piattaforma	Ambito dell'handle non condiviso
z/OS	• CICS: l'attività CICS • IMS: l'attività, fino al punto di sincronizzazione successivo (escludendo le attività secondarie dell'attività) • z/OS batch e TSO: l'attività (escludendo le attività secondarie dell'attività)
IBM i	Lavoro
AIX and Linux	Sottoprocesso
Applicazioni Windows a 32 bit	Sottoprocesso
Applicazioni Windows a 64 bit	Sottoprocesso

Su applicazioni z/OS per CICS , il valore restituito è:

DEF_MQH_HCONN
Handle di connessione predefinito.

CompCode
Tipo: MQLONG - output

Per i dettagli, consultare il parametro **CompCode** descritto in “MQCONN - Connetti gestore code” a pagina 679 .

Motivo
Tipo: MQLONG - output

I seguenti codici possono essere restituiti dalle chiamate MQCONN e MQCONNX. Per un elenco di codici aggiuntivi che possono essere restituiti dalla chiamata MQCONNX, consultare i seguenti codici.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

MQRC_ALREADY_CONNECTED
(2002, X'7D2') Applicazione già connessa.

ERRORE DI CARICAMENTO MQRC_CLUSTER_EXIT_LOAD
(2267, X'8DB') Impossibile caricare l'uscita del carico di lavoro del cluster.

MQRC_SSL_ALREADY_INITIALIZED
(2391, X' 957 ') SSL già inizializzato.

Se *CompCode* è MQCC_FAILED:

ERRORE CARICAMENTO MQRC_ADAPTER_CONN_LOAD
(2129, X'851 ') Impossibile caricare il modulo di collegamento dell'adattatore.

ERRORE DEFS MQRC_ADAPTER_
(2131, X'853 ') Modulo definizione sottosistema adattatore non valido.

ERRORE CARICAMENTO MQRC_ADAPTER_DEFS_
(2132, X'854 ') Impossibile caricare il modulo di definizione del sottosistema dell'adattatore.

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'89C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

SCARSE_ARCHIVIAZIONE_ADATTATORE_MQRC_
(2127, X'84F') Memoria insufficiente per l'adattatore.

MQRC_ANOTHER_Q_MGR_CONNECTED
(2103, X'837 ') Un altro gestore code è già connesso.

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita.

MQRC_API_EXIT_INIT_ERROR
(2375, X' 947 ') Inizializzazione dell'uscita API non riuscita.

ERRORE USCITA MQRC_API
(2376, X' 948 ') Terminazione uscita API non riuscita.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

ERRORE MQRC_BUFFER_LENGTH
(2005, X'7D5') Parametro di lunghezza del buffer non valido.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

MQRC_CONN_ID_IN_USE
(2160, X'870 ') Identificativo di collegamento già in uso.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

ERRORE MQRC_CONNECTION_
(2273, X'8E1') Errore durante l'elaborazione della chiamata MQCONN.

MQRC_CONNECTION_NON_DISPONIBILE
(2568, X'A08') Si verifica su una chiamata MQCONN o MQCONNX quando il gestore code non è in grado di fornire una connessione del tipo di connessione richiesto sull'installazione corrente. Non è possibile effettuare una connessione client solo su un'installazione server. Non è possibile stabilire una connessione locale solo su un client.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Connessione in fase di sospensione.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Chiusura della connessione.

MQRC_CRYPTO_HARDWARE_ERROR
(2382, X'94E') Errore di configurazione hardware crittografico.

MQRC_DUPLICATE_RECOV_COORD
(2163, X'873 ') Il coordinatore di recupero esiste.

ERRORE MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Chiamata non valida nell'ambiente.

Inoltre, sulla chiamata MQCONNX, passando il blocco di controllo “MQCSP - Parametri di sicurezza” a pagina 340 da un'applicazione CICS o IMS .

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

MQRC_HOST_NON_DISPONIBILE
(2538, X'9EA') È stata emessa una chiamata MQCONN da un client per connettersi a un gestore code, ma il tentativo di assegnare una conversazione al sistema remoto non è riuscito.

MQRC_INSTALLATION_MISMATCH
(2583, X'A17') Mancata corrispondenza tra l'installazione del gestore code e la libreria selezionata.

MQRC_KEY_REPOSITORY_ERROR
(2381, X'94D') Repository chiavi non valido.

MQRC_MAX_CONNS_LIMIT_REACHED
(2025, X'7E9') Numero massimo di connessioni raggiunto.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

MQRC_OPEN_NON RIUSCITO
(2137, X'859 ') Oggetto non aperto correttamente.

ERRORE MQRC_Q_MGR_NAME_
(2058, X'80A') Nome gestore code non valido o sconosciuto.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Gestore code non disponibile per la connessione.

MQRC_Q_MGR_QUIESCING
(2161, X'871 ') Gestore code in fase di sospensione.

MQRC_Q_MGR_STOPPING
(2162, X'872 ') Chiusura del gestore code.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

ERRORE MQRC_SECURITY_ERROR
(2063, X'80F') Si è verificato un errore di sicurezza.

ERRORE MQRC_SSL_INITIALIZATION_ERROR
(2393, X' 959 ') Errore di inizializzazione SSL.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

I seguenti codici motivo aggiuntivi possono essere restituiti dalla chiamata MQCONNX:

Se *CompCode* è MQCC_FAILED:

ERRORE MQRC_AIR
(2385, X' 951 ') Record informazioni di autenticazione non valido.

ERRORE MQRC_AUTH_INFO_CONN_NAME_ERROR
(2387, X' 953 ') Nome connessione informazioni di autenticazione non valido.

ERRORE MQRC_AUTH_INFO_REC_COUNT_
(2383, X'94F') Conteggio record delle informazioni di autenticazione non valido.

ERRORE MQRC_AUTH_INFO_REC_
(2384, X' 950 ') Campi record informazioni di autenticazione non validi.

MQRC_AUTH_INFO_TYPE_ERRORE
(2386, X' 952 ') Tipo di informazioni di autenticazione non valido.

ERRORE MQRC_CD
(2277, X'8E5') Definizione di canale non valida.

ERRORE MQRC_CLIENT_CONN_
(2278, X'8E6') Campi di collegamento client non validi.

ERRORE_ERRORE_MQRC
(2139, X'85B') Struttura delle opzioni di connessione non valida.

MQRC_CONN_TAG_IN_USO
(2271, X'8DF') Tag di connessione in uso.

MQRC_CONN_TAG_NO_USABLE
(2350, X'92E') Tag di connessione non utilizzabile.

ERRORE MQRC_CSP
(2595, X'A23') La struttura MQCSP non è valida.

V 9.4.0 **MQRC_FUNZIONE_NON_SUPPORTATA**
(2298, X'8FA') La funzione richiesta non è disponibile nell'ambiente corrente.

ERRORE MQRC_LDAP_PASSWORD_ERRORE
(2390, X' 956 ') Password LDAP non valida.

ERRORE MQRC_LDAP_USER_NAME_ERROR
(2388, X' 954 ') I campi del nome utente LDAP non sono validi.

MQRC_LDAP_USER_NAME_LENGTH_ERR
(2389, X' 955 ') Lunghezza nome utente LDAP non valida.

ERRORE MQRC_OPTIONS_
(2046, X'7FE') Opzioni non valide o non congruenti.

ERRORE MQRC_SCO
(2380, X'94C') Struttura delle opzioni di configurazione SSL non valida.

ERRORE MQRC_SSL_CONFIG
(2392, X' 958 ') Errore di configurazione SSL.

MQRC_TOKEN_TIMESTAMP_NOT_VALID
(2064, X'810 ') Il token di autenticazione non è ancora valido o è scaduto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note d'utilizzo

Per il linguaggio di programmazione Visual Basic, si applica il seguente punto:

- Il parametro **ConnectOpts** è dichiarato di tipo MQCNO. Se l'applicazione è in esecuzione come IBM MQ MQI cliente si desidera specificare i parametri del canale di connessione client, dichiarare il parametro **ConnectOpts** come di tipo Any, in modo che l'applicazione possa specificare una struttura MQCNOCD sulla chiamata al posto di una struttura MQCNO. Tuttavia, ciò significa che non è possibile controllare il parametro **ConnectOpts** per assicurarsi che sia il tipo di dati corretto.

Richiamo C

```
MQCONNX (QMgrName, &ConnectOpts, &Hconn, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQCHAR48  QMgrName;     /* Name of queue manager */
MQCNO     ConnectOpts;  /* Options that control the action of MQCONNX */
MQHCONN   Hconn;        /* Connection handle */
MQLONG    CompCode;     /* Completion code */
MQLONG    Reason;       /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQCONNX' USING QMGRNAME, CONNECTOPTS, HCONN, COMPCODE,
REASON.
```

Dichiarare i parametri come segue:

```
**   Name of queue manager
 01  QMGRNAME     PIC X(48).
```

```
**   Options that control the action of MQCONNX
 01  CONNECTOPTS.
     COPY CMQCNOV.
**   Connection handle
 01  HCONN        PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE     PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON       PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQCONNX (QMgrName, ConnectOpts, Hconn, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl QMgrName     char(48);       /* Name of queue manager */
dcl ConnectOpts  like MQCNO;     /* Options that control the action of
                                    MQCONNX */
dcl Hconn        fixed bin(31);  /* Connection handle */
dcl CompCode     fixed bin(31);  /* Completion code */
dcl Reason       fixed bin(31);  /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQCONNX,(QMGRNAME,CONNECTOPTS,HCONN,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
QMGRNAME     DS       CL48  Name of queue manager
CONNECTOPTS  CMQCNOA  ,     Options that control the action of MQCONNX
HCONN        DS       F     Connection handle
COMPCODE     DS       F     Completion code
REASON       DS       F     Reason code qualifying COMPCODE
```

Richiamo Visual Basic

```
MQCONNX QMgrName, ConnectOpts, Hconn, CompCode, Reason
```

Dichiarare i parametri come segue:

```
Dim QMgrName    As String*48 'Name of queue manager'
Dim ConnectOpts As MQCNO     'Options that control the action of'
                             'MQCONNX'
Dim Hconn       As Long      'Connection handle'
Dim CompCode    As Long      'Completion code'
Dim Reason      As Long      'Reason code qualifying CompCode'
```

MQCRTMH - Creazione handle messaggio

La chiamata MQCRTMH restituisce un handle del messaggio.

Un'applicazione può utilizzare la chiamata MQCRTMH nelle chiamate di accodamento messaggi successive:

- Utilizzare la chiamata MQSETMP per impostare una proprietà dell'handle del messaggio.
- Utilizzare la chiamata MQINQMP per esaminare il valore di una proprietà dell'handle del messaggio.
- Utilizzare la chiamata MQDLTMP per eliminare una proprietà dell'handle del messaggio.

L'handle del messaggio può essere utilizzato nelle chiamate MQPUT e MQPUT1 per associare le proprietà dell'handle del messaggio con quelle del messaggio inserito. Allo stesso modo, specificando una gestione del messaggio nella chiamata MQGET, è possibile accedere alle proprietà del messaggio richiamato utilizzando la gestione del messaggio quando la chiamata MQGET viene completata.

Utilizzare MQDLTMH per eliminare l'handle del messaggio.

Sintassi

MQCRTMH (*Hconn*, *CrtMsgHOpts*, *Hmsg*, *CompCode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente. Se la connessione al gestore code cessa di essere valida e nessuna chiamata IBM MQ è in esecuzione sull'handle del messaggio, MQDLTMH viene richiamato implicitamente per eliminare il messaggio.

In alternativa, è possibile specificare il valore seguente:

MQHC_UNASSOCIAT_HCONN
L'handle di connessione non rappresenta una connessione ad alcun particolare gestore code.

Quando si utilizza questo valore, l'handle del messaggio deve essere eliminato con una chiamata esplicita a MQDLTMH per rilasciare qualsiasi memoria ad esso assegnata; IBM MQ non elimina mai implicitamente l'handle del messaggio.

È necessaria almeno una connessione valida a un gestore code stabilito sul thread che crea l'handle del messaggio, altrimenti la chiamata ha esito negativo con MQRC_HCONN_ERROR.

In un ambiente con più installazioni su un singolo sistema, il valore MQHC_UNASSOCIATED_HCONN è limitato all'utilizzo con la prima installazione caricata nel processo. Il codice motivo MQRC_HMSG_NOT_AVAILABLE viene restituito se l'handle del messaggio viene fornito a un'altra installazione.

In applicazioni z/OS per CICS la chiamata MQCONN può essere omessa ed è possibile specificare il valore seguente per *Hconn*:

CONN MQHC_DEF
Handle di connessione predefinito

CrtMsgHOpts
Tipo: MQCMHO - input

Le opzioni che controllano l'azione di MQCRTMH. Consultare MQCMHO per i dettagli.

Msg
Tipo: MQHMSG - output

Nell'output viene restituito un handle del messaggio che può essere utilizzato per impostare, analizzare ed eliminare le proprietà dell'handle del messaggio. Inizialmente l'handle del messaggio non contiene proprietà.

Un handle del messaggio ha anche un descrittore del messaggio associato. Inizialmente contiene i valori predefiniti. I valori dei campi descrittori dei messaggi associati possono essere impostati e interrogati utilizzando le chiamate MQSETMP e MQINQMP. La chiamata MQDLTMP reimposta un campo del descrittore del messaggio sul valore predefinito.

Se il parametro *Hconn* viene specificato come valore MQHC_UNASSOCIATED_HCONN, l'handle del messaggio restituito può essere utilizzato su chiamate MQGET, MQPUT o MQPUT1 con qualsiasi connessione all'interno dell'unità di elaborazione, ma può essere utilizzato solo da una chiamata IBM MQ alla volta. Se la gestione è in uso quando una seconda chiamata IBM MQ tenta di utilizzare la

stessa gestione dei messaggi, la seconda chiamata IBM MQ ha esito negativo con codice di errore MQRC_MSG_HANDLE_IN_USE.

Se il parametro *Hconn* non è MQHC_UNASSOCIATED_HCONN, l'handle del messaggio restituito può essere utilizzato solo sulla connessione specificata.

Lo stesso valore del parametro *Hconn* deve essere utilizzato nelle successive chiamate MQI in cui viene utilizzato questo handle del messaggio:

- MQDLTMH
- MQSETMP
- MQINQMP
- MQDLTMP
- MQMHBUF
- MQBUFMH

L'handle del messaggio restituito cessa di essere valido quando viene emessa la chiamata MQDLTMH per l'handle del messaggio o quando termina l'unità di elaborazione che definisce l'ambito dell'handle. MQDLTMH viene richiamato implicitamente se viene fornita una connessione specifica quando viene creato l'handle del messaggio e la connessione al gestore code cessa di essere valida, ad esempio, se viene richiamato MQDBC.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'089C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

MQRC_CALL_IN_PROVERDE
(2219, X'08AB') Chiamata MQI immessa prima del completamento della chiamata precedente.

ERRORE MQRC_CMHO_
(2461, X'099D') Struttura delle opzioni di gestione del messaggio non valida.

MQRC_CONNECTION_BROKEN
(2273, X'7D9') Connessione al gestore code persa.

MQRC_HANDLE_NON_DISPONIBILE
(2017, X'07E1') Nessun ulteriore handle disponibile.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

ERRORE MQRC_HMSG_
(2460, X'099C') Il puntatore della gestione messaggi non è valido.

ERRORE MQRC_OPTIONS_
(2046, X'07FE') Opzioni non valide o non congruenti.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

C

```
MQCRTMH (Hconn, &CrtMsgHOpts, &Hmsg, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;       /* Connection handle */
MQCMHO   CrtMsgHOpts; /* Options that control the action of MQCRTMH */
MQHMSG   Hmsg;        /* Message handle */
MQLONG   CompCode;    /* Completion code */
MQLONG   Reason;      /* Reason code qualifying CompCode */
```

COBOL

```
CALL 'MQCRTMH' USING HCONN, CRTMSGHOPTS, HMSG, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Options that control the action of MQCRTMH
 01  CRTMSGHOPTS.
     COPY CMQCMHOV.
**   Message handle
 01  HMSG    PIC S9(18) BINARY.
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

PL/I

```
call MQCRTMH (Hconn, CrtMsgHOpts, Hmsg, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn       fixed bin(31);  /* Connection handle */
dcl CrtMsgHOpts like MQCMHO;    /* Options that control the action of MQCRTMH */
dcl Hmsg        fixed bin(63);  /* Message handle */
dcl CompCode    fixed bin(31);  /* Completion code */
dcl Reason      fixed bin(31);  /* Reason code qualifying CompCode */
```

High Level Assembler

```
CALL MQCRTMH,(HCONN,CRTMSGHOPTS,HMSG,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN          DS        F  Connection handle
CRTMSGHOPTS    CMQCMHOA  ,  Options that control the action of MQCRTMH
HMSG           DS        D  Message handle
COMPCODE       DS        F  Completion code
REASON         DS        F  Reason code qualifying COMPCODE
```

MQCTL - Controllo callback

La chiamata MQCTL esegue azioni di controllo sui callback e gli handle di oggetto aperti per una connessione.

Sintassi

MQCTL (*Hconn*, *Operation*, *ControlOpts*, *CompCode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

Su applicazioni z/OS per CICS la chiamata MQCONN può essere omessa ed è possibile specificare il seguente valore speciale per *Hconn*:

DEF_MQH_HCONN
Handle di connessione predefinito.

Operazione
Tipo: MQLONG - input

L'operazione in fase di elaborazione sul callback definito per l'handle oggetto specificato. È necessario specificare solo una delle seguenti opzioni:

INIZIO_MQOP

Avviare l'utilizzo dei messaggi per tutte le funzioni del consumatore di messaggi definite per l'handle di connessione specificato.

I callback vengono eseguiti su un thread avviato dal sistema, che è diverso da qualsiasi thread dell'applicazione.

Questa operazione fornisce il controllo dell'handle di connessione fornito al sistema. Le uniche chiamate MQI che possono essere emesse da un thread diverso da quello consumer sono:

- MQCTL con operazione MQOP_STOP
- MQCTL con operazione MQOP_SUSPEND
- MQDISC - Esegue MQCTL con l'operazione MQOP_STOP prima di disconnettere HConn.

MQRC_HCONN_ASYNC_ACTIVE viene restituito se viene emessa una chiamata API IBM MQ mentre l'handle di connessione è avviato e la chiamata non ha origine da una funzione del consumer del messaggio.

Se un utente del messaggio arresta la connessione durante MQCBCT_START_CALL, la chiamata MQCTL restituisce un codice di errore MQRC_CONNECTION_STOPPED.

Questo può essere emesso in una funzione consumer. Per la stessa connessione della routine di callback, il solo scopo è annullare un'operazione MQOP_STOP precedentemente emessa.

Questa opzione non è supportata nei seguenti ambienti: CICS su z/OS o se l'applicazione è associata a una libreria IBM MQ senza thread.

MQOP_START_WAIT
Avviare l'utilizzo dei messaggi per tutte le funzioni del consumatore di messaggi definite per l'handle di connessione specificato.

I destinatari dei messaggi vengono eseguiti sullo stesso thread e il controllo non viene restituito al chiamante di MQCTL fino a quando:

- Rilasciato dall'utilizzo delle operazioni MQCTL MQOP_STOP o MQOP_SUSPEND oppure
- Tutte le routine consumer sono state annullate o sospese.

Se tutti i consumer vengono annullati o sospesi, viene emessa un'operazione MQOP_STOP implicita.

Questa opzione non può essere utilizzata dall'interno di una routine di callback, per l'handle di connessione corrente o per qualsiasi altro handle di connessione. Se la chiamata viene tentata, viene restituita con MQRC_ENVIRONMENT_ERROR.

Se, in qualsiasi momento durante un'operazione MQOP_START_WAIT, non vi sono consumer registrati e non sospesi, la chiamata ha esito negativo con codice motivo MQRC_NO_CALLBACKS_ACTIVE.

Se, durante un'operazione MQOP_START_WAIT, la connessione viene sospesa, la chiamata MQCTL restituisce un codice di errore di avvertenza MQRC_CONNECTION_SUSPENDED; la connessione rimane 'avviata'.

L'applicazione può scegliere di emettere MQOP_STOP o MQOP_RESUME. In questa istanza, i blocchi dell'operazione MQOP_RESUME.

Questa opzione non è supportata in un client a thread singolo.

MQOP_STOP
Arrestare l'utilizzo dei messaggi e attendere che tutti i consumer completino le operazioni prima del completamento di questa opzione. Questa operazione rilascia l'handle di connessione.

Se emessa dall'interno di una routine di callback, questa opzione non diventa effettiva fino a quando la routine non termina. Non vengono richiamate ulteriori routine del destinatario del messaggio dopo che sono state completate le routine del destinatario per i messaggi già letti e dopo che sono state effettuate le chiamate di arresto (se richieste) alle routine di richiamata.

Se emesso al di fuori di una routine di callback, il controllo non ritorna al chiamante fino a quando non sono state completate le routine consumer per i messaggi già letti e dopo che sono state effettuate le chiamate di arresto (se richieste) ai callback. I callback stessi, tuttavia, rimangono registrati.

Questa funzione non ha alcun effetto sui messaggi di lettura anticipata. È necessario assicurarsi che i consumatori eseguano MQCLOSE (MQCO_QUIESCE), dall'interno della funzione di callback, per determinare se sono disponibili ulteriori messaggi da consegnare.

MQOP_SOSPENSIONE
Sospendere l'utilizzo dei messaggi. Questa operazione rilascia l'handle di connessione.

Ciò non ha alcun effetto sulla lettura prima dei messaggi per l'applicazione. Se si intende interrompere il consumo dei messaggi per un lungo periodo di tempo, considerare la chiusura della coda e riaprirla quando il consumo continua.

Se emesso dall'interno di una routine di callback, non diventa effettivo fino a quando la routine non esce. Non verranno richiamate ulteriori routine del destinatario del messaggio dopo l'uscita della routine corrente.

Se emesso all'esterno di un callback, il controllo non ritorna al chiamante fino a quando non viene completata la routine consumer corrente e non vengono richiamate altre.

MQOP_RESUME
Riprendere l'utilizzo dei messaggi.

Questa opzione viene normalmente emessa dal thread dell'applicazione principale, ma può essere utilizzata anche dall'interno di una routine di callback per annullare una precedente richiesta di sospensione emessa nella stessa routine.

Se MQOP_RESUME viene utilizzato per riprendere un MQOP_START_WAIT, l'operazione si blocca.

ControlOpts
Tipo: MQCTLO - input

Opzioni che controllano l'azione di MQCTL

Consultare MQCTLO per i dettagli della struttura.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE CARICAMENTO MQRC_ADAPTER_CONV
(2133, X'855 ') Impossibile caricare i moduli dei servizi di conversione dati.

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'89C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita.

ERRORE USCITA MQRC_API
(2183, X'887 ') Impossibile caricare l'uscita API.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

ERRORE MQRC_BUFFER_LENGTH
(2005, X'7D5') Parametro di lunghezza del buffer non valido.

ERRORE MQRC_CALLBACK_LINK
(2487, X'9B7') Impossibile richiamare la routine di callback

MQRC_CALLBACK_NOT_ REGISTRATO
(2448, X' 990 ') Impossibile annullare la registrazione, sospendere o riprendere perché non è presente alcun callback registrato

ERRORE DI MQRC_CALLBACK_ROUTINE_ERROR
(2486, X'9B6') Sia CallbackFunction che CallbackName sono stati specificati in una chiamata MQOP_REGISTER.

Oppure è stato specificato CallbackFunction o CallbackName ma non corrisponde alla funzione di callback attualmente registrata.

ERRORE MQRC_CALLBACK_TIPO
(2483, X'9B3') Campo tipo CallBacknon corretto.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

ERRORE MQRC_CBD
(2444, X'98C') Il blocco di opzione è errato.

ERRORE MQRC_CBD_OPTIONS_
(2484, X'9B4') Campo di opzioni MQCBD non corretto.

MQRC_CICS_WAIT_NON RIUSCITO
(2140, X'85C') Richiesta di attesa rifiutata da CICS.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Non autorizzato per la connessione.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Connessione in fase di sospensione.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Chiusura della connessione.

ERRORE ID CORREL_MQR_
(2207, X'89F') Errore identificativo di correlazione.

ERRORE MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Chiamata non valida nell'ambiente.

MQRC_FUNZIONE_NON_SUPPORTATA
(2298, X'8FA') La funzione richiesta non è disponibile nell'ambiente corrente.

MQRC_GET_INHIBITED
(2016, X'7E0') Ottiene inibiti per la coda.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Unità globali di conflitto di lavoro.

ERRORE MQRC_GMO
(2186, X'88A') Struttura delle opzioni Get - message non valida.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X' 931 ') Handle in uso per l'unità di lavoro globale.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

ERRORE MQRC_HOBJ_R
(2019, X'7E3') Handle oggetto non valido.

BROWSE INCONSIST_MQRC
(2259, X'8D3') Specifica di ricerca incongruente.

UOW MQRC_INCONSISTENT_
(2245, X'8C5') Specifica dell'unità di lavoro non congruente.

MQRC_INVALID_MSG_UNDER_CURSOR
(2246, X'8C6') Messaggio sotto il cursore non valido per il recupero.

MQRC_LOCAL_UOW_CONFLICT
(2352, X' 930 ') L'unità di lavoro globale è in conflitto con l'unità di lavoro locale.

ERRORE MQRC_MATCH_OPTIONS_
(2247, X'8C7') Opzioni di corrispondenza non valide.

ERRORE MQRC_MAX_MSG_LENGTH
(2485, X'9B5') Campo di lunghezza MaxMsgnon corretto

ERRORE MQRC_MD
(2026, X'7EA') Descrittore messaggio non valido.

MQRC_MODULE_ENTRY_NOT_FOUND
(2497, X'9C1') Il punto di ingresso della funzione specificato non è stato trovato nel modulo.

ID_NON_VALIDO MQRC_MODULE
(2496, X'9C0') Il modulo è stato trovato ma è di tipo errato (32 bit/64 bit) o non è una dll valida.

MQRC_MODULE_NOT_FOUND
(2495, X'9BF') Modulo non trovato nel percorso di ricerca o non autorizzato al caricamento.

ERRORE ID MQRC_MSG_
(2206, X'89E') Errore identificativo messaggio.

ERRORE MQRC_MSG_SEQ_NUMBER_
(2250, X'8CA') Numero di sequenza messaggio non valido.

ERRORE MQRC_MSG_TOKEN_
(2331, X'91B') L'utilizzo del token del messaggio non è valido.

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') Coda non aperta per la ricerca.

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') Coda non aperta per l'input.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Definizione oggetto modificata dall'apertura.

MQRC_OBJECT_DAMAGED
(2101, X'835 ') Oggetto danneggiato.

ERRORE OPERAZIONE MQRC
(2488, X'9B8') Codice operazione non corretto nella chiamata API

ERRORE MQRC_OPTIONS_
(2046, X'7FE') Opzioni non valide o non congruenti.

ERRORE MQRC_PAGESET_
(2193, X'891 ') Errore durante l'accesso al dataset della serie di pagine.

MQRC_Q_XX_ENCODE_CASE_ONE eliminato
(2052, X'804 ') La coda è stata eliminata.

MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') La coda ha un tipo di indice errato.

ERRORE MQRC_Q_MGR_NAME_
(2058, X'80A') Nome gestore code non valido o sconosciuto.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Gestore code non disponibile per la connessione.

MQRC_Q_MGR_QUIESCING
(2161, X'871 ') Gestore code in fase di sospensione.

MQRC_Q_MGR_STOPPING
(2162, X'872 ') Chiusura del gestore code.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

MQRC_SIGNAL_OUTSTANDING
(2069, X'815 ') Segnale eccezionale per questa maniglia.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Chiamata eliminata dal programma di uscita.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818 ') Supporto punto di sincronizzazione non disponibile.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

ERRORE MQRC_UOW_ENLISTMENT_ERROR
(2354, X' 932 ') L'inserimento nell'unità di lavoro globale non è riuscito.

MQRC_UOW_MIX_NON_SUPPORTATO
(2355, X' 933 ') La miscelazione delle chiamate UOW non è supportata.

MQRC_UOW_NON_DISPONIBILE
(2255, X'8CF') Unità di lavoro non disponibile per il gestore code.

ERRORE INTERVAL_WAIT_MQRC
(2090, X'82A') Intervallo di attesa in MQGMO non valido.

MQRC_WRONG_GMO_VERSIONE
(2256, X'8D0') Versione errata di MQGMO fornita.

MQRC_WRONG_MD_VERSIONE
(2257, X'8D1') Versione non corretta di MQMD fornita.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note d'utilizzo

1. Le routine di callback devono controllare le risposte da tutti i servizi richiamati e, se la routine rileva una condizione che non può essere risolta, deve emettere un comando MQCB MQOP_DEREGISTER per impedire chiamate ripetute alla routine di callback.
2. Se si utilizza un utilizzo asincrono in un'applicazione in cui un gestore transazioni XA gestisce le transazioni globali, inclusi gli aggiornamenti a IBM MQ, è necessario considerare i seguenti punti aggiuntivi:
   a. Non è valido richiamare MQCTL (MQOP_START) per un **HConn**, dopo che è stato creato, dopo aver richiamato **xa_open**.

      Il motivo è che **HConn** è stato collegato ad un contesto XA e quindi non è possibile accedervi sul thread separato, o sui thread, in uso dal meccanismo di utilizzo asincrono.

   b. Se si richiama MQCTL (MQOP_START) in tale scenario, la chiamata ha esito negativo con il codice motivo MQRC_ASYNC_XA_CONFLICT (2350).

   c. È valido per richiamare MQCTL (MQOP_START_WAIT) per un **HConn**, dopo che è stato creato, dopo aver richiamato **xa_open**.

      Il motivo è che questo metodo di avvio del meccanismo di utilizzo asincrono causa ulteriori callback per **HConn** da eseguire sul thread in cui viene eseguita la chiamata MQCTL. Pertanto, il link tra **HConn** e il thread non viene perso.
3. z/OS Su z/OS, quando l'operazione è MQOP_START:
   - I programmi che utilizzano routine di callback asincrona devono essere autorizzati a utilizzare z/OS UNIX System Services (z/OS UNIX).
   - I programmi LE (Language Environment) che utilizzano routine di callback asincrona devono utilizzare l'opzione di runtime LE POSIX(ON).
   - I programmi non LE che utilizzano routine di callback asincrone non devono utilizzare l'interfaccia z/OS UNIX pthread_create (servizio richiamabile BPX1PTC).
4. z/OS MQCTL non è supportato all'interno dell'adattatore IMS .

Nota: In CICS, MQOP_START non è supportata. Utilizzare invece la chiamata di funzione MQOP_START_WAIT.

Richiamo C

```
MQCTL (Hconn, Operation, &ControlOpts, &CompCode, &Reason)
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;         /* Connection handle */
MQLONG   Operation;     /* Operation being processed */
MQCTLO   ControlOpts    /* Options that control the action of MQCTL */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQCTL' USING HCONN, OPERATION, CTLOPTS, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN    PIC S9(9) BINARY.
**   Operation
 01  OPERATION PIC S9(9) BINARY.
**   Control Options
 01  CTLOPTS.
     COPY CMQCTLOV.
**   Completion code
 01  COMPCODE PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON   PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQCTL(Hconn, Operation, CtlOpts, CompCode, Reason)
```

Dichiarare i parametri come segue:

```
dcl Hconn        fixed bin(31); /* Connection handle */
dcl Operation    fixed bin(31); /* Operation */
dcl CtlOpts like MQCTLO;        /* Options that control the action of MQCTL */
dcl CompCode     fixed bin(31); /* Completion code */
dcl Reason       fixed bin(31); /* Reason code qualifying CompCode */
```

MQDISC - Disconnetti gestore code

La chiamata MQDISC interrompe la connessione tra il gestore code e il programma applicativo ed è l'inverso della chiamata MQCONN o MQCONNX.

- Su z/OS, tutte le applicazioni che utilizzano il consumo asincrono dei messaggi, la gestione degli eventi o il callback, il thread di controllo principale deve emettere una chiamata MQDISC prima di terminare. Per ulteriori dettagli, consultare Utilizzo asincrono dei messaggi IBM MQ .
- Su z/OS, le applicazioni CICS non devono emettere questa chiamata per disconnettersi dal gestore code.

  Se un'applicazione CICS effettua questa chiamata, non ha alcun effetto a meno che non sia stata effettuata una chiamata MQCONNX precedente, specificando una delle seguenti opzioni:

  MQCNO_SERIALIZE_CONN_TAG_Q_MGR
  MQCNO_SERIALIZE_CONN_TAG_QSG
  MQCNO_RESTRICT_CONN_TAG_Q_MGR o
  MQCNO_RESTRICT_CONN_TAG_QSG

  , nel qual caso tutti gli handle di oggetti attualmente aperti vengono chiusi.

Sintassi

MQDISC (*Hconn*, *CompCode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input/output

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

Su z/OS per applicazioni CICS è possibile omettere la chiamata MQCONN e specificare il valore seguente per *Hconn*:

DEF_MQH_HCONN
Handle di connessione predefinito.

Una volta completata correttamente la chiamata, il gestore code imposta *Hconn* su un valore che non è un handle valido per l'ambiente. Questo valore è:

MQH_UNUSABLE_HCONN
Handle di connessione inutilizzabile.

Su z/OS, *Hconn* è impostato su un valore non definito.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti codici:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

MQRC_BACK_OUT
(2003, X'7D3') Unità di lavoro ripristinata.

MQRC_CONN_TAG_NOT_RELEASED
(2344, X' 928 ') Tag di connessione non rilasciata.

MQRC_OUTCOME_PENDING
(2124, X'84C') Il risultato dell'operazione di commit è in sospeso.

Se *CompCode* è MQCC_FAILED:

ERRORE CARICAMENTO MQRC_ADAPTER_DISC
(2138, X'85A') Impossibile caricare il modulo di disconnessione dell'adattatore.

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'89C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita.

MQRC_API_EXIT_INIT_ERROR
(2375, X' 947 ') Inizializzazione dell'uscita API non riuscita.

ERRORE USCITA MQRC_API
(2376, X' 948 ') Terminazione uscita API non riuscita.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Chiusura della connessione.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

MQRC_OUTCOME_MIXED
(2123, X'84B') Il risultato dell'operazione di commit o di backout è misto.

ERRORE MQRC_PAGESET_
(2193, X'891 ') Errore durante l'accesso al dataset della serie di pagine.

ERRORE MQRC_Q_MGR_NAME_
(2058, X'80A') Nome gestore code non valido o sconosciuto.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Gestore code non disponibile per la connessione.

MQRC_Q_MGR_STOPPING
(2162, X'872 ') Chiusura del gestore code.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note d'utilizzo

1. Se viene emessa una chiamata MQDISC quando la connessione ha ancora oggetti aperti in quella connessione, il gestore code chiude tali oggetti, con le opzioni di chiusura impostate su MQCO_NONE.
2. Se l'applicazione termina con modifiche di cui non è stato eseguito il commit in un'unità di lavoro, la disposizione di tali modifiche dipende dal modo in cui termina l'applicazione:

   a. Se l'applicazione emette la chiamata MQDISC prima di terminare:

   - Per un'unità di lavoro coordinata dal gestore code, il gestore code emette la chiamata MQCMIT per conto dell'applicazione. Se possibile, viene eseguito il commit dell'unità di lavoro e, in caso contrario, viene eseguito il backout.
   - Per un'unità di lavoro coordinata esternamente, non vi è alcuna variazione nello stato dell'unità di lavoro; tuttavia, il gestore code generalmente indica che l'unità di lavoro deve essere sottoposta a commit quando richiesto dal coordinatore dell'unità di lavoro.

     Su z/OS, CICS, IMS (diversi dai programmi DL/1 batch) e le applicazioni RRS sono simili.

   b. Se l'applicazione termina normalmente ma senza emettere la chiamata MQDISC, l'azione intrapresa dipende dall'ambiente:

   - In z/OS, ad eccezione delle applicazioni MQ Java o MQ JMS , si verificano le azioni descritte nella nota 2a .
   - In tutti gli altri casi, si verificano le azioni descritte nella nota 2c .

A causa delle differenze tra gli ambienti, assicurarsi che le applicazioni che si desidera trasferire eseguano il commit o il backout dell'unità di lavoro prima che terminino.

c. Se l'applicazione termina *in modo anomalo* senza emettere la chiamata MQDISC, viene eseguito il backout dell'unità di lavoro.

3. In z/OS, si applicano i punti seguenti:

- Le applicazioni CICS non devono emettere la chiamata MQDISC per disconnettersi dal gestore code, poiché il sistema CICS stesso si connette al gestore code e la chiamata MQDISC non ha alcun effetto su questa connessione.
- Le applicazioni CICS, IMS (diverse dai programmi batch DL/1) e RRS utilizzano unità di lavoro coordinate da un coordinatore dell'unità di lavoro esterno. Di conseguenza, la chiamata MQDISC non influisce sullo stato dell'unità di lavoro (se presente) che esiste quando viene emessa la chiamata.

  Tuttavia, la chiamata MQDISC indica la fine dell'utilizzo della tag di connessione *ConnTag* associata alla connessione da una precedente chiamata MQCONNX emessa dall'applicazione. Se è presente un'unità di lavoro attiva che fa riferimento alla tag di connessione quando viene emessa la chiamata MQDISC, la chiamata viene completata con il codice di completamento MQCC_WARNING e il codice motivo MQRC_CONN_TAG_NOT_RELEASED. La tag di connessione non diventa disponibile per il riutilizzo fino a quando il coordinatore dell'unità di lavoro esterna non ha risolto l'unità di lavoro.

Nota: In CICS, MQOP_START non è supportata. Utilizzare invece la chiamata di funzione MQOP_START_WAIT.

Richiamo C

```
MQDISC (&Hconn, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQDISC' USING HCONN, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQDISC (Hconn, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Richiamo assembler System/390

```
CALL MQDISC,(HCONN,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Richiamo Visual Basic

```
MQDISC Hconn, CompCode, Reason
```

Dichiarare i parametri come segue:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQDLTMH - Elimina gestione messaggi

La chiamata MQDLTMH elimina un handle del messaggio ed è l'inverso della chiamata MQCRTMH.

Sintassi

MQDLTMH (*Hconn*, *Hmsg*, *DltMsgHOpts*, *CompCode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code.

Il valore deve corrispondere all'handle di connessione utilizzato per creare l'handle del messaggio specificato nel parametro **Hmsg** .

Se l'handle del messaggio è stato creato utilizzando MQHC_UNASSOCIATED_HCONN, è necessario stabilire una connessione valida sul thread che elimina l'handle del messaggio, altrimenti la chiamata non riesce con MQRC_CONNECTION_BROKEN.

Msg
Tipo: MQHMSG - input/output

Questo è l'handle del messaggio da eliminare. Il valore è stato restituito da una precedente chiamata MQCRTMH.

Una volta completata correttamente la chiamata, l'handle viene impostato su un valore non valido per l'ambiente. Questo valore è:

MQHM_UNUSABLE_HMSG
Gestore messaggi inutilizzabile.

L'handle del messaggio non può essere eliminato se è in corso un'altra chiamata IBM MQ che ha passato lo stesso handle del messaggio.

DltMsgHOpts
Tipo: MQDMHO - input

Consultare MQDMHO per i dettagli.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'089C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

MQRC_CALL_IN_PROVERDE
(2219, X'08AB') Chiamata MQI immessa prima del completamento della chiamata precedente.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Connessione al gestore code persa.

ERRORE DI MQRC_DMHO_
(2462, X'099E') Struttura delle opzioni di eliminazione dell'handle del messaggio non valida.

ERRORE MQRC_HMSG_
(2460, X'099C') Il puntatore della gestione messaggi non è valido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') handle del messaggio già in uso.

ERRORE MQRC_OPTIONS_
(2046, X'07FE') Opzioni non valide o non congruenti.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Richiamo C

```
MQDLTMH (Hconn, &Hmsg, &DltMsgHOpts, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;       /* Connection handle */
MQHMSG   Hmsg;        /* Message handle */
MQDMHO   DltMsgHOpts; /* Options that control the action of MQDLTMH */
MQLONG   CompCode;    /* Completion code */
MQLONG   Reason;      /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQDLTMH' USING HCONN, HMSG, DLTMSGHOPTS, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**  Connection handle
01   HCONN   PIC S9(9) BINARY.

**  Options that control the action of MQDLTMH
01   DLTMSGHOPTS.
COPY CMQDMHOL.

**  Completion code
01  COMPCODE   PIC S9(9) BINARY.

** Reason code qualifying COMPCODE
01   REASON      PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQDLTMH (Hconn, Hmsg, DltMsgHOpts, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn          /* Connection handle */
dcl Hmsg           /* Message handle */
dcl DltMsgHOpts like MQDMHO;    /* Options that control the action of MQDLTMH */
dcl CompCode       /* Completion code */
dcl Reason         /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQDLTMH,(HCONN,HMSG,DLTMSGHOPTS,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN          DS        F  Connection handle
HMSG           DS        D  Message handle
DLTMSGHOPTS    CMQDMHOA  ,  Options that control the action of MQDLTMH
COMPCODE       DS        F  Completion code
REASON         DS        F  Reason code qualifying COMPCODE
```

MQDLTMP - Proprietà Elimina messaggio

La chiamata MQDLTMP elimina una proprietà da un handle del messaggio ed è l'inverso della chiamata MQSETMP.

Sintassi

MQDLTMP (*Hconn*, *Hmsg*, *DltPropOpts*, *Name*, *CompCode*, *Reason*)

Parametri

Hconn

Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore deve corrispondere all'handle di connessione utilizzato per creare l'handle del messaggio specificato nel parametro **Hmsg** .

Se l'handle del messaggio è stato creato utilizzando MQHC_UNASSOCIATED_HCONN, è necessario stabilire una connessione valida sul thread eliminando l'handle del messaggio, altrimenti la chiamata non riesce con MQRC_CONNECTION_BROKEN.

Msg
Tipo: MQHMSG - input

Questo è l'handle del messaggio contenente la proprietà da eliminare. Il valore è stato restituito da una precedente chiamata MQCRTMH.

Opzioni DltProp
Tipo: MQDMPO - input

Consultare il tipo di dati MQDMPO per i dettagli.

Nome
Tipo: MQCHARV - input

Il nome della proprietà da eliminare. Consultare Nomi proprietà per ulteriori informazioni sui nomi proprietà.

I caratteri jolly non sono consentiti nel nome proprietà.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

PROPRIETÀ_MQRC_NON_DISPONIBILE
(2471, X'09A7') Proprietà non disponibile.

ERRORE MQRC_RFH_FORMATO
(2421, X'0975 ') Impossibile analizzare una cartella MQRFH2 contenente le proprietà.

Se *CompCode* è MQCC_FAILED:

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'089C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'0852 ') Impossibile caricare il modulo di servizio adattatore.

MQRC_ASID_MISMATCH
(2157, X'086D') Gli ASID principale e principale differiscono.

MQRC_CALL_IN_PROVERDE
(2219, X'08AB') Chiamata MQI immessa prima del completamento della chiamata precedente.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Connessione al gestore code persa.

ERRORE DMPO_MQRC_
(2481, X'09B1') Struttura delle opzioni di eliminazione della proprietà del messaggio non valida.

ERRORE MQRC_HMSG_
(2460, X'099C') Gestione messaggio non valida.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') handle del messaggio già in uso.

ERRORE MQRC_OPTIONS_
(2046, X'07FE') Opzioni non valide o non congruenti.

ERRORE MQRC_PROPERTY_NAME_
(2442, X'098A') Nome proprietà non valido.

ERRORE CCSID DI MQRC_SOURCE_
(2111, X'083F') Identificativo serie di caratteri codificato del nome proprietà non valido.

ERRORE MQRC_UNEXPECTED_
(2195, X'0893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Richiamo C

```
MQDLTMP (Hconn, Hmsg, &DltPropOpts, &Name, &CompCode, &Reason)
```

Dichiarare i parametri come segue:

```
MQHCONN Hconn;       /* Connection handle */
MQHMSG  Hmsg;        /* Message handle */
MQDMPO  DltPropOpts; /* Options that control the action of MQDLTMP */
MQCHARV Name;        /* Property name */
MQLONG  CompCode;    /* Completion code */
MQLONG  Reason;      /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQDLTMP' USING HCONN, HMSG, DLTPROPOPTS, NAME, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN    PIC S9(9) BINARY.
**   Message handle
 01  HMSG         PIC S9(18) BINARY.
**   Options that control the action of MQDLTMP
 01  DLTPROPOPTS.
     COPY CMQDMPOV.
**   Property name
 01  NAME.
     COPY CMQCHRVV.
**   Completion code
 01  COMPCODE PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON   PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQDLTMP (Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn       fixed bin(31); /* Connection handle */
dcl Hmsg        fixed bin(63); /* Message handle */
dcl DltPropOpts like MQDMPO;   /* Options that control the action of MQDLTMP */
```

```
dcl Name        like MQCHARV;  /* Property name */
dcl CompCode    fixed bin(31); /* Completion code */
dcl Reason      fixed bin(31); /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQDLTMP,(HCONN,HMSG,DLTPROPOPTS,NAME,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN        DS         F     Connection handle
HMSG         DS         D     Message handle
DLTPROPOPTS  CMQDMPOA   ,     Options that control the action of MQDLTMP
NAME         CMQCHRVA   ,     Property name
COMPCODE     DS         F     Completion code
REASON       DS         F     Reason code qualifying COMPCODE
```

MQGET - Richiama messaggio

La chiamata MQGET richiama un messaggio da una coda locale che è stata aperta utilizzando la chiamata MQOPEN.

Sintassi

MQGET (*Hconn*, *Hobj*, *MsgDesc*, *GetMsgOpts*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

In applicazioni z/OS per CICS , la chiamata MQCONN può essere omessa e il seguente valore specificato per *Hconn*:

DEF_MQH_HCONN
Handle di connessione predefinito.

HOBJ
Tipo: MQHOBJ - input

Questo handle rappresenta la coda da cui deve essere richiamato un messaggio. Il valore di *Hobj* è stato restituito da una chiamata MQOPEN precedente. La coda deve essere stata aperta con una o più delle seguenti opzioni (consultare “MQOPEN - Apri oggetto” a pagina 751 per i dettagli):

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_SFOGLIA

MsgDesc
Tipo: MQMD - input/output

Questa struttura descrive gli attributi del messaggio richiesto e gli attributi del messaggio richiamato. Vedi “MQMD - Descrittore messaggi” a pagina 431 per i dettagli.

Se *BufferLength* è inferiore alla lunghezza del messaggio, *MsgDesc* viene riempito dal gestore code, se MQGMO_ACCEPT_TRUNCATED_MSG è specificato nel parametro **GetMsgOpts** (consultare MQGMO - campo Opzioni).

Se l'applicazione fornisce un MQMD version-1 , il messaggio restituito ha un prefisso MQMDE ai dati del messaggio dell'applicazione, ma solo se uno o più campi in MQMDE hanno un valore non predefinito. Se tutti i campi in MQMDE hanno valori predefiniti, MQMDE viene omesso. Un nome formato MQFMT_MD_EXTENSION nel campo *Formato* in MQMD indica che è presente un MQMDE.

L'applicazione non deve fornire una struttura MQMD se viene fornito un handle del messaggio valido nel campo *MsgHandle* . Se non viene fornito nulla in questo campo, il descrittore del messaggio viene preso dal descrittore associato agli handle del messaggio.

Se l'applicazione fornisce un handle del messaggio invece di una struttura MQMD e specifica MQGMO_PROPERTIES_FORCE_MQRFH2, la chiamata ha esito negativo con codice motivo MQRC_MD_ERROR. La chiamata, inoltre, ha esito negativo, con codice motivo MQRC_MD_ERROR, se l'applicazione non fornisce una struttura MQMD e specifica MQGMO_PROPERTIES_AS_Q_DEF e l'attributo della coda **PropertyControl** è MQPROP_FORCE_MQRFH2.

Se vengono specificate le opzioni di corrispondenza e viene utilizzato il descrittore del messaggio associato all'handle del messaggio, i campi di immissione utilizzati per la corrispondenza provengono dall'handle del messaggio.

Opzioni GetMsg
Tipo: MQGMO - input/output

Vedi "MQGMO - Opzioni Get - message" a pagina 376 per i dettagli.

BufferLength
Tipo: MQLONG - input

Si tratta della lunghezza in byte dell'area *Buffer* . Specificare zero per i messaggi che non hanno dati o se il messaggio deve essere rimosso dalla coda e i dati eliminati (in questo caso, è necessario specificare MQGMO_ACCEPT_TRUNCATED_MSG).

Nota: La lunghezza del messaggio più lungo che è possibile leggere dalla coda viene fornita dall'attributo coda **MaxMsgLength** ; consultare "Attributi per le code" a pagina 858.

Memorizza nel buffer
Tipo: MQBYTExBufferLength - output

Questa è l'area che contiene i dati del messaggio. Allineare il buffer su un limite appropriato alla natura dei dati nel messaggio. L'allineamento a 4 byte è adatto per la maggior parte dei messaggi (inclusi i messaggi contenenti strutture di intestazione IBM MQ), ma alcuni messaggi potrebbero richiedere un allineamento più rigoroso. Ad esempio, un messaggio contenente un numero intero binario a 64 bit potrebbe richiedere un allineamento a 8 byte.

Se *BufferLength* è inferiore alla lunghezza del messaggio, la maggior parte del messaggio viene spostata in **Buffer**. Ciò si verifica se MQGMO_ACCEPT_TRUNCATED_MSG viene specificato nel parametro **GetMsgOpts** (per ulteriori informazioni, consultare MQGMO - campo Opzioni).

La serie di caratteri e la codifica dei dati in **Buffer** vengono forniti dai campi *CodedCharSetId* e *Encoding* restituiti nel parametro **MsgDesc** . Se questi valori sono diversi da quelli richiesti dal destinatario, il destinatario deve convertire i dati del messaggio dell'applicazione nella serie di caratteri e nella codifica richiesti. L'opzione MQGMO_CONVERT può essere utilizzata (con un'uscita scritta dall'utente, se necessario) per convertire i dati del messaggio; consultare "MQGMO - Opzioni Get - message" a pagina 376 per i dettagli di questa opzione.

Nota: Tutti gli altri parametri della chiamata MQGET si trovano nella serie di caratteri e nella codifica del gestore code locale (forniti dall'attributo del gestore code **CodedCharSetId** e MQENC_NATIVE).

Se la chiamata ha esito negativo, è possibile che il contenuto del buffer sia stato ancora modificato.

Nel linguaggio di programmazione C, il parametro viene dichiarato come un puntatore a void: l'indirizzo di qualsiasi tipo di dati può essere specificato come parametro.

Se il parametro **BufferLength** è zero, *Buffer* non viene indicato; in questo caso, l'indirizzo del parametro passato dai programmi scritti nell'assembler C o System/390 può essere null.

DataLength
Tipo: MQLONG - output

Questa è la lunghezza in byte dei dati dell'applicazione *nel messaggio*. Se questo valore è maggiore di *BufferLength*, nel parametro **Buffer** vengono restituiti solo *BufferLength* byte (ossia, il messaggio viene troncato). Se il valore è zero, il messaggio non contiene dati dell'applicazione.

Se *BufferLength* è inferiore alla lunghezza del messaggio, *DataLength* viene ancora completato dal gestore code, se MQGMO_ACCEPT_TRUNCATED_MSG è specificato nel parametro **GetMsgOpts** (per ulteriori informazioni, consultare MQGMO - campo Opzioni). Ciò consente all'applicazione di determinare la dimensione del buffer richiesto per contenere i dati del messaggio e quindi emettere nuovamente la chiamata con un buffer della dimensione appropriata.

Tuttavia, se l'opzione MQGMO_CONVERT è specificata e i dati del messaggio convertito sono troppo lunghi per rientrare in *Buffer*, il valore restituito per *DataLength* è:

- La lunghezza dei dati *non convertiti* , per formati definiti dal gestore code.

  In questo caso, se la natura dei dati causa l'espansione durante la conversione, l'applicazione deve assegnare un buffer più grande del valore restituito dal gestore code per *DataLength*.
- Il valore restituito dall'uscita di conversione dati, per formati definiti dall'applicazione.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

I codici di errore elencati sono quelli che il gestore code può restituire per il parametro **Reason** . Se l'applicazione specifica l'opzione MQGMO_CONVERT e viene richiamata un'uscita scritta dall'utente per convertire alcuni o tutti i dati del messaggio, l'uscita decide quale valore viene restituito per il parametro **Reason** . Di conseguenza, sono possibili valori diversi da quelli documentati.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848 ') Dati convertiti troppo grandi per il buffer.

MQRC_CONVERTED_STRING_TOO_BIG
(2190, X'88E') Stringa convertita troppo grande per il campo.

ERRORE MQRC_DBCS
(2150, X'866 ') Stringa DBCS non valida.

ERRORE MQRC_FORMAZIONE
(2110, X'83E') Formato del messaggio non valido.

GRUPPO_INCOMPLE_MQRC
(2241, X'8C1') Gruppo di messaggi non completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Messaggio logico non completo.

CCSIDS INCONSIST_MQRC
(2243, X'8C3') I segmenti di messaggi hanno CCSID differenti.

MQRC_INCONSISTENT_ENCODINGS
(2244, X'8C4') I segmenti dei messaggi hanno codifiche differenti.

UOW MQRC_INCONSISTENT_
(2245, X'8C5') Specifica dell'unità di lavoro non congruente.

ERRORE MQRC_MSG_TOKEN_
(2331, X'91B') Utilizzo non valido del token del messaggio.

MQRC_NO_MSG_LOCKED
(2209, X'8A1') Nessun messaggio bloccato.

MQRC_NOT_CONVERTED
(2119, X'847 ') Dati del messaggio non convertiti.

MQRC_OPTIONS_CHANGED
(nnnn, X'xxx ') Le opzioni che dovevano essere congruenti sono state modificate.

MQRC_PARTIALLY_CONVERTED
(2272, X'8E0') Dati del messaggio parzialmente convertiti.

MQRC_SIGNAL_REQUEST_ACCEPTED
(2070, X'816 ') Nessun messaggio restituito (ma richiesta di segnale accettata).

ERRORE_ORIGINE_RISORSE MQRC
(2145, X'861 ') Parametro del buffer di origine non valido.

ERRORE CCSID DI MQRC_SOURCE_
(2111, X'83F') Identificativo serie di caratteri codificati origine non valido.

ERRORE DI RIMOZIONE MQRC_SOURCE_DECIMAL_ENC_
(2113, X'841 ') Codifica decimale compresso nel messaggio non riconosciuta.

ERRORE_ERRORE_ORIGINE_RISORSE MQRC
(2114, X'842 ') La codifica a virgola mobile nel messaggio non è stata riconosciuta.

ERRORE DI INIZIALIZZAZIONE MQRC_SOURCE_INTEGER_
(2112, X'840 ') Numero intero di origine non riconosciuto.

ERRORE_ORIGINE_RISORSE MQRC
(2143, X'85F') Parametro di lunghezza origine non valido.

ERRORE - BUFFER_MQRC_TARGET_
(2146, X'862 ') Parametro buffer di destinazione non valido.

ERRORE MQRC_TARGET_CCSID_
(2115, X'843 ') Identificativo serie di caratteri codificati di destinazione non valido.

ERRORE DI RETE MQRC_TARGET_DECIMAL_ENC_ERROR
(2117, X'845 ') La codifica decimale compresso specificata dal ricevitore non è stata riconosciuta.

ERRORE DI RETE MQRC_TARGET_FLOAT_
(2118, X'846 ') La codifica a virgola mobile specificata dal ricevitore non è stata riconosciuta.

ERRORE MQRC_TARGET_INTEGER_ENC
(2116, X'844 ') Codifica numero intero di destinazione non riconosciuta.

MQRC_TRUNCATED_MSG_ACCEPTED
(2079, X'81F') Messaggio troncato restituito (elaborazione completata).

MQRC_TRUNCATED_MSG_NON RIUSCITO
(2080, X'820 ') Messaggio troncato restituito (elaborazione non completata).

Se *CompCode* è MQCC_FAILED:

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'89C') Adattatore non disponibile.

ERRORE CARICAMENTO MQRC_ADAPTER_CONV
(2133, X'855 ') Impossibile caricare i moduli dei servizi di conversione dati.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita.

ERRORE USCITA MQRC_API
(2183, X'887 ') Impossibile caricare l'uscita API.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

MQRC_BACK_OUT
(2003, X'7D3') Unità di lavoro ripristinata.

ERRORE MQRC_BUFFER_
(2004, X'7D4') Parametro del buffer non valido.

ERRORE MQRC_BUFFER_LENGTH
(2005, X'7D5') Parametro di lunghezza del buffer non valido.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

MQRC_CF_NOT_AVAILABLE
(2345, X' 929 ') La funzione di accoppiamento non è disponibile.

MQRC_CF_STRUC_NON RIUSCITO
(2373, X' 945 ') La struttura della funzione di accoppiamento non è riuscita.

MQRC_CF_STRUC_IN_USO
(2346, X'92A') Struttura CF in uso.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') L'elenco - intestazione della struttura CFS (Coupling Facility Structure) è in uso.

MQRC_CICS_WAIT_NON RIUSCITO
(2140, X'85C') Richiesta di attesa rifiutata da CICS.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Non autorizzato per la connessione.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Connessione in fase di sospensione.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Chiusura della connessione.

ERRORE ID CORREL_MQR_
(2207, X'89F') Errore identificativo di correlazione.

ERRORE MQRC_DATA_LENGTH
(2010, X'7DA') Parametro di lunghezza dati non valido.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') Db2 sottosistema non disponibile.

MQRC_GET_INHIBITED
(2016, X'7E0') Ottiene inibiti per la coda.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Unità globali di conflitto di lavoro.

ERRORE MQRC_GMO
(2186, X'88A') Struttura delle opzioni Get - message non valida.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X' 931 ') Handle in uso per l'unità di lavoro globale.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

ERRORE MQRC_HOBJ_R
(2019, X'7E3') Handle oggetto non valido.

BROWSE INCONSIST_MQRC
(2259, X'8D3') Specifica di ricerca incongruente.

UOW MQRC_INCONSISTENT_
(2245, X'8C5') Specifica dell'unità di lavoro non congruente.

MQRC_INVALID_MSG_UNDER_CURSOR
(2246, X'8C6') Messaggio sotto il cursore non valido per il recupero.

MQRC_LOCAL_UOW_CONFLICT
(2352, X' 930 ') L'unità di lavoro globale è in conflitto con l'unità di lavoro locale.

ERRORE MQRC_MATCH_OPTIONS_
(2247, X'8C7') Opzioni di corrispondenza non valide.

ERRORE MQRC_MD
(2026, X'7EA') Descrittore messaggio non valido.

ERRORE ID MQRC_MSG_
(2206, X'89E') Errore identificativo messaggio.

ERRORE MQRC_MSG_SEQ_NUMBER_
(2250, X'8CA') Numero di sequenza messaggio non valido.

ERRORE MQRC_MSG_TOKEN_
(2331, X'91B') L'utilizzo del token del messaggio non è valido.

MQRC_NO_MSG_AVAILABLE
(2033, X'7F1') Nessun messaggio disponibile.

MQRC_NO_MSG_UNDER_CURSOR
(2034, X'7F2') Il cursore di ricerca non è posizionato sul messaggio.

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') Coda non aperta per la ricerca.

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') Coda non aperta per l'input.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Definizione oggetto modificata dall'apertura.

MQRC_OBJECT_DAMAGED
(2101, X'835 ') Oggetto danneggiato.

ERRORE MQRC_OPTIONS_
(2046, X'7FE') Opzioni non valide o non congruenti.

ERRORE MQRC_PAGESET_
(2193, X'891 ') Errore durante l'accesso al dataset della serie di pagine.

MQRC_Q_XX_ENCODE_CASE_ONE eliminato
(2052, X'804 ') La coda è stata eliminata.

MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') La coda ha un tipo di indice errato.

ERRORE MQRC_Q_MGR_NAME_
(2058, X'80A') Nome gestore code non valido o sconosciuto.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Gestore code non disponibile per la connessione.

MQRC_Q_MGR_QUIESCING
(2161, X'871 ') Gestore code in fase di sospensione.

MQRC_Q_MGR_STOPPING
(2162, X'872 ') Chiusura del gestore code.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

MQRC_SECOND_MARK_NON_CONSENTITO
(2062, X'80E') Un messaggio è già contrassegnato.

MQRC_SIGNAL_OUTSTANDING
(2069, X'815 ') Segnale eccezionale per questa maniglia.

MQRC_SIGNAL1_ERROR
(2099, X'833 ') Campo segnale non valido.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890 ') Il supporto di memoria esterna è pieno.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Chiamata eliminata dal programma di uscita.

MQRC_SYNCPOINT_LIMITE_RAGGIUNTO
(2024, X'7E8') Non è possibile gestire ulteriori messaggi all'interno dell'unità di lavoro corrente.

MQRC_SYNCPOINT_NOT_AVAILABLE
Supporto punto di sincronizzazione (2072, X'818 ') non disponibile.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

ERRORE MQRC_UOW_ENLISTMENT_ERROR
(2354, X' 932 ') L'inserimento nell'unità di lavoro globale non è riuscito.

MQRC_UOW_MIX_NON_SUPPORTATO
(2355, X' 933 ') La miscelazione delle chiamate UOW non è supportata.

MQRC_UOW_NON_DISPONIBILE
(2255, X'8CF') Unità di lavoro non disponibile per il gestore code.

ERRORE INTERVAL_WAIT_MQRC
(2090, X'82A') Intervallo di attesa in MQGMO non valido.

MQRC_WRONG_GMO_VERSIONE
(2256, X'8D0') Versione errata di MQGMO fornita.

MQRC_WRONG_MD_VERSIONE
(2257, X'8D1') Versione non corretta di MQMD fornita.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note d'utilizzo

1. Il messaggio richiamato viene normalmente eliminato dalla coda. Questa eliminazione può verificarsi come parte della chiamata MQGET stessa o come parte di un punto di sincronizzazione.

   Le opzioni di ricerca sono: MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT e MQGMO_BROWSE_MSG_UNDER_CURSOR.

2. Se l'opzione MQGMO_LOCK viene specificata con una delle opzioni di ricerca, il messaggio visualizzato viene bloccato in modo che sia visibile solo a questo handle.

   Se viene specificata l'opzione MQGMO_UNLOCK, un messaggio precedentemente bloccato viene sbloccato. In questo caso, non viene richiamato alcun messaggio e i parametri **MsgDesc**, **BufferLength**, **Buffer**e **DataLength** non vengono controllati o modificati.

3. Per le applicazioni che emettono una chiamata MQGET, il messaggio richiamato può essere perso se l'applicazione termina in modo anomalo o se la connessione viene interrotta durante l'elaborazione della chiamata. Questo problema si verifica perché il surrogato in esecuzione sulla stessa piattaforma del gestore code che emette la chiamata MQGET per conto dell'applicazione non può rilevare la perdita dell'applicazione fino a quando il surrogato non sta per restituire il messaggio all'applicazione, dopo che il messaggio è stato rimosso dalla coda. Questo problema può verificarsi sia per i messaggi persistenti che per quelli non persistenti.

Per eliminare il rischio di perdere i messaggi in questo modo, richiamare sempre i messaggi all'interno delle unità di lavoro. Vale a dire, specificando l'opzione MQGMO_SYNCPOINT sulla chiamata MQGET e utilizzando le chiamate MQCMIT o MQBACK per eseguire il commit o il backout dell'unità di lavoro quando l'elaborazione del messaggio è completa. Se viene specificato MQGMO_SYNCPOINT e il client viene terminato in modo anomalo o la connessione viene interrotta, il surrogato esegue il backout dell'unità di lavoro sul gestore code e il messaggio viene reintegrato nella coda. Per ulteriori informazioni sui punti di sincronizzazione, consultare Considerazioni sui punti di sincronizzazione nelle applicazioni IBM MQ.

Questa situazione può verificarsi con i client IBM MQ e con le applicazioni in esecuzione sulla stessa piattaforma del gestore code.

4. Se un'applicazione inserisce una sequenza di messaggi su un particolareall'interno di una singola unità di lavoro e quindi esegue il commit di tale unità di lavoro correttamente, i messaggi diventano disponibili per il richiamo nel modo seguente:
   - Se la coda è una *coda non condivisa* (ossia, una coda locale), tutti i messaggi all'interno dell'unità di lavoro diventano disponibili contemporaneamente.
   - Se la coda è una *coda condivisa*, i messaggi all'interno dell'unità di lavoro diventano disponibili nell'ordine in cui sono stati inseriti, ma non tutti contemporaneamente. Quando il sistema è molto carico, è possibile che il primo messaggio nell'unità di lavoro venga richiamato correttamente, ma che la chiamata MQGET per il secondo messaggio o il messaggio successivo nell'unità di lavoro abbia esito negativo con MQRC_NO_MSG_AVAILABLE. Se si verifica questo problema, l'applicazione deve attendere un breve periodo e riprovare l'operazione.
5. Se un'applicazione inserisce una sequenza di messaggi nella stessa coda senza utilizzare i gruppi di messaggi, l'ordine di tali messaggi viene conservato se sono soddisfatte determinate condizioni. Consultare Note sull'utilizzo di MQPUT per i dettagli. Se le condizioni sono soddisfatte, i messaggi vengono presentati all'applicazione ricevente nell'ordine in cui sono stati inviati, se:
   - Solo un destinatario riceve i messaggi dalla coda.

     Se ci sono due o più applicazioni che ricevono i messaggi dalla coda, devono concordare con il mittente il meccanismo da utilizzare per identificare i messaggi che appartengono a una sequenza. Ad esempio, il mittente potrebbe impostare tutti i campi `CorrelId` nei messaggi in una sequenza su un valore univoco per tale sequenza di messaggi.
   - Il destinatario non modifica deliberatamente l'ordine di richiamo, ad esempio specificando un particolare `MsgId` o `CorrelId`.

   Se l'applicazione mittente inserisce i messaggi come gruppo di messaggi, i messaggi vengono presentati all'applicazione ricevente nell'ordine corretto se l'applicazione ricevente specifica l'opzione MQGMO_LOGICAL_ORDER sulla chiamata MQGET. Per ulteriori informazioni sui gruppi di messaggi, consultare:
   - Campo MQMD - MsgFlags
   - MQPMO_LOGICAL_ORDER
   - MQGMO_LOGICAL_ORDER

   Se l'utente riceve i messaggi in un gruppo nel punto di sincronizzazione, deve assicurarsi che il gruppo completo venga elaborato prima di tentare di terminare la transazione.
6. Le applicazioni devono verificare il codice di feedback MQFB_QUIT nel campo `Feedback` del parametro **MsgDesc** e terminare se trovano questo valore. Per ulteriori informazioni, consultare il campo MQMD - Feedback .
7. Se la coda identificata da `Hobj` è stata aperta con l'opzione MQOO_SAVE_ALL_CONTEXT e il codice di completamento dalla chiamata MQGET è MQCC_OK o MQCC_WARNING, il contesto associato all'handle della coda `Hobj` è impostato sul contesto del messaggio che è stato richiamato (a meno che non sia impostata l'opzione MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT o MQGMO_BROWSE_MSG_UNDER_CURSOR, nel qual caso il contesto è contrassegnato come non disponibile).

È possibile utilizzare il contesto salvato su una chiamata MQPUT o MQPUT1 successiva specificando le opzioni MQPMO_PASS_IDENTITY_CONTEXT o MQPMO_PASS_ALL_CONTEXT. Ciò consente al contesto del messaggio ricevuto di essere trasferito in tutto o in parte a un altro messaggio (ad esempio, quando il messaggio viene inoltrato a un'altra coda). Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio.

8. Se si include l'opzione MQGMO_CONVERT nel parametro **GetMsgOpts** , i dati del messaggio dell'applicazione vengono convertiti nella rappresentazione richiesta dall'applicazione ricevente, prima che i dati vengano inseriti nel parametro **Buffer** :

   - Il campo `Format` nelle informazioni di controllo nel messaggio identifica la struttura dei dati dell'applicazione e i campi `CodedCharSetId` e `Encoding` nelle informazioni di controllo nel messaggio specificano l'identificativo della serie di caratteri e la codifica.
   - L'applicazione che emette la chiamata MQGET specifica nei campi `CodedCharSetId` e `Encoding` nel parametro **MsgDesc** l'identificativo della serie di caratteri e la codifica in cui convertire i dati del messaggio dell'applicazione.

   Quando la conversione dei dati del messaggio è necessaria, la conversione viene eseguita dal gestore code stesso o da un'uscita scritta dall'utente, a seconda del valore del campo `Format` nelle informazioni di controllo nel messaggio:

   - I seguenti nomi di formato sono formati convertiti dal gestore code; questi formati sono denominati formati "integrati":
     - MMQFMT_ADMIN
     - MQFMT_CICS (solo z/OS)
     - MQFMT_COMMAND_1
     - MQFMT_COMMAND_2
     - MQFMT_DEAD_LETTER_HEADER
     - INTESTAZIONE_DIST_MQFM
     - MQFMT_EVENT versione 1
     - MQFMT_EVENT versione 2 (solo z/OS)
     - IMS MQFMT
     - MQFMT_IMS_VAR_STRING
     - MQFMT_MD_EXTENSIONE
     - MQFMT_PCF
     - MQFMT_REF_MSG_HEADER
     - MQFMT_RF_HEADER
     - MQFMT_RF_HEADER_2
     - MQFMT_STRING
     - MQFMT_TRIGGER
     - MQFMT_WORK_INFO_HEADER (solo z/OS)
     - MQFMT_XMIT_Q_HEADER
   - Il nome formato MQFMT_NONE è un valore speciale che indica che la natura dei dati nel messaggio non è definita. Di conseguenza, il gestore code non tenta la conversione quando il messaggio viene richiamato dalla coda.

     Nota: Se MQGMO_CONVERT viene specificato sulla chiamata MQGET per un messaggio che ha un nome formato MQFMT_NONE e la serie di caratteri o la codifica del messaggio differiscono da quella specificata nel parametro **MsgDesc** , il messaggio viene restituito nel parametro **Buffer** (supponendo che non vi siano altri errori), ma la chiamata viene completata con il codice di completamento MQCC_WARNING e il codice motivo MQRC_FORMAT_ERROR.

È possibile utilizzare MQFMT_NONE quando la natura dei dati del messaggio indica che non richiede la conversione o quando le applicazioni di invio e ricezione hanno concordato tra loro il modulo in cui inviare i dati del messaggio.

- Tutti gli altri nomi di formato inoltrano il messaggio ad un'uscita scritta dall'utente per la conversione. L'uscita ha lo stesso nome del formato, oltre alle aggiunte specifiche dell'ambiente. I nomi formato specificati dall'utente non devono iniziare con le lettere IBM MQ.

  Consultare "Uscita conversione dati" a pagina 932 per i dettagli dell'uscita di conversione dati.

I dati utente nel messaggio possono essere convertiti tra tutte le serie di caratteri e le codifiche supportate. Tuttavia, tenere presente che, se il messaggio contiene una o più strutture di intestazione IBM MQ , il messaggio non può essere convertito da o in una serie di caratteri che ha caratteri a doppio byte o a più byte per uno qualsiasi dei caratteri validi nei nomi delle code. Il codice di errore MQRC_SOURCE_CCSID_ERROR o MQRC_TARGET_CCSID_ERROR risulta se viene tentato e il messaggio viene restituito non convertito. La serie di caratteri Unicode UTF-16 è un esempio di tale serie di caratteri.

Al ritorno da MQGET, il seguente codice di errore indica che il messaggio è stato convertito correttamente:

- MQRC_NONE

Il seguente codice di errore indica che il messaggio potrebbe essere stato convertito correttamente; l'applicazione deve controllare i campi `CodedCharSetId` e `Encoding` nel parametro **MsgDesc** per scoprire:

- MQRC_TRUNCATED_MSG_ACCEPTED

Tutti gli altri codici di errore indicano che il messaggio non è stato convertito.

Nota: L'interpretazione di questo codice di errore è vera per le conversioni eseguite da un'uscita scritta dall'utente solo se l'uscita è conforme alle linee guida di elaborazione descritte in "Uscita conversione dati" a pagina 932.

9. Quando si utilizza l'interfaccia orientata agli oggetti per richiamare i messaggi, è possibile scegliere di non specificare un buffer per contenere i dati del messaggio per una chiamata MQGET. Quando si riceve un messaggio utilizzando un'applicazione orientata agli oggetti senza limitare la dimensione del buffer di ricezione messaggi, l'applicazione non ha esito negativo con MQRC_CONVERTED_MSG_TOO_BIG e riceve il messaggio convertito. Ciò è vero per i seguenti ambienti:

   - .NET, incluse le applicazioni completamente gestite
   - C++
   - Java (IBM MQ classes for Java)

   Nota: Per tutti i client, se il valore di `sharingConversations` è zero e se il buffer è troppo piccolo per ricevere il messaggio convertito, viene restituito il messaggio non convertito, con codice motivo MQRC_CONVERTED_MSG_TOO_BIG. Per ulteriori informazioni su `sharingConversations`, consultare Utilizzo della condivisione di conversazioni in un'applicazione client.

10. Per i formati integrati, il gestore code può eseguire la *conversione predefinita* delle stringhe di caratteri nel messaggio quando viene specificata l'opzione MQGMO_CONVERT. La conversione predefinita consente al gestore code di utilizzare una serie di caratteri predefinita specificata dall'installazione che si avvicina alla serie di caratteri effettiva, durante la conversione dei dati stringa. Di conseguenza, la chiamata MQGET può essere eseguita correttamente con codice di completamento MQCC_OK, invece di essere completata con MQCC_WARNING e codice motivo MQRC_SOURCE_CCSID_ERROR o MQRC_TARGET_CCSID_ERROR.

    Nota: Il risultato dell'utilizzo di una serie di caratteri approssimativa per convertire i dati stringa è che alcuni caratteri potrebbero essere convertiti in modo non corretto. Per evitare ciò, utilizzare i caratteri nella stringa che sono comuni sia alla serie di caratteri effettiva che alla serie di caratteri predefinita.

    La conversione predefinita si applica ai dati del messaggio dell'applicazione e ai campi carattere nelle strutture MQMD e MQMDE:

- La conversione predefinita dei dati del messaggio dell'applicazione si verifica solo quando tutte le seguenti istruzioni sono vere:
  - L'applicazione specifica MQGMO_CONVERT.
  - Il messaggio contiene dati che devono essere convertiti da o in una serie di caratteri non supportata.
  - La conversione predefinita è stata abilitata quando il gestore code è stato installato o riavviato.
- La conversione predefinita dei campi di caratteri nelle strutture MQMD e MQMDE si verifica come necessario, se la conversione predefinita è abilitata per il gestore code. La conversione viene eseguita anche se l'opzione MQGMO_CONVERT non è specificata dall'applicazione nella chiamata MQGET.

11. Per il linguaggio di programmazione Visual Basic, si applicano i punti seguenti:
    - Se la dimensione del parametro **Buffer** è inferiore alla lunghezza specificata dal parametro **BufferLength** , la chiamata ha esito negativo con codice di errore MQRC_STORAGE_NOT_AVAILABLE.
    - Il parametro **Buffer** viene dichiarato come di tipo `String`. Se i dati da richiamare dalla coda non sono di tipo `String`, utilizzareChiamata MQGETAny al posto di MQGET.

      La chiamata MQGETAny ha gli stessi parametri della chiamata MQGET, tranne per il fatto che il parametro **Buffer** è dichiarato di tipo `Any`, consentendo il richiamo di qualsiasi tipo di dati. Tuttavia, ciò significa che `Buffer` non può essere controllato per garantire che abbia una dimensione di almeno `BufferLength` byte.
12. Non tutte le opzioni MQGET sono supportate quando la lettura anticipata è abilitata. La seguente tabella indica quali opzioni sono consentite e se possono essere modificate tra le chiamate MQGET.

Tabella 548. Opzioni MQGET consentite quando la lettura anticipata è abilitata

	Consentito quando la lettura anticipata è abilitata e può essere modificato tra le chiamate MQGET	**Consentito quando la lettura anticipata è abilitata ma non può essere modificato tra le chiamate MQGET** [a]	**Le opzioni MQGET non consentite quando la lettura anticipata è abilitata** [b]
Valori MQGET MD	MsgId [c] CorrelId [c]	Codifica CodedCharSetId	
Opzioni MQGET MQGMO	MQGMO_WAIT MQGMO_NO_WAIT MQGMO_FAIL_IF_QUIESCING MQGMO_BROWSE_FIRST [g] MQGMO_BROWSE_NEXT [g] MESSAGGIO_BROWSE_MQGMO_ _UNDER_CURSOR [g]	MQGMO_SYNCPOINT_IF_PERSISTENT MQGMO_NO_SYNCPOINT MQGMO_ACCEPT_TRUNCATED_MSG MQGMO_CONVERT ORDER LOGICAL_MQGMO_ MQGMO_COMPLETE_MSG MQGMO_ALL_MSGS_AVAILABLE MQGMO_ALL_SEGMENTS_AVAILABLE MQGMO_MARK_BROWSE_HANDLE MQGMO_MARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_HANDLE MQGMO_UNMARKED_BROWSE_MSG MQGMO_PROPERTIES_FORCE_MQRFH2 MQGMO_NO_PROPERTIES MQGMO_PROPERTIES_IN_HANDLE MQGMO_PROPERTIES_COMPATIBILITY	MQGMO_SET_SIGNAL SYNCPOINT MQGMO SKIP_MARK_MQGMO _BACKOUT MQGMO_MSG_UNDER _CURSORE [d] LOCK_MQGMO MQGMO_UNLOCK
Valori MQGMO		MsgHandle	

a. Se queste opzioni vengono modificate tra le chiamate MQGET, viene restituito un codice motivo MQRC_OPTIONS_CHANGED.

b. Se queste opzioni vengono specificate nella prima chiamata MQGET, il read ahead è disabilitato. Se queste opzioni vengono specificate in una successiva chiamata MQGET, viene restituito il codice motivo MQRC_OPTIONS_ERROR.

c. Le applicazioni client devono essere consapevoli che se i valori MsgId e CorrelId vengono modificati tra le chiamate MQGET, i messaggi con i valori precedenti potrebbero essere già stati inviati al client e rimanere nel buffer di lettura anticipata del client finché non vengono consumati (o automaticamente eliminati).

d. La prima chiamata MQGET determina se i messaggi devono essere ricercati o richiamati da una coda quando il read ahead è abilitato. Se l'applicazione tenta di utilizzare una combinazione di BROWSE e GET, viene restituito il codice motivo MQRC_OPTIONS_CHANGED.

e. MQGMO_MSG_UNDER_CURSOR non è consentito con il read ahead. Quando il read ahead è abilitato, i messaggi possono essere sfogliati o richiamati, ma non entrambe le cose.

13. Le applicazioni possono ottenere in modo distruttivo i messaggi di cui non è stato eseguito il commit solo se tali messaggi vengono inseriti nella stessa unità di lavoro locale del get. Le applicazioni non possono ricevere messaggi senza commit in modo non distruttivo.
14. I messaggi sotto un cursore di ricerca possono essere richiamati in un'unità di lavoro. Non è possibile richiamare un messaggio di cui non è stato eseguito il commit in questo modo.

Richiamo C

```
MQGET (Hconn, Hobj, &MsgDesc, &GetMsgOpts, BufferLength, Buffer,
      &DataLength, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;         /* Connection handle */
MQHOBJ   Hobj;          /* Object handle */
MQMD     MsgDesc;       /* Message descriptor */
MQGMO    GetMsgOpts;    /* Options that control the action of MQGET */
MQLONG   BufferLength;  /* Length in bytes of the Buffer area */
MQBYTE   Buffer[n];     /* Area to contain the message data */
MQLONG   DataLength;    /* Length of the message */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQGET' USING HCONN, HOBJ, MSGDESC, GETMSGOPTS, BUFFERLENGTH,
BUFFER, DATALENGTH, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Object handle
 01  HOBJ          PIC S9(9) BINARY.
**   Message descriptor
 01  MSGDESC.
     COPY CMQMDV.
**   Options that control the action of MQGET
 01  GETMSGOPTS.
     COPY CMQGMOV.
**   Length in bytes of the BUFFER area
 01  BUFFERLENGTH  PIC S9(9) BINARY.
**   Area to contain the message data
 01  BUFFER        PIC X(n).
**   Length of the message
 01  DATALENGTH    PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE      PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQGET (Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer,
           DataLength, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn         fixed bin(31);  /* Connection handle */
dcl Hobj          fixed bin(31);  /* Object handle */
dcl MsgDesc       like MQMD;      /* Message descriptor */
dcl GetMsgOpts    like MQGMO;     /* Options that control the action of
                                     MQGET */
```

```
dcl BufferLength  fixed bin(31);  /* Length in bytes of the Buffer
                                     area */
dcl Buffer        char(n);        /* Area to contain the message data */
dcl DataLength    fixed bin(31);  /* Length of the message */
dcl CompCode      fixed bin(31);  /* Completion code */
dcl Reason        fixed bin(31);  /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
         CALL MQGET,(HCONN,HOBJ,MSGDESC,GETMSGOPTS,BUFFERLENGTH,
               BUFFER,DATALENGTH,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN         DS       F      Connection handle
HOBJ          DS       F      Object handle
MSGDESC       CMQMDA   ,      Message descriptor
GETMSGOPTS    CMQGMOA  ,      Options that control the action of MQGET
BUFFERLENGTH  DS       F      Length in bytes of the BUFFER area
BUFFER        DS       CL(n)  Area to contain the message data
DATALENGTH    DS       F      Length of the message
COMPCODE      DS       F      Completion code
REASON        DS       F      Reason code qualifying COMPCODE
```

Richiamo Visual Basic

```
MQGET Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer,
DataLength, CompCode, Reason
```

Dichiarare i parametri come segue:

```
Dim Hconn        As Long   'Connection handle'
Dim Hobj         As Long   'Object handle'
Dim MsgDesc      As MQMD   'Message descriptor'
Dim GetMsgOpts   As MQGMO  'Options that control the action of MQGET'
Dim BufferLength As Long   'Length in bytes of the Buffer area'
Dim Buffer       As String 'Area to contain the message data'
Dim DataLength   As Long   'Length of the message'
Dim CompCode     As Long   'Completion code'
Dim Reason       As Long   'Reason code qualifying CompCode'
```

MQINQ - Richiedi attributi oggetto

La chiamata MQINQ restituisce un array di numeri interi e una serie di stringhe di carattere contenenti attributi di un oggetto.

Sono validi i seguenti tipi di oggetto:

- Gestore code
- Coda
- Elenco nomi
- Definizione di processo

Sintassi

MQINQ (*Hconn*, *Hobj*, *SelectorCount*, *Selettori*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *CompCode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

Su z/OS per applicazioni CICS la chiamata MQCONN può essere omessa e il seguente valore specificato per *Hconn*:

MQHC_DEF_HCONN
Handle di connessione predefinito.

HOBJ
Tipo: MQHOBJ - input

Questo handle rappresenta l'oggetto (di qualsiasi tipo) con gli attributi richiesti. L'handle deve essere restituito da una precedente chiamata MQOPEN che specificava l'opzione MQOO_INQUIRE .

SelectorCount
Tipo: MQLONG - input

Questo è il conteggio dei selettori forniti nell'array *Selectors* . È il numero di attributi da restituire. Zero è un valore valido. Il numero massimo consentito è 256.

Selettori
Tipo: MQLONG x *SelectorCount* - input

Si tratta di un array di selettori di attributi **SelectorCount** ; ogni selettore identifica un attributo (numero intero o carattere) con un valore richiesto.

Ogni selettore deve essere valido per il tipo di oggetto che *Hobj* rappresenta, altrimenti la chiamata non riesce con codice di completamento MQCC_FAILED e codice motivo MQRC_SELECTOR_ERROR.

Nel caso speciale delle code:

- Se il selettore non è valido per le code di qualsiasi tipo, la chiamata ha esito negativo con codice di completamento MQCC_FAILED e codice motivo MQRC_SELECTOR_ERROR.
- Se il selettore si applica solo alle code di tipi diversi dal tipo dell'oggetto, la chiamata riesce con il codice di completamento MQCC_WARNING e il codice motivo MQRC_SELECTOR_NOT_FOR_TYPE.
- Se la coda oggetto della richiesta è una coda cluster, i selettori validi dipendono da come è stata risolta la coda; Vedere"Note d'utilizzo" a pagina 738 per ulteriori dettagli.

È possibile specificare i selettori in qualsiasi ordine. I valori degli attributi che corrispondono ai selettori di attributi interi (selettori MQIA_*) vengono restituiti in *IntAttrs* nello stesso ordine in cui si verificano in *Selectors*. I valori degli attributi che corrispondono ai selettori degli attributi dei caratteri (selettori MQCA_*) vengono restituiti in *CharAttrs* nello stesso ordine in cui si verificano tali selettori. I selettori MQIA_* possono essere intercalati con i selettori MQCA_* ; è importante solo l'ordine relativo all'interno di ciascun tipo.

Nota:

1. I selettori degli attributi integer e character sono assegnati in due intervalli differenti; i selettori MQIA_* si trovano nell'intervallo MQIA_FIRST - MQIA_LASTe i selettori MQCA_* nell'intervallo MQCA_FIRST - MQCA_LAST.

   Per ogni intervallo, le costanti MQIA_LAST_USED e MQCA_LAST_USED definiscono il valore più alto accettato dal gestore code.
2. Se tutti i selettori MQIA_* si verificano per primi, è possibile utilizzare gli stessi numeri elemento per indirizzare gli elementi corrispondenti negli array *Selectors* e *IntAttrs* .
3. Se il parametro **SelectorCount** è zero, non si fa riferimento a *Selectors* . In questo caso, l'indirizzo del parametro passato dai programmi scritti nell'assembler C o S/390 potrebbe essere null.

Gli attributi che è possibile interrogare sono elencati nelle seguenti tabelle. Per i selettori MQCA_* , la costante che definisce la lunghezza in byte della stringa risultante in *CharAttrs* viene fornita tra parentesi.

Le tabelle che seguono elencano i selettori, per oggetto, in ordine alfabetico, come segue:

- Tabella 549 a pagina 725MQINQselettori di attributo per le code
- Tabella 550 a pagina 727MQINQselettori di attributi per elenchi di nomi
- Tabella 551 a pagina 728MQINQselettori di attributo per le definizioni di processo
- Tabella 552 a pagina 728MQINQselettori di attributo per il gestore code

Tabella 549. Selettori di attributo MQINQ per code

Selettore	Lunghezza del campo	Descrizione
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Data della modifica più recente
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Ora della modifica più recente
MQCA_BACKOUT_REQ_Q_NAME	MQ_Q_NAME_LENGTH	Nome riaccodamento di backout eccessivo
MQCA_BASE_Q_NAME	MQ_Q_NAME_LENGTH	Nome della coda in cui si risolve l'alias
z/OS MQCA_CF_STRUC_NAME	MQ_CF_STRUC_NAME_LENGTH	Nome CF
MQCA_CLUS_CHL_NAME	MQ_CHANNEL_NAME_LENGTH	Nome del canale mittente del cluster che utilizza questa coda come coda di trasmissione.
MQCA_CLUSTER_NAME	MQ_CLUSTER_NAME_LENGTH	Nome cluster
MQCA_CLUSTER_NAMELIST	MQ_NAMELIST_NAME_LENGTH	Elenco nomi cluster
MQCA_CREATION_DATE	MQ_CREATION_DATE_LENGTH	Data di creazione della coda
MQCA_CREATION_TIME	MQ_CREATION_TIME_LENGTH	Ora di creazione della coda
MQCA_CUSTOM	MQ_CUSTOM_LENGTH	L'attributo personalizzato per le nuove funzioni
MQCA_INITIATION_Q_NAME	MQ_Q_NAME_LENGTH	Nome coda di attivazione
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	Nome della definizione del processo
MQCA_Q_DESC	MQ_Q_DESC_LENGTH	Descrizione coda
MQCA_Q_NAME	MQ_Q_NAME_LENGTH	Nome coda
MQCA_REMOTE_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	Nome del gestore code remoto
MQCA_REMOTE_Q_NAME	MQ_Q_NAME_LENGTH	Nome della coda remota come noto sul gestore code remoto
z/OS MQCA_STORAGE_CLASS	MQ_STORAGE_CLASS_LENGTH	Nome della classe di archiviazione
MQCA_TRIGGER_DATA	MQ_TRIGGER_DATA_LENGTH	Dati trigger

Tabella 549. Selettori di attributo MQINQ per code (Continua)		
Selettore	**Lunghezza del campo**	**Descrizione**
MQCA_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	Nome coda di trasmissione
Multi MQIA_ACCOUNTING_Q	MQLONG	Controlla la raccolta dei dati di account per la coda
MQIA_BACKOUT_THRESHOLD	MQLONG	Soglia di ripristino
MQIA_CLWL_Q_PRIORITY	MQLONG	Priorità della coda
MQIA_CLWL_Q_RANK	MQLONG	Classificazione della coda
MQIA_CLWL_USEQ	MQLONG	Utilizza code remote
MQIA_CURRENT_Q_DEPTH	MQLONG	Numero di messaggi in coda
MQIA_DEF_BIND	MQLONG	Binding predefinito
MQIA_DEF_INPUT_OPEN_OPTION	MQLONG	Opzione open - for - input predefinita
MQIA_DEF_PERSISTENCE	MQLONG	Persistenza predefinita messaggio
MQIA_DEF_PRIORITY	MQLONG	Priorità predefinita messaggio
MQIA_DEFINITION_TYPE	MQLONG	Il tipo di definizione della coda
Multi MQIA_DIST_LISTS	MQLONG	Supporto elenco di distribuzione
MQIA_HARDEN_GET_BACKOUT	MQLONG	Indica se rafforzare il conteggio di backout
z/OS MQIA_INDEX_TYPE	MQLONG	Tipo di indice gestito per la coda
MQIA_INHIBIT_GET	MQLONG	Se le operazioni get sono consentite
MQIA_INHIBIT_PUT	MQLONG	Se le operazioni di inserimento sono consentite
MQIA_MAX_MSG_LENGTH	MQLONG	Lunghezza massima dei messaggi
MQIA_MAX_Q_DEPTH	MQLONG	Numero massimo di messaggi consentiti nella coda
MQIA_MSG_DELIVERY_SEQUENCE	MQLONG	Se la priorità del messaggio è rilevante
MQIA_NPM_CLASS	MQLONG	Livello di affidabilità per i messaggi non persistenti
MQIA_OPEN_INPUT_COUNT	MQLONG	Numero di chiamate MQOPEN con la coda aperta per l'input
MQIA_OPEN_OUTPUT_COUNT	MQLONG	Numero di chiamate MQOPEN con la coda aperta per l'output
MQIA_PROPERTY_CONTROL	MQLONG	Attributo controllo proprietà
Multi MQIA_Q_DEPTH_HIGH_EVENT	MQLONG	Attributo di controllo per gli eventi di grandezza della coda elevata
Multi MQIA_Q_DEPTH_HIGH_LIMIT	MQLONG	Limite massimo per la profondità della coda
Multi MQIA_Q_DEPTH_LOW_EVENT	MQLONG	Attributo di controllo per gli eventi di profondità della coda bassa

Tabella 549. Selettori di attributo MQINQ per code (Continua)		
Selettore	**Lunghezza del campo**	**Descrizione**
Multi MQIA_Q_DEPTH_LOW_LIMIT	MQLONG	Limite inferiore per la profondità della coda
Multi MQIA_Q_DEPTH_MAX_EVENT	MQLONG	Attributo di controllo per il numero massimo di eventi di profondità della coda
Multi MQIA_Q_SERVICE_INTERVAL	MQLONG	Limite per intervallo di servizio coda
Multi MQIA_Q_SERVICE_INTERVAL_EVENT	MQLONG	Attributo di controllo per gli eventi dell'intervallo di servizio della coda
MQIA_Q_TYPE	MQLONG	Tipo coda
z/OS MQIA_QSG_DISP	MQLONG	Disposizione del gruppo di condivisione code
MQIA_RETENTION_INTERVAL	MQLONG	Intervallo di conservazione coda
Multi MQIA_SCOPE	MQLONG	Ambito definizione coda
MQIA_SHAREABILITY	MQLONG	Indica se la coda può essere condivisa per l'input
Multi MQIA_STATISTICS_Q	MQLONG	Controlla la raccolta dei dati statistici per la coda
MQIA_TRIGGER_CONTROL	MQLONG	Controllo trigger
MQIA_TRIGGER_DEPTH	MQLONG	Capacità di Trigger
MQIA_TRIGGER_MSG_PRIORITY	MQLONG	Priorità messaggi di soglia per i trigger
MQIA_TRIGGER_TYPE	MQLONG	Tipo trigger
MQIA_USAGE	MQLONG	Utilizzo

Tabella 550. Selettori di attributo MQINQ per gli elenchi nomi		
Selettore	**Lunghezza del campo**	**Descrizione**
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Data della modifica più recente
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Ora della modifica più recente
MQCA_NAMELIST_DESC	MQ_NAMELIST_DESC_LENGTH	Descrizione elenco nomi
MQCA_NAMELIST_NAME	MQ_NAMELIST_NAME_LENGTH	Nome dell'oggetto elenco nomi
z/OS MQIA_NAMELIST_TYPE	MQLONG	Tipo di elenco nomi
MQCA_NAMES	MQ_Q_NAME_LENGTH x *Number of names in the list*	Nomi nell'elenco nomi

Tabella 550. Selettori di attributo MQINQ per gli elenchi nomi (Continua)		
Selettore	**Lunghezza del campo**	**Descrizione**
MQIA_NAME_COUNT	MQLONG	Numero di nomi nell'elenco nomi
z/OS MQIA_QSG_DISP	MQLONG	Disposizione del gruppo di condivisione code

Tabella 551. Selettori di attributo MQINQ per definizioni di processo		
Selettore	**Lunghezza del campo**	**Descrizione**
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Data della modifica più recente
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Ora della modifica più recente
MQCA_APPL_ID	MQ_PROCESS_APPL_ID_LENGTH	Identificativo applicazione
MQCA_ENV_DATA	MQ_PROCESS_ENV_DATA_LENGTH	Dati ambiente
MQCA_PROCESS_DESC	MQ_PROCESS_DESC_LENGTH	Descrizione della definizione del processo
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	Nome della definizione del processo
MQCA_USER_DATA	MQ_PROCESS_USER_DATA_LENGTH	Dati utente
MQIA_APPL_TYPE	MQLONG	Tipo di applicazione
z/OS MQIA_QSG_DISP	MQLONG	Disposizione del gruppo di condivisione code

Tabella 552. Selettori di attributi MQINQ per il gestore code		
Selettore	**Lunghezza del campo**	**Descrizione**
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Data della modifica più recente
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Ora della modifica più recente
MQCA_CHANNEL_AUTO_DEF_EXIT	MQ_EXIT_NAME_LENGTH	Nome uscita definizione canale automatica
MQCA_CHINIT_SERVICE_PARM		Riservato per l'utilizzo da parte di IBM
MQCA_CLUSTER_WORKLOAD_DATA	MQ_EXIT_DATA_LENGTH	Dati passati all'uscita del workload del cluster
MQCA_CLUSTER_WORKLOAD_EXIT	MQ_EXIT_NAME_LENGTH	Nome dell'uscita del carico di lavoro del cluster
MQCA_COMMAND_INPUT_Q_NAME	MQ_Q_NAME_LENGTH	Nome coda di immissione comandi di sistema
MQCA_CUSTOM	MQ_CUSTOM_LENGTH	L'attributo personalizzato per le nuove funzioni
MQCA_DEAD_LETTER_Q_NAME	MQ_Q_NAME_LENGTH	Nome della coda di messaggi non recapitabili
MQCA_DEF_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	Nome coda di trasmissione predefinita

Tabella 552. Selettori di attributi MQINQ per il gestore code (Continua)		
Selettore	**Lunghezza del campo**	**Descrizione**
z/OS MQCA_DNS_GROUP	MQ_DNS_GROUP_NAME_LENGTH	Il nome del gruppo per il listener TCP che gestisce le trasmissioni in entrata per il gruppo di condivisione code da unire. Il nome viene applicato quando si utilizza Workload Manager Dynamic Domain Name Services.
z/OS MQCA_IGQ_USER_ID	MQ_USER_ID_LENGTH	Identificativo utente accodamento all'interno del gruppo
MQCA_INITIAL_KEY	MQ_INITIAL_KEY_LENGTH	Chiave iniziale per il sistema di protezione password ritorna******** se non vuoto o vuoto se è in uso la chiave iniziale predefinita.
ALW MQCA_INSTALLATION_DESC	MQ_INSTALLATION_DESC_LENGTH	Descrizione dell'installazione associata
ALW MQCA_INSTALLATION_NAME	MQ_INSTALLATION_NAME_LENGTH	Nome dell'installazione associata al gestore code
ALW MQCA_INSTALLATION_PATH	MQ_INSTALLATION_PATH_LENGTH	Percorso in cui è installato il IBM MQ associato
Multi MQCA_LU_GROUP_NAME	MQ_LU_NAME_LENGTH	Nome LU generico per il listener LU 6.2 che gestisce le trasmissioni in entrata per il gruppo di condivisione code da utilizzare
z/OS MQCA_LU_NAME	MQ_LU_NAME_LENGTH	Nome della LU da utilizzare per le trasmissioni della LU in uscita 6.2 . Impostare questo nome sullo stesso LU utilizzato dal listener per le trasmissioni in entrata
z/OS MQCA_LU62_ARM_SUFFIX	MQ_ARM_SUFFIX_LENGTH	Suffisso del SYS1.PARMLIB membro APPCPM *xx*, che nomina LUADD per questo iniziatore di canali
MQCA_PARENT	MQ_Q_MGR_NAME_LENGTH	Nome di un gestore code connesso gerarchicamente nominato come principale di questo gestore code
MQCA_Q_MGR_DESC	MQ_Q_MGR_DESC_LENGTH	La descrizione del gestore code
MQCA_Q_MGR_IDENTIFIER	MQ_Q_MGR_IDENTIFIER_LENGT	Identificativo gestore code (H)
MQCA_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	Nome del gestore code locale
z/OS MQCA_QSG_NAME	MQ_QSG_NAME_LENGTH	Nome gruppo di condivisione code
MQCA_REPOSITORY_NAME	MQ_CLUSTER_NAME_LENGTH	Nome del cluster per cui il gestore code fornisce i servizi del repository

Tabella 552. Selettori di attributi MQINQ per il gestore code (Continua)		
Selettore	**Lunghezza del campo**	**Descrizione**
MQCA_REPOSITORY_NAMELIST	MQ_NAMELIST_NAME_LENGTH	Nome dell'oggetto elenco nomi contenente i nomi dei cluster per cui il gestore code fornisce i servizi del repository
MQCA_SSL_KEY_REPO_PASSWORD	MQ_SSL_ENCRYP_KEY_REPO_PWD_LEN	Password repository chiavi ritorna******** se non vuoto o vuoto se non impostato. Codificato quando impostato prima della memorizzazione.
z/OS MQCA_TCP_NAME	MQ_TCP_NAME_LENGTH	Il nome del sistema TCP/IP che si sta utilizzando
Multi MQIA_ACCOUNTING_CONN_OVERRIDE	MQLONG	Sovrascrivi impostazioni di account
Multi MQIA_ACCOUNTING_INTERVAL	MQLONG	Frequenza di scrittura dei record di contabilità intermedi
Multi MQIA_ACCOUNTING_MQI	MQLONG	Controlla la raccolta di informazioni di account per i dati MQI
Multi MQIA_ACCOUNTING_Q	MQLONG	Controlla la raccolta di informazioni di account per code
z/OS MQIA_ACTIVE_CHANNELS	MQLONG	Numero massimo di canali che possono essere attivi in qualsiasi momento
z/OS MQIA_ADOPTNEWMCA_CHECK	MQLONG	Elementi controllati per determinare se adottare un MCA. Il controllo viene eseguito quando viene rilevato un nuovo canale in entrata con lo stesso nome di un MCA già attivo.
Multi MQIA_ADOPTNEWMCA_INTERVAL	MQLONG	Quantità di tempo, in secondi, per cui il nuovo canale attende la fine del canale orfano
z/OS MQIA_ADOPTNEWMCA_TYPE	MQLONG	Se riavviare automaticamente un'istanza orfana di un MCA di un particolare tipo di canale quando viene rilevata una nuova richiesta di canale in entrata che corrisponde ai parametri AdoptNewMCACheck
Multi MQIA_AUTHORITY_EVENT	MQLONG	Attributo di controllo per gli eventi di autorizzazione
z/OS MQIA_BRIDGE_EVENT	MQLONG	Attributo di controllo per eventi bridge IMS

Tabella 552. Selettori di attributi MQINQ per il gestore code (Continua)		
Selettore	**Lunghezza del campo**	**Descrizione**
Multi MQIA_CHANNEL_AUTO_DEF	MQLONG	Attributo di controllo per la definizione di canale automatica
Multi MQIA_CHANNEL_AUTO_DEF_EVENT	MQLONG	Attributo di controllo per gli eventi di definizione canale automatici
MQIA_CHANNEL_EVENT	MQLONG	Attributo di controllo per gli eventi canale
z/OS MQIA_CHINIT_ADAPTERS	MQLONG	Numero di attività secondarie dell'adattatore da utilizzare per elaborare le chiamate IBM MQ
z/OS MQIA_CHINIT_DISPATCHERS	MQLONG	Numero di dispatcher da utilizzare per l'iniziatore di canali
z/OS MQIA_CHINIT_TRACE_AUTO_START	MQLONG	Indica se avviare automaticamente la traccia dell'iniziatore di canali
z/OS MQIA_CHINIT_TRACE_TABLE_SIZE	MQLONG	Dimensione dello spazio dati di traccia (in MB) dell'iniziatore di canali
MQIA_CLUSTER_WORKLOAD_LENGTH	MQLONG	Lunghezza carico di lavoro cluster.
MQIA_CLWL_MRU_CHANNELS	MQLONG	Numero di canali utilizzati più di recente per il bilanciamento del carico di lavoro del cluster
MQIA_CLWL_USEQ	MQLONG	Utilizza code remote
MQIA_CODED_CHAR_SET_ID	MQLONG	CCSID (Coded character set identifier)
MQIA_COMMAND_EVENT	MQLONG	Attributo di controllo per gli eventi comando
MQIA_COMMAND_LEVEL	MQLONG	Livello di comando supportato dal gestore code
Multi MQIA_CONFIGURATION_EVENT	MQLONG	Attributo di controllo per gli eventi di configurazione
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	MQLONG	Tipo di coda di trasmissione predefinita da utilizzare per i canali mittenti del cluster.
Multi MQIA_DIST_LISTS	MQLONG	Supporto elenco di distribuzione
z/OS MQIA_DNS_WLM	MQLONG	Se il listener TCP che gestisce le trasmissioni in entrata per il gruppo di condivisione code esegue la registrazione con Workload Manager for Dynamic Domain Name Services

Tabella 552. Selettori di attributi MQINQ per il gestore code (Continua)		
Selettore	**Lunghezza del campo**	**Descrizione**
z/OS MQIA_EXPIRY_INTERVAL	MQLONG	Intervallo tra le scansioni per i messaggi scaduti
z/OS MQIA_GROUP_UR	MQLONG	Attributo di controllo per specificare se le unità di ripristino GROUP sono abilitate per questo gestore code. La disposizione dell'unità di ripristino GROUP è disponibile solo se il gestore code è un membro di un gruppo di condivisione code
z/OS MQIA_IGQ_PUT_AUTHORITY	MQLONG	Autorizzazione di inserimento nella coda all'interno del gruppo
Multi MQIA_INHIBIT_EVENT	MQLONG	Attributo di controllo per gli eventi di inibizione
z/OS MQIA_INTRA_GROUP_QUEUING	MQLONG	Supporto di accodamento all'interno del gruppo
z/OS MQIA_LISTENER_TIMER	MQLONG	L'intervallo di tempo (in secondi) tra i tentativi di IBM MQ di riavviare il listener se APPC o TCP/IP hanno esito negativo.
Multi MQIA_LOCAL_EVENT	MQLONG	Attributo di controllo per gli eventi locali
Multi MQIA_LOGGER_EVENT	MQLONG	Attributo di controllo per gli eventi di inibizione
z/OS MQIA_LU62_CHANNELS	MQLONG	Numero massimo di canali che possono essere correnti o client che possono essere connessi, utilizzando il protocollo di trasmissione LU 6.2
MQIA_MSG_MARK_BROWSE_INTERVAL	MQLONG	Intervallo di tempo (in millesimi di secondo) dopo il quale il gestore code può rimuovere automaticamente un contrassegno dai messaggi di ricerca. **Attenzione:** Non impostare questo valore al di sotto del valore predefinito di 5000.
z/OS MQIA_MAX_CHANNELS	MQLONG	Numero massimo di canali correnti (inclusi i canali di connessione server con i client connessi)
MQIA_MAX_HANDLES	MQLONG	Il numero massimo di puntatori
MQIA_MAX_MSG_LENGTH	MQLONG	Lunghezza massima dei messaggi
MQIA_MAX_PRIORITY	MQLONG	Priorità massima
MQIA_MAX_UNCOMMITTED_MSGS	MQLONG	Numero massimo di messaggi di cui non è stato eseguito il commit in un'unità di lavoro

Tabella 552. Selettori di attributi MQINQ per il gestore code (Continua)		
Selettore	**Lunghezza del campo**	**Descrizione**
z/OS MQIA_OUTBOUND_PORT_MAX	MQLONG	Con MQIA_OUTBOUND_PORT_MIN, definisce l'intervallo di numeri porta da utilizzare durante il bind dei canali in uscita
z/OS MQIA_OUTBOUND_PORT_MIN	MQLONG	Con MQIA_OUTBOUND_PORT_MAX, definisce l'intervallo di numeri porta da utilizzare durante il bind dei canali in uscita
Multi MQIA_PERFORMANCE_EVENT	MQLONG	Attributo di controllo per gli eventi delle prestazioni
MQIA_PLATFORM	MQLONG	Piattaforma su cui risiede il gestore code
MQIA_PROT_POLICY_CAPABILITY	MQLONG	Indica se le funzionalità di sicurezza di Advanced Message Security sono disponibili per un gestore code.
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	MQLONG	Il numero di tentativi di rielaborazione di un messaggio di comando non riuscito nel punto di sincronizzazione
MQIA_PUBSUB_MODE	MQLONG	Indica se il motore di pubblicazione / sottoscrizione e l'interfaccia di pubblicazione / sottoscrizione accodata sono in esecuzione. Le applicazioni da pubblicare o sottoscrivere utilizzando l'API (Application Programming Interface) richiedono il motore di pubblicazione / sottoscrizione. Le code monitorate dall'interfaccia di pubblicazione / sottoscrizione in coda richiedono l'esecuzione dell'interfaccia di pubblicazione / sottoscrizione in coda.
MQIA_PUBSUB_NP_MSG	MQLONG	Se eliminare (o conservare) un messaggio di input non consegnato
MQIA_PUBSUB_NP_RESP	MQLONG	Controlla il funzionamento dei messaggi di risposta non distribuiti
MQIA_PUBSUB_SYNC_PT	MQLONG	Se solo i messaggi persistenti (o tutti) vengono elaborati nel punto di sincronizzazione
z/OS MQIA_QMGR_CFCONLOS	MQLONG	Specifica l'azione da intraprendere quando il gestore code perde la connettività alla struttura di amministrazione o a qualsiasi struttura CF con CFCONLOS impostato su ASQMGR

Tabella 552. Selettori di attributi MQINQ per il gestore code (Continua)		
Selettore	**Lunghezza del campo**	**Descrizione**
z/OS MQIA_RECEIVE_TIMEOUT	MQLONG	Circa il tempo di attesa di un canale TCP/IP per ricevere i dati, inclusi gli heartbeat, dal partner, prima di ritornare allo stato inattivo. Il valore è numerico, qualificato da MQIA_RECEIVE_TIMEOUT_TYPE.
z/OS MQIA_RECEIVE_TIMEOUT_MIN	MQLONG	Tempo minimo di attesa di un canale TCP/IP per ricevere i dati, inclusi gli heartbeat, dal relativo partner, prima di tornare allo stato inattivo
z/OS MQIA_RECEIVE_TIMEOUT_TYPE	MQLONG	Circa il tempo di attesa di un canale TCP/IP per ricevere i dati, inclusi gli heartbeat, dal partner, prima di ritornare allo stato inattivo. MQIA_RECEIVE_TIMEOUT_TYPE è il qualificatore applicato a MQIA_RECEIVE_TIMEOUT.
Multi MQIA_REMOTE_EVENT	MQLONG	Attributo Controllo per eventi remoti
z/OS MQIA_SECURITY_CASE	MQLONG	Caso dei profili di sicurezza
MQIA_SSL_EVENT	MQLONG	Attributo di controllo per gli eventi canale
MQIA_SSL_FIPS_REQUIRED	MQLONG	Utilizza solo algoritmi certificati FIPS per la crittografia
MQIA_SSL_RESET_COUNT	MQLONG	Conteggio reimpostazioni chiave TLS
Multi MQIA_START_STOP_EVENT	MQLONG	Attributo di controllo per gli eventi di avvio arresto
MQIA_STATISTICS_AUTO_CLUSSDR	MQLONG	Controlla la raccolta delle informazioni di controllo delle statistiche per i canali mittenti del cluster
MQIA_STATISTICS_CHANNEL	MQLONG	Controlla la raccolta dei dati statistici per i canali
Multi MQIA_STATISTICS_INTERVAL	MQLONG	Frequenza di scrittura dei dati di monitoraggio delle statistiche
Multi MQIA_STATISTICS_MQI	MQLONG	Controlla la raccolta delle informazioni di controllo delle statistiche per il gestore code
Multi MQIA_STATISTICS_Q	MQLONG	Controlla la raccolta dei dati statistici per le code
MQIA_SYNCPOINT	MQLONG	disponibilità punto di sincronizzazione

Tabella 552. Selettori di attributi MQINQ per il gestore code (Continua)		
Selettore	**Lunghezza del campo**	**Descrizione**
z/OS MQIA_TCP_CHANNELS	MQLONG	Numero massimo di canali che possono essere correnti o client che possono essere connessi, utilizzando il protocollo di trasmissione TCP/IP
z/OS MQIA_TCP_KEEP_ALIVE	MQLONG	Se utilizzare la funzione TCP KEEPALIVE per verificare che l'altra estremità della connessione sia ancora disponibile
z/OS MQIA_TCP_STACK_TYPE	MQLONG	Se l'iniziatore del canale può utilizzare solo lo spazio di indirizzo TCP/IP specificato in TCPNAME o può facoltativamente collegarsi a qualsiasi indirizzo TCP/IP selezionato
z/OS MQIA_TRACE_ROUTE_RECORDING	MQLONG	Controlla la registrazione delle informazioni di traccia - instradamento
MQIA_TREE_LIFE_TIME	MQLONG	Durata degli argomenti non amministrativi non utilizzati
MQIA_TRIGGER_INTERVAL	MQLONG	Intervallo trigger

IntAttrCount
Tipo: MQLONG - input

Questo è il numero di elementi nell'array *IntAttrs* . Zero è un valore valido.

Se IntAttrCount è almeno il numero di MQIA_* selettori nel parametro **Selectors** , vengono restituiti tutti gli attributi integer richiesti.

IntAttrs
Tipo: MQLONG x *IntAttrCount* - output

Questo è un array di *IntAttrCount* valori di attributo integer.

I valori degli attributi interi sono restituiti nello stesso ordine dei selettori di MQIA_* nel parametro **Selectors** . Se l'array contiene più elementi rispetto al numero di selettori MQIA_* , gli elementi in eccesso non vengono modificati.

Se *Hobj* rappresenta una coda, ma un selettore di attributi non si applica a tale tipo di coda, viene restituito il valore specifico MQIAV_NOT_APPLICABLE . Viene restituito per l'elemento corrispondente nell'array *IntAttrs* .

Se il parametro **IntAttrCount** o **SelectorCount** è zero, non si fa riferimento a *IntAttrs* .
In questo caso, l'indirizzo del parametro passato dai programmi scritti nell'assembler C o S/390 potrebbe essere null.

CharAttrLunghezza
Tipo: MQLONG - input

È la lunghezza in byte del parametro **CharAttrs** .

CharAttrLength deve essere almeno la somma delle lunghezze degli attributi dei caratteri richiesti (consultare Selettori). Zero è un valore valido.

CharAttrs
Tipo: MQCHAR x *CharAttrLength* - output

Si tratta del buffer in cui vengono restituiti gli attributi carattere, concatenati insieme. La lunghezza del buffer viene fornita dal parametro **CharAttrLength** .

Gli attributi dei caratteri vengono restituiti nello stesso ordine dei selettori MQCA_* nel parametro **Selectors** . La lunghezza di ciascuna stringa attributo è fissa per ciascun attributo (vedere Selettori) e il valore in essa contenuto viene riempito a destra con spazi vuoti, se necessario. È possibile fornire un buffer più grande del necessario per contenere tutti gli attributi dei caratteri richiesti e il riempimento. I byte oltre l'ultimo valore di attributo restituito non vengono modificati.

Se *Hobj* rappresenta una coda, ma un selettore di attributi non si applica a quel tipo di coda, viene restituita una stringa di caratteri composta interamente da asterischi (*). L'asterisco viene restituito come valore di tale attributo in *CharAttrs*.

Se il parametro *CharAttrLength* o **SelectorCount** è zero, non si fa riferimento a *CharAttrs* . In questo caso, l'indirizzo del parametro passato dai programmi scritti nell'assembler C o S/390 potrebbe essere null.

CompCode
Tipo: MQLONG - output

Il codice di completamento:

MQCC_OK
Completamento con esito positivo.

MQCC_WARNING
Avvertenza (completamento parziale).

MQCC_FAILED
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000') Nessun motivo per la notifica.

Se *CompCode* è MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT
(2008, X'7D8') Spazio non sufficiente consentito per gli attributi carattere.

MQRC_INT_ATTR_COUNT_TOO_SMALL
(2022, X'7E6') Spazio non sufficiente consentito per gli attributi integer.

MQRC_SELECTOR_NOT_FOR_TYPE
(2068, X'814') Selettore non applicabile al tipo di coda.

Se *CompCode* è MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') L'adattatore non è disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Impossibile caricare il modulo di servizio dell'adattatore.

MQRC_API_EXIT_ERROR
(2374, X'946') uscita API non riuscita.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') Impossibile caricare l'uscita API.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La chiamata MQI immessa prima del completamento della chiamata precedente.

MQRC_CF_STRUC_FAILED
(2373, X'945') La struttura della funzione di accoppiamento non è riuscita.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') Struttura della funzione di accoppiamento in uso.

MQRC_CHAR_ATTR_LENGTH_ERROR
(2006, X'7D6') Lunghezza degli attributi carattere non valida.

MQRC_CHAR_ATTRS_ERROR
(2007, X'7D7') Stringa di attributi carattere non valida.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Richiesta di attesa rifiutata da CICS.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Non autorizzato per la connessione.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Chiusura della connessione.

MQRC_HCONN_ERROR
(2018, X'7E2') Handle di collegamento non valido.

MQRC_HOBJ_ERROR
(2019, X'7E3') Gestione oggetto non valida.

MQRC_INT_ATTR_COUNT_ERROR
(2021, X'7E5') Conteggio degli attributi interi non valido.

MQRC_INT_ATTRS_ARRAY_ERROR
(2023, X'7E7') Array di attributi interi non valido.

MQRC_NOT_OPEN_FOR_INQUIRE
(2038, X'7F6') La coda non è aperta per l'interrogazione.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Definizione oggetto modificata dall'apertura.

MQRC_OBJECT_DAMAGED
(2101, X'835') Oggetto danneggiato.

MQRC_PAGESET_ERROR
(2193, X'891') Errore di accesso al dataset della serie di pagine.

MQRC_Q_DELETED
(2052, X'804') Coda eliminata.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Il nome del gestore code non è valido o non è noto.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Gestore code non disponibile per la connessione.

MQRC_Q_MGR_STOPPING
(2162, X'872') Chiusura del gestore code.

MQRC_RESOURCE_PROBLEM
(2102, X'836') Risorse di sistema insufficienti.

MQRC_SELECTOR_COUNT_ERROR
(2065, X'811') Conteggio dei selettori non valido.

MQRC_SELECTOR_ERROR
(2067, X'813') Selettore attributo non valido.

MQRC_SELECTOR_LIMIT_EXCEEDED
(2066, X'812') Conteggio dei selettori troppo grande.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Memoria disponibile insufficiente.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Chiamata eliminata dal programma di uscita.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici motivo

Note d'utilizzo

1. I valori restituiti sono un'istantanea degli attributi selezionati. Non vi è alcuna garanzia che gli attributi rimangano gli stessi prima che l'applicazione possa agire sui valori restituiti.
2. Quando si apre una coda modello, viene creata una coda locale dinamica. Una coda locale dinamica viene creata anche se si apre la coda modello per informarsi sui suoi attributi.

   Gli attributi della coda dinamica sono in gran parte gli stessi degli attributi della coda modello al momento della creazione della coda dinamica. Se, quindi, si utilizza la chiamata MQINQ su questa coda, il gestore code restituisce gli attributi della coda dinamica e non gli attributi della coda modello. Consultare Tabella 561 a pagina 860 per i dettagli su quali attributi della coda modello vengono ereditati dalla coda dinamica.
3. Se l'oggetto che viene interrogato è una coda alias, i valori di attributo restituiti dalla chiamata MQINQ sono gli attributi della coda alias. Non sono gli attributi della coda di base o dell'argomento in cui si risolve l'alias.
4. Se l'oggetto da interrogare è una coda cluster, gli attributi che possono essere interrogati dipendono dalla modalità di apertura della coda:
   - È possibile aprire una coda cluster per l'interrogazione più una o più operazioni di input, di ricerca o di impostazione. Per fare ciò, è necessario che ci sia un'istanza locale della coda del cluster perché l'apertura abbia esito positivo. In questo caso, gli attributi che possono essere interrogati sono quelli validi per le code locali.

     Se la coda del cluster è aperta per l'interrogazione senza input, ricerca o impostazione specificata, la chiamata restituisce il codice di completamento MQCC_WARNING e il codice motivo MQRC_SELECTOR_NOT_FOR_TYPE (2068) se si tenta di interrogare gli attributi validi solo per le code locali e non per le code cluster.
   - È possibile aprire una coda cluster per l'interrogazione durante la trasmissione del nome del gestore code di base del gestore code connesso.

     Per fare ciò, è necessario che ci sia un'istanza locale della coda del cluster perché l'apertura abbia esito positivo. Se il gestore code di base non viene passato, la chiamata restituisce il codice di completamento MQCC_WARNING e il codice motivo MQRC_SELECTOR_NOT_FOR_TYPE (2068) se si tenta di interrogare gli attributi validi solo per le code locali e non le code cluster
   - Se la coda del cluster è aperta solo per l'interrogazione o per l'interrogazione e l'emissione, è possibile interrogare solo gli attributi elencati. L'attributo **QType** ha il valore MQQT_CLUSTER in questo caso:
     - MQCA_Q_DESC
     - MQCA_Q_NAME
     - MQIA_DEF_BIND
     - MQIA_DEF_PERSISTENCE
     - MQIA_DEF_PRIORITY
     - MQIA_INHIBIT_PUT
     - MQIA_Q_TYPE

     È possibile aprire la coda cluster senza alcun collegamento fisso. È possibile aprirlo con MQOO_BIND_NOT_FIXED specificato nella chiamata MQOPEN . In alternativa, specificare MQOO_BIND_AS_Q_DEFe impostare l'attributo **DefBind** della coda su MQBND_BIND_NOT_FIXED. Se si apre una coda cluster senza alcun collegamento fisso, le successive chiamate MQINQ per la coda potrebbero richiedere istanze differenti della coda cluster. Tuttavia, è tipico per tutte le istanze che hanno gli stessi valori di attributo.

- È possibile definire un oggetto coda alias per un cluster. Poiché TARGTYPE e TARGET non sono attributi cluster, il processo che esegue un processo MQOPEN sulla coda alias non è a conoscenza dell'oggetto su cui si risolve l'alias.

  Durante il MQOPENiniziale, la coda alias si risolve in un gestore code e in una coda nel cluster. La risoluzione dei nomi si verifica nuovamente sul gestore code remoto ed è qui che viene risolta la TARGTPYE della coda alias.

  Se la coda alias si risolve in un alias dell'argomento, la pubblicazione dei messaggi inseriti nella coda alias avviene in questo gestore code remoto.

Consultare Code cluster

5. È possibile che si desideri interrogare un numero di attributi e quindi impostarne alcuni utilizzando la chiamata MQSET . Per programmare inquire e impostare in modo efficiente, posizionare gli attributi da impostare all'inizio degli array del selettore. In questo caso, gli stessi array con conteggi ridotti possono essere utilizzati per MQSET.
6. Se si verifica più di una delle situazioni di avvertenza (consultare il parametro **CompCode**), il codice di errore restituito è il primo nel seguente elenco che si applica:
   a. MQRC_SELECTOR_NOT_FOR_TYPE
   b. MQRC_INT_ATTR_COUNT_TOO_SMALL
   c. MQRC_CHAR_ATTRS_TOO_SHORT
7. Il seguente argomento contiene informazioni sugli attributi dell'oggetto:
   - “Attributi per le code” a pagina 858
   - “Attributi per gli elenchi nomi” a pagina 892
   - “Attributi per le definizioni di processi” a pagina 895
   - “Attributi per il gestore code” a pagina 818

Richiamo C

```
MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;           /* Connection handle */
MQHOBJ   Hobj;            /* Object handle */
MQLONG   SelectorCount;   /* Count of selectors */
MQLONG   Selectors[n];    /* Array of attribute selectors */
MQLONG   IntAttrCount;    /* Count of integer attributes */
MQLONG   IntAttrs[n];     /* Array of integer attributes */
MQLONG   CharAttrLength;  /* Length of character attributes buffer */
MQCHAR   CharAttrs[n];    /* Character attributes */
MQLONG   CompCode;        /* Completion code */
MQLONG   Reason;          /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
     CALL 'MQINQ' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,
                        INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,
                        CHARATTRS, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN            PIC S9(9) BINARY.
**   Object handle
 01  HOBJ             PIC S9(9) BINARY.
```

```
**   Count of selectors
 01  SELECTORCOUNT    PIC S9(9) BINARY.
**   Array of attribute selectors
 01  SELECTORS-TABLE.
  02  SELECTORS        PIC S9(9) BINARY OCCURS n TIMES.
**   Count of integer attributes
 01  INTATTRCOUNT     PIC S9(9) BINARY.
**   Array of integer attributes
 01  INTATTRS-TABLE.
  02  INTATTRS         PIC S9(9) BINARY OCCURS n TIMES.
**   Length of character attributes buffer
 01  CHARATTRLENGTH   PIC S9(9) BINARY.
**   Character attributes
 01  CHARATTRS        PIC X(n).
**   Completion code
 01  COMPCODE         PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON           PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
           IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn           fixed bin(31);  /* Connection handle */
dcl Hobj            fixed bin(31);  /* Object handle */
dcl SelectorCount   fixed bin(31);  /* Count of selectors */
dcl Selectors(n)    fixed bin(31);  /* Array of attribute selectors */
dcl IntAttrCount    fixed bin(31);  /* Count of integer attributes */
dcl IntAttrs(n)     fixed bin(31);  /* Array of integer attributes */
dcl CharAttrLength  fixed bin(31);  /* Length of character attributes
                                       buffer */
dcl CharAttrs       char(n);        /* Character attributes */
dcl CompCode        fixed bin(31);  /* Completion code */
dcl Reason          fixed bin(31);  /* Reason code qualifying
                                       CompCode */
```

Chiamata High Level Assembler

```
         CALL MQINQ,(HCONN,HOBJ,SELECTORCOUNT,SELECTORS,INTATTRCOUNT,  X
               INTATTRS,CHARATTRLENGTH,CHARATTRS,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN           DS  F      Connection handle
HOBJ            DS  F      Object handle
SELECTORCOUNT   DS  F      Count of selectors
SELECTORS       DS  (n)F   Array of attribute selectors
INTATTRCOUNT    DS  F      Count of integer attributes
INTATTRS        DS  (n)F   Array of integer attributes
CHARATTRLENGTH  DS  F      Length of character attributes buffer
CHARATTRS       DS  CL(n)  Character attributes
COMPCODE        DS  F      Completion code
REASON          DS  F      Reason code qualifying COMPCODE
```

Richiamo Visual Basic

```
MQINQ Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, CompCode, Reason
```

Dichiarare i parametri come segue:

```
Dim Hconn          As Long   'Connection handle'
Dim Hobj           As Long   'Object handle'
Dim SelectorCount  As Long   'Count of selectors'
Dim Selectors      As Long   'Array of attribute selectors'
Dim IntAttrCount   As Long   'Count of integer attributes'
Dim IntAttrs       As Long   'Array of integer attributes'
Dim CharAttrLength As Long   'Length of character attributes buffer'
Dim CharAttrs      As String 'Character attributes'
Dim CompCode       As Long   'Completion code'
Dim Reason         As Long   'Reason code qualifying CompCode'
```

MQINQMP - Proprietà del messaggio di interrogazione

La chiamata a MQINQMP restituisce il valore di una proprietà di un messaggio.

Sintassi

MQINQMP (*Hconn*, *Hmsg*, *InqPropOpts*, *Name*, *PropDesc*, *Type*, *ValueLength*, *Value*, *DataLength*, *CompCode*, *Reason*)

Parametri

Hconn

Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* deve corrispondere all'handle di connessione utilizzato per creare l'handle del messaggio specificato nel parametro **Hmsg** .

Se l'handle del messaggio è stato creato utilizzando MQHC_UNASSOCIATED_HCONN, è necessario stabilire una connessione valida sul thread che richiede una proprietà dell'handle del messaggio, altrimenti la chiamata non riesce con MQRC_CONNECTION_BROKEN.

Msg

Tipo: MQHMSG - input

Questo è l'handle del messaggio da interrogare. Il valore è stato restituito da una precedente chiamata **MQCRTMH** .

Opzioni InqProp

Tipo: MQIMPO - input/output

Consultare il tipo di dati MQIMPO per i dettagli.

Nome

Tipo: MQCHARV - input/output

Il nome della proprietà da analizzare.

Se non è possibile trovare alcuna proprietà con questo nome, la chiamata non riesce con motivo MQRC_PROPERTY_NOT_AVAILABLE.

È possibile utilizzare il carattere jolly percentuale (%) alla fine del nome della proprietà. Il carattere jolly corrisponde a zero o più caratteri, incluso il carattere punto (.). Ciò consente a un'applicazione di analizzare il valore di molte proprietà. Richiamare MQINQMP con l'opzione MQIMPO_INQ_FIRST per ottenere la prima proprietà corrispondente e di nuovo con l'opzione MQIMPO_INQ_NEXT per ottenere la proprietà corrispondente successiva. Quando non sono più disponibili proprietà corrispondenti, la chiamata non riesce con MQRC_PROPERTY_NOT_AVAILABLE. Se il campo *ReturnedName* della struttura InqPropOpts viene inizializzato con un indirizzo o un offset per il nome restituito della proprietà, questo viene completato al ritorno da MQINQMP con il nome della proprietà corrispondente. Se il campo *VSBufSize* di *ReturnedName* nella struttura InqPropOpts è inferiore alla lunghezza del nome della proprietà restituita, il codice di completamento viene impostato MQCC_FAILED con motivo MQRC_PROPERTY_NAME_TOO_BIG.

Le proprietà che hanno sinonimi noti vengono restituite come segue:

1. Proprietà con il prefisso "mqps." vengono restituiti come nome proprietà IBM MQ . Ad esempio, "MQTopicString" è il nome restituito piuttosto che "mqps.Top"
2. Proprietà con il prefisso "jms." o "mcd." sono restituiti come il nome del campo di intestazione JMS , ad esempio, "JMSExpiration" è il nome restituito invece di "jms.Exp".
3. Proprietà con il prefisso "usr." vengono restituiti senza tale prefisso, ad esempio, viene restituito "Colore" invece di "usr.Color".

Le proprietà con sinonimi vengono restituite una volta sola.

Nel linguaggio di programmazione C, le seguenti variabili macro sono definite per analizzare tutte le proprietà e quindi tutte le proprietà che iniziano con "usr.":

MQPROP_INQUIRE_TUTTI
Analizzare tutte le proprietà del messaggio.

MQPROP_INQUIRE_ALL può essere utilizzato nel seguente modo:

```
MQCHARV Name = {MQPROP_INQUIRE_ALL};
```

MQPROP_INQUIRE_ALL_USR
Interrogare tutte le proprietà del messaggio che avviano "usr.". Il nome restituito viene restituito senza "usr." .

Se MQIMP_INQ_NEXT è specificato ma il nome è stato modificato rispetto alla chiamata precedente o questa è la prima chiamata, MQIMPO_INQ_FIRST è implicito.

Consultare Nomi proprietà e Limitazioni nome proprietà per ulteriori informazioni sull'utilizzo dei nomi proprietà.

PropDesc
Tipo: MQPD - output

Questa struttura viene utilizzata per definire gli attributi di una proprietà, inclusi gli eventi che si verificano se la proprietà non è supportata, il contesto del messaggio a cui appartiene la proprietà e i messaggi in cui la proprietà deve essere copiata. Consultare MQPD per dettagli su questa struttura.

Tipo
Tipo: MQLONG - input/output

Al ritorno dalla chiamata MQINQMP, questo parametro è impostato sul tipo di dati *Valore*. Il tipo di dati può essere uno dei seguenti:

BOOLEAN MQTIPO
Un valore booleano.

MQTYPE_BYTE_STRING
una stringa di byte.

MQTYPE_INT8
Un numero intero con segno a 8 bit.

MQTYPE_INT16
Un numero intero con segno a 16 bit.

MQTYPE_INT32
Un numero intero con segno a 32 bit.

MQTYPE_INT64
Un numero intero con segno a 64 bit.

MQTYPE_FLOAT32
Un numero a virgola mobile a 32 bit.

MQTYPE_FLOAT64
Un numero a virgola mobile a 64 bit.

MQTYPE_STRING
Una stringa di caratteri.

MQTYPE_NULL
La proprietà esiste ma ha un valore null.

Se il tipo di dati del valore della proprietà non viene riconosciuto, viene restituito MQTYPE_STRING e una rappresentazione stringa del valore viene inserita nell'area *Valore* . È possibile trovare una rappresentazione stringa del tipo di dati nel campo *TypeString* del parametro *InqPropInqProp* . Viene restituito un codice di completamento di avvertenza con motivo MQRC_PROP_TYPE_NOT_SUPPORTED.

Inoltre, se viene specificata l'opzione MQIMPO_CONVERT_TYPE, è richiesta la conversione del valore della proprietà. Utilizzare *Tipo* come input per specificare il tipo di dati per cui si desidera restituire la proprietà. Fare riferimento alla descrizione dell'opzione MQIMPO_CONVERT_TYPE della struttura MQIMPO per i dettagli sulla conversione del tipo di dati.

Se non si richiede la conversione del tipo, è possibile utilizzare il seguente valore nell'input:

MQTYPE_AS_SET
Il valore della proprietà viene restituito senza convertirne il tipo di dati.

ValueLength
Tipo: MQLONG - input

La lunghezza in byte dell'area Valore. Specificare zero per le proprietà per cui non è richiesto il valore restituito. Queste potrebbero essere proprietà progettate da un'applicazione per avere un valore null o una stringa vuota. Specificare anche zero se è stata specificata l'opzione MQIMPO_QUERY_LENGTH ; in questo caso non viene restituito alcun valore.

Valore
Tipo: MQBYTEx *ValueLength* - output

Questa è l'area che deve contenere il valore della proprietà richiesta. Il buffer deve essere allineato su un limite appropriato per il valore restituito. In caso contrario, potrebbe verificarsi un errore quando si accede al valore in un secondo momento.

Se *ValueLength* è inferiore alla lunghezza del valore della proprietà, la maggior parte del valore della proprietà viene spostata in *Valore* e la chiamata ha esito negativo con codice di completamento MQCC_FAILED e motivo MQRC_PROPERTY_VALUE_TOO_BIG.

La serie di caratteri dei dati in *Valore* viene fornita dal campo ReturnedCCSID nel parametro InqPropOpts. La codifica dei dati in *Valore* viene fornita dal campo ReturnedEncoding nel parametro Opts InqProp.

Nel linguaggio di programmazione C, il parametro viene dichiarato come un puntatore a void; l'indirizzo di qualsiasi tipo di dati può essere specificato come parametro.

Se il parametro *ValueLength* è zero, non si fa riferimento a *Valore* e il suo valore passato dai programmi scritti in C o nell'assembler System/390 può essere null.

DataLength
Tipo: MQLONG - output

Questa è la lunghezza in byte del valore della proprietà effettiva come restituito nell'area *Valore* .

Se *DataLength* è inferiore alla lunghezza del valore della proprietà, *DataLength* viene ancora compilato al ritorno dalla chiamata MQINQMP. Ciò consente all'applicazione di determinare la dimensione del buffer richiesta per contenere il valore della proprietà e quindi emettere nuovamente la chiamata con un buffer della dimensione appropriata.

Possono essere restituiti anche i seguenti valori.

Se il parametro *Type* è impostato su MQTYPE_STRING o MQTYPE_BYTE_STRING:

MQVL_EMPTY_STRING
La proprietà esiste ma non contiene caratteri o byte.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

MQRC_PROP_NAME_NON_CONVERTITO
(2492, X'09BC') Nome proprietà restituito non convertito.

PROP_MQRC_XX_ENCODE_CASE_ONE valore_non_CONVERTED
(2466, X'09A2') Valore proprietà non convertito.

MQRC_PROP_TYPE_NON_SUPPORTATO
(2467, X'09A3') Il tipo di dati della proprietà non è supportato.

ERRORE MQRC_RFH_FORMATO
(2421, X'0975 ') Impossibile analizzare una cartella MQRFH2 contenente le proprietà.

Se *CompCode* è MQCC_FAILED:

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'089C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'0852 ') Impossibile caricare il modulo di servizio adattatore.

MQRC_ASID_MISMATCH
(2157, X'086D') Gli ASID principale e principale differiscono.

ERRORE MQRC_BUFFER_
(2004, X'07D4') Parametro valore non valido.

ERRORE MQRC_BUFFER_LENGTH
(2005, X'07D5') Parametro di lunghezza valore non valido.

MQRC_CALL_IN_PROVERDE
(2219, X'08AB') Chiamata MQI immessa prima del completamento della chiamata precedente.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Connessione al gestore code persa.

ERRORE MQRC_DATA_LENGTH
(2010, X'07DA') Parametro di lunghezza dati non valido.

ERRORE MQRC_IMPO
(2464, X'09A0') Struttura delle opzioni della proprietà del messaggio di interrogazione non valida.

ERRORE MQRC_HMSG_
(2460, X'099C') Gestione messaggio non valida.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') handle del messaggio già in uso.

ERRORE MQRC_OPTIONS_
(2046, X'07F8') Opzioni non valide o non congruenti.

ERRORE MQRC_PD
(2482, X'09B2') Struttura descrittore proprietà non valido.

MQRC_PROP_CONV_NON supportato
(2470, X'09A6') Conversione dal tipo di dati effettivo a quello richiesto non supportata.

ERRORE MQRC_PROPERTY_NAME_
(2442, X'098A') Nome proprietà non valido.

MQRC_PROPERTY_NAME_TOO_BIG
(2465, X'09A1') Nome proprietà troppo grande per il buffer dei nomi restituito.

PROPRIETÀ_MQRC_NON_DISPONIBILE
(2471, X'09A7') Proprietà non disponibile.

MQRC_PROPERTY_VALUE_TOO_BIG
(2469, X'09A5') Valore della proprietà troppo grande per l'area Valore.

MQRC_PROP_NUMBER_FORMAT_ERRORE
(2472, X'09A8') Errore di formato numero rilevato nei dati del valore.

ERRORE TIPO_PROFILO_XX_ENCODE_CASE_CAPS_LOCK_OFF MQRC_
(2473, X'09A9') Tipo di proprietà richiesto non valido.

ERRORE CCSID DI MQRC_SOURCE_
(2111, X'083F') Identificativo serie di caratteri codificato del nome proprietà non valido.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'0871 ') Memoria disponibile insufficiente.

ERRORE MQRC_UNEXPECTED_
(2195, X'0893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Richiamo C

```
MQINQMP (Hconn, Hmsg, &InqPropOpts, &Name, &PropDesc, &Type,
ValueLength, Value, &DataLength, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN Hconn;       /* Connection handle */
MQHMSG  Hmsg;        /* Message handle */
MQIMPO  InqPropOpts; /* Options that control the action of MQINQMP */
MQCHARV Name;        /* Property name */
MQPD    PropDesc;    /* Property descriptor */
MQLONG  Type;        /* Property data type */
MQLONG  ValueLength; /* Length in bytes of the Value area */
MQBYTE  Value[n];    /* Area to contain the property value */
MQLONG  DataLength;  /* Length of the property value */
MQLONG  CompCode;    /* Completion code */
MQLONG  Reason;      /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQINQMP' USING HCONN, HMSG, INQMSGOPTS, NAME, PROPDESC, TYPE,
VALUELENGTH, VALUE, DATALENGTH, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN       PIC S9(9) BINARY.
**   Message handle
 01  HMSG        PIC S9(18) BINARY.
**   Options that control the action of MQINQMP
 01  INQMSGOPTS.
```

```
     COPY CMQIMPOV.
**   Property name
 01  NAME.
     COPY CMQCHRVV.
**   Property descriptor
 01  PROPDESC.
     COPY CMQPDV.
**   Property data type
 01  TYPE        PIC S9(9) BINARY.
**   Length in bytes of the VALUE area
 01  VALUELENGTH PIC S9(9) BINARY.
**   Area to contain the property value
 01  VALUE       PIC X(n).
**   Length of the property value
 01  DATALENGTH  PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE    PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON      PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQINQMP (Hconn, Hmsg, InqPropOpts, Name, PropDesc, Type,
ValueLength, Value, DataLength, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn       fixed bin(31);  /* Connection handle */
dcl Hmsg        fixed bin(63);  /* Message handle */
dcl InqPropOpts like MQIMPO;    /* Options that control the action of MQINQMP */
dcl Name        like MQCHARV;   /* Property name */
dcl PropDesc    like MQPD;      /* Property descriptor */
dcl Type        fixed bin (31); /* Property data type */
dcl ValueLength fixed bin (31); /* Length in bytes of the Value area */
dcl Value       char (n);       /* Area to contain the property value */
dcl DataLength  fixed bin (31); /* Length of the property value */
dcl CompCode    fixed bin (31); /* Completion code */
dcl Reason      fixed bin (31); /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQINQMP,(HCONN,HMSG,INQMSGOPTS,NAME,PROPDESC,TYPE,
VALUELENGTH,VALUE,DATALENGTH,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN        DS         F     Connection handle
HMSG         DS         D     Message handle
INQMSGOPTS   CMQIMPOA   ,     Options that control the action of MQINQMP
NAME         CMQCHRVA   ,     Property name
PROPDESC     CMQPDA     ,     Property descriptor
TYPE         DS         F     Property data type
VALUELENGTH  DS         F     Length in bytes of the VALUE area
VALUE        DS         CL(n) Area to contain the property value
DATALENGTH   DS         F     Length of the property value
COMPCODE     DS         F     Completion code
REASON       DS         F     Reason code qualifying COMPCODE
```

MQMHBUF - Conversione dell'handle del messaggio in buffer

La chiamata MQMHBUF converte un gestore messaggi in un buffer ed è l'inverso della chiamata MQBUFMH.

Sintassi

MQMHBUF (*Hconn*, *Hmsg*, *MsgHBufOpts*, *Name*, *MsgDesc*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* deve corrispondere all'handle di connessione utilizzato per creare l'handle del messaggio specificato nel parametro **Hmsg** .

Se l'handle del messaggio è stato creato utilizzando MQHC_UNASSOCIATED_HCONN, è necessario stabilire una connessione valida sul thread che elimina l'handle del messaggio. Se non viene stabilita una connessione valida, la chiamata ha esito negativo con MQRC_CONNECTION_BROKEN.

Msg
Tipo: MQHMSG - input

Questo è l'handle del messaggio per cui è richiesto un buffer. Il valore è stato restituito da una precedente chiamata MQCRTMH.

Opzioni MsgHBuf
Tipo: MQMHBO - input

La struttura MQMHBO consente alle applicazioni di specificare le opzioni che controllano la modalità di produzione dei buffer dagli handle dei messaggi.

Vedi "MQMHBO - Gestore messaggi per opzioni buffer" a pagina 490 per i dettagli.

Nome
Tipo: MQCHARV - input

Il nome della proprietà o delle proprietà da inserire nel buffer.

Se non è possibile trovare alcuna proprietà corrispondente al nome, la chiamata non riesce con MQRC_PROPERTY_NOT_AVAILABLE.

È possibile utilizzare un carattere jolly per inserire più di una proprietà nel buffer. A tale scopo, utilizzare il carattere jolly '%' alla fine del nome della proprietà. Questo carattere jolly corrisponde a zero o più caratteri, incluso il carattere '.' carattere.

Nel linguaggio di programmazione C, le seguenti variabili macro sono definite per analizzare tutte le proprietà e tutte le proprietà che iniziano con 'usr':

MQPROP_INQUIRE_TUTTI
Inserire tutte le proprietà del messaggio nel buffer

MQPROP_INQUIRE_ALL_USR
Inserire tutte le proprietà del messaggio che iniziano con i caratteri 'usr.' nel buffer.

Consultare Nomi proprietà e Limitazioni nome proprietà per ulteriori informazioni sull'utilizzo dei nomi proprietà.

MsgDesc
Tipo: MQMD - input/output

La struttura *MsgDesc* descrive il contenuto dell'area buffer.

In fase di output, i campi *Encoding*, *CodedCharSetId* e *Format* sono impostati per descrivere correttamente la codifica, l'identificativo della serie di caratteri e il formato dei dati nell'area di buffer come scritto dalla chiamata.

I dati in questa struttura sono nella serie di caratteri e nella codifica dell'applicazione.

BufferLength
Tipo: MQLONG - input

BufferLength è la lunghezza dell'area Buffer, in byte.

Memorizza nel buffer
Tipo: MQBYTExBufferLength - output

Buffer definisce l'area che deve contenere le proprietà del messaggio. È necessario allineare il buffer su un limite di 4 byte.

Se *BufferLength* è inferiore alla lunghezza richiesta per memorizzare le proprietà in *Buffer*, MQMHBUF non riesce con MQRC_PROPERTY_VALUE_TOO_BIG.

Il contenuto del buffer può cambiare anche se la chiamata ha esito negativo.

DataLength
Tipo: MQLONG - output

DataLength è la lunghezza, in byte, delle proprietà restituite nel buffer. Se il valore è zero, nessuna proprietà corrisponde al valore fornito in *Name* e la chiamata ha esito negativo con codice motivo MQRC_PROPERTY_NOT_AVAILABLE.

Se *BufferLength* è inferiore alla lunghezza richiesta per memorizzare le proprietà nel buffer, la chiamata MQMHBUF ha esito negativo con MQRC_PROPERTY_VALUE_TOO_BIG, ma viene ancora immesso un valore in *DataLength*. Ciò consente all'applicazione di determinare la dimensione del buffer richiesto per contenere le proprietà e quindi emettere nuovamente la chiamata con il *BufferLength*richiesto.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Il codice di errore che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'089C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

ERRORE MQRC_MHBO_
(2501, X'095C') L'handle del messaggio nella struttura delle opzioni del buffer non è valida.

ERRORE MQRC_BUFFER_
Parametro buffer (2004, X'07D4') non valido.

ERRORE MQRC_BUFFER_LENGTH
Parametro di lunghezza buffer (2005, X'07D5') non valido.

MQRC_CALL_IN_PROVERDE
(2219, X'08AB') Chiamata MQI immessa prima del completamento della chiamata precedente.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Connessione al gestore code persa.

ERRORE MQRC_DATA_LENGTH
(2010, X'07DA') Parametro di lunghezza dati non valido.

ERRORE MQRC_HMSG_
(2460, X'099C') Gestione messaggio non valida.

ERRORE MQRC_MD
(2026, X'07EA') Descrittore messaggio non valido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') handle del messaggio già in uso.

ERRORE MQRC_OPTIONS_
(2046, X'07FE') Opzioni non valide o non congruenti.

ERRORE MQRC_PROPERTY_NAME_
(2442, X'098A') Il nome proprietà non è valido.

PROPRIETÀ_MQRC_NON_DISPONIBILE
(2471, X'09A7') Proprietà non disponibile.

MQRC_PROPERTY_VALUE_TOO_BIG
(2469, X'09A5') Il valore BufferLength è troppo piccolo per contenere le proprietà specificate.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Richiamo C

```
MQMHBUF (Hconn, Hmsg, &MsgHBufOpts, &Name, &MsgDesc, BufferLength, Buffer,
         &DataLength, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN Hconn;        /* Connection handle */
MQHMSG  Hmsg;         /* Message handle */
MQMHBO  MsgHBufOpts;  /* Options that control the action of MQMHBUF */
MQCHARV Name;         /* Property name */
MQMD    MsgDesc;      /* Message descriptor */
MQLONG  BufferLength; /* Length in bytes of the Buffer area */
MQBYTE  Buffer[n];    /* Area to contain the properties */
MQLONG  DataLength;   /* Length of the properties */
MQLONG  CompCode;     /* Completion code */
MQLONG  Reason;       /* Reason code qualifying CompCode */
```

Note d'utilizzo

MQMHBUF converte un handle del messaggio in un buffer.

È possibile utilizzarla con un'uscita API MQGET per accedere a determinate proprietà, utilizzando le API delle proprietà del messaggio, e quindi inoltrarle di nuovo in un buffer a un'applicazione progettata per utilizzare le intestazioni MQRFH2 piuttosto che gli handle del messaggio.

Questa chiamata è l'inverso della chiamata MQBUFMH, che è possibile utilizzare per analizzare le proprietà del messaggio da un buffer in un handle del messaggio.

Richiamo COBOL

```
     CALL 'MQMHBUF' USING HCONN, HMSG, MSGHBUFOPTS, NAME, MSGDESC,
                          BUFFERLENGTH, BUFFER, DATALENGTH, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Message handle
 01  HMSG          PIC S9(18) BINARY.
**   Options that control the action of MQMHBUF
 01  MSGHBUFOPTS.
     COPY CMQMHBOV.
**   Property name
 01  NAME
     COPY CMQCHRVV.
**   Message descriptor
 01  MSGDESC
     COPY CMQMDV.
**   Length in bytes of the Buffer area */
 01  BUFFERLENGTH  PIC S9(9) BINARY.
**   Area to contain the properties
 01  BUFFER        PIC X(n).
**   Length of the properties
 01  DATALENGTH    PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE      PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQMHBUF (Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer,
DataLength, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn        fixed bin(31); /* Connection handle */
dcl Hmsg         fixed bin(63); /* Message handle */
dcl MsgHBufOpts  like MQMHBO;   /* Options that control the action of MQMHBUF */
dcl Name         like MQCHARV;  /* Property name */
dcl MsgDesc      like MQMD;     /* Message descriptor */
dcl BufferLength fixed bin(31); /* Length in bytes of the Buffer area */
dcl Buffer       char(n);       /* Area to contain the properties */
dcl DataLength   fixed bin(31); /* Length of the properties */
dcl CompCode     fixed bin(31); /* Completion code */
dcl Reason       fixed bin(31); /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQMHBUF,(HCONN,HMSG,MSGHBUFOPTS,NAME,MSGDESC,BUFFERLENGTH,
               BUFFER,DATALENGTH,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN        DS       F      Connection handle
HMSG         DS       D      Message handle
MSGHBUFOPTS  CMQMHBOA ,      Options that control the action of MQMHBUF
NAME         CMQCHRVA ,      Property name
MSGDESC      CMQMDA   ,      Message descriptor
BUFFERLENGTH DS       F      Length in bytes of the BUFFER area
BUFFER       DS       CL(n)  Area to contain the properties
DATALENGTH   DS       F      Length of the properties
```

```
COMPCODE     DS       F      Completion code
REASON       DS       F      Reason code qualifying COMPCODE
```

MQOPEN - Apri oggetto

La chiamata MQOPEN stabilisce l'accesso a un oggetto.

Sono validi i seguenti tipi di oggetto:

- Coda (inclusi elenchi di distribuzione)
- Elenco nomi
- Definizione di processo
- Gestore code
- Argomento

Sintassi

MQOPEN (*Hconn*, *ObjDesc*, *Opzioni*, *Hobj*, *CompCode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di Hconn è stato restituito da una chiamata MQCONN o MQCONNX precedente.

z/OS In applicazioni z/OS per CICS , la chiamata MQCONN può essere omessa e il seguente valore specificato per *Hconn*:

DEF_MQH_HCONN
Handle di connessione predefinito.

ObjDesc
Tipo: MQOD - input/output

Si tratta di una struttura che identifica l'oggetto da aprire; consultare "MQOD - Descrittore oggetto" a pagina 492 per i dettagli.

Se il campo ObjectName nel parametro **ObjDesc** è il nome di una coda modello, una coda locale dinamica viene creato con gli attributi della coda modello; ciò si verifica indipendentemente dalle opzioni specificate nel parametro **Options** . Le successive operazioni che utilizzano il Hobj restituito dalla chiamata MQOPEN vengono eseguite sulla nuova coda dinamica e non sulla coda modello. Ciò è vero anche per le chiamate MQINQ e MQSET. Il nome della coda modello nel parametro **ObjDesc** viene sostituito con il nome della coda dinamica creata. Il tipo di coda dinamica è determinato dal valore dell'attributo **DefinitionType** della coda modello (consultare "Attributi per le code" a pagina 858). Per informazioni sulle opzioni di chiusura applicabili alle code dinamiche, consultare la descrizione della chiamata MQCLOSE.

Opzioni
Tipo: MQLONG - input

È possibile specificare almeno una delle seguenti opzioni:

- MQOO_SFOGLIA
- MQOO_INPUT_ * (solo uno di questi)
- MQOO_INQUIRE
- OUTPUT MQOO
- SET MQOO
- MQOO_BIND_ * (solo uno di questi)

Consultare la seguente tabella per i dettagli di queste opzioni; è possibile specificare altre opzioni come richiesto. Per specificare più di un'opzione, aggiungere i valori insieme (non aggiungere la stessa costante più di una volta) o combinare i valori utilizzando l'operazione OR bit per bit (se il linguaggio di programmazione supporta le operazioni bit). Le combinazioni non valide sono indicate; tutte le altre combinazioni sono valide. Sono consentite solo le opzioni applicabili al tipo di oggetto specificato da `ObjDesc` .

Tabella 553. Opzioni MQOPEN valide per code e argomenti

Opzione	Alias [1]	Locale e modello	Remoto	Cluster non locale	Elenco di distribuzione	Argomento
MQOO_INPUT_AS_Q_DEF	Sì	Sì	No	No	No	No
MQOO_INPUT_SHARED	Sì	Sì	No	No	No	No
MQOO_INPUT_ESCLUSIVO	Sì	Sì	No	No	No	No
MQOO_OUTPUT	Sì	Sì	Sì	Sì	Sì	Sì
MQOO_SFOGLIA	Sì	Sì	No	No	No	No
CO_MQOO	Sì	Sì	No	No	No	No
MQOO_INQUIRE	Sì	Sì	[2]	Sì	No	No
MQOO_SET	Sì	Sì	[2]	No	No	No
MQOO_BIND_ON_OPEN [3]	Sì	Sì	Sì	Sì	Sì	No
MQOO_BIND_NON_FISSO [3]	Sì	Sì	Sì	Sì	Sì	No
MQOO_BIND_ON_GROUP [3]	Sì	Sì	Sì	Sì	Sì	No
MQOO_BIND_AS_Q_DEF [3]	Sì	Sì	Sì	Sì	Sì	No
MQOO_SAVE_ALL_CONTEXT	Sì	Sì	No	No	No	No
MQOO_PASS_IDENTITY_CONTEXT	Sì	Sì	Sì	Sì	Sì	[4]
MQOO_PASS_ALL_CONTEXT	Sì	Sì	Sì	Sì	Sì	Sì
MQOO_SET_IDENTITY_CONTEXT	Sì	Sì	Sì	Sì	Sì	[4]
MQOO_SET_ALL_CONTEXT	Sì	Sì	Sì	Sì	Sì	Sì
MQOO_NO_READ_AHEAD	Sì	Sì	No	No	No	No
MQOO_READ_AHEAD	Sì	Sì	No	No	No	No
MQOO_READ_AHEAD_AS_DEF	Sì	Sì	No	No	No	No
MQOO_ALTERNATE_USER_AUTHORITY	Sì	Sì	Sì	Sì	Sì	Sì
MQOO_FAIL_IF_QUIESCING	Sì	Sì	Sì	Sì	Sì	Sì
MQOO_RESOLVE_LOCAL_Q	Sì	Sì	Sì	Sì	No	No
MQOO_RESOLVE_LOCAL_TOPIC	No	No	No	No	No	Sì
MQOO_NO_MULTICAST	No	No	No	No	No	Sì

Note:

1. La validità delle opzioni per gli alias dipende dalla validità dell'opzione per la coda in cui l'alias si risolve.
2. Questa opzione è valida solo per la definizione locale di una coda remota.
3. Questa opzione può essere specificata per qualsiasi tipo di coda, ma viene ignorata se la coda non è una coda cluster. Tuttavia, l'attributo della coda **DefBind** sovrascrive la coda di base anche quando la coda alias non è in un cluster.
4. Questi attributi possono essere utilizzati con un argomento, ma interessano solo il contesto impostato per il messaggio conservato, non i campi di contesto inviati a qualsiasi sottoscrittore.

Opzioni di accesso: le seguenti opzioni controllano il tipo di operazioni che è possibile eseguire sull'oggetto:

MQOO_INPUT_AS_Q_DEF
Aprire la coda per richiamare i messaggi utilizzando il valore predefinito definito dalla coda.

La coda viene aperta per essere utilizzata con successive chiamate MQGET. Il tipo di accesso è condiviso o esclusivo, a seconda del valore dell'attributo della coda **DefInputOpenOption** ; consultare “Attributi per le code” a pagina 858 per i dettagli.

Questa opzione è valida solo per le code locali, alias e modello; non è valida per le code remote, gli elenchi di distribuzione e gli oggetti che non sono code.

MQOO_INPUT_SHARED
Aprire la coda per richiamare i messaggi con accesso condiviso.

La coda viene aperta per essere utilizzata con successive chiamate MQGET. La chiamata può avere esito positivo se la coda è attualmente aperta da questa o da un'altra applicazione con MQOO_INPUT_SHARED, ma ha esito negativo con codice motivo MQRC_OBJECT_IN_USE se la coda è attualmente aperta con MQOO_INPUT_EXCLUSIVE.

Questa opzione è valida solo per le code locali, alias e modello; non è valida per le code remote, gli elenchi di distribuzione e gli oggetti che non sono code.

MQOO_INPUT_EXCLUSIVE
Aprire la coda per ottenere i messaggi con accesso esclusivo.

La coda viene aperta per essere utilizzata con successive chiamate MQGET. La chiamata ha esito negativo con codice motivo MQRC_OBJECT_IN_USE se la coda è attualmente aperta da questa o da un'altra applicazione per l'input di qualsiasi tipo (MQOO_INPUT_SHARED o MQOO_INPUT_EXCLUSIVE).

Questa opzione è valida solo per le code locali, alias e modello; non è valida per le code remote, gli elenchi di distribuzione e gli oggetti che non sono code.

OUTPUT MQOO

Aprire la coda per inserire i messaggi o un argomento o una stringa di argomenti per pubblicare i messaggi.

La coda o l'argomento viene aperto per essere utilizzato con le chiamate MQPUT successive.

Una chiamata MQOPEN con questa opzione può avere esito positivo anche se l'attributo della coda **InhibitPut** è impostato su MQQA_PUT_INIBITED (anche se le chiamate MQPUT successive non riescono mentre l'attributo è impostato su questo valore).

Questa opzione è valida per tutti i tipi di coda, inclusi gli elenchi di distribuzione e gli argomenti.

Le seguenti note si applicano a queste opzioni:

- È possibile specificare solo una di queste opzioni.
- Una chiamata MQOPEN con una di queste opzioni può avere esito positivo anche se l'attributo della coda **InhibitGet** è impostato su MQQA_GET_INIBITED (anche se le chiamate MQGET successive non riescono mentre l'attributo è impostato su questo valore).
- Se la coda è definita come non condivisibile (ovvero, l'attributo della coda **Shareability** ha il valore MQQA_NOT_SHAREABLE), i tentativi di aprire la coda per l'accesso condiviso vengono trattati come tentativi di aprire la coda con accesso esclusivo.
- Se una coda alias viene aperta con una di queste opzioni, la verifica per l'uso esclusivo (o per il fatto che un'altra applicazione abbia un uso esclusivo) è rispetto alla coda di base in cui l'alias si risolve.
- Queste opzioni non sono valide se **ObjectQMgrName** è il nome di un alias del gestore code; ciò è vero anche se il valore dell'attributo **RemoteQMgrName** nella definizione locale di una coda remota utilizzata per l'alias del gestore code è il nome del gestore code locale.

MQOO_SFOGLIA
Aprire la coda per esaminare i messaggi.

La coda viene aperta per essere utilizzata con le chiamate MQGET successive con una delle seguenti opzioni:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_SUCCESSIVO
- MQGMO_BROWSE_MSG_UNDER_CURSOR

Ciò è consentito anche se la coda è attualmente aperta per MQOO_INPUT_EXCLUSIVE. Una chiamata MQOPEN con l'opzione MQOO_BROWSE stabilisce un cursore di ricerca e lo posiziona logicamente prima del primo messaggio sulla coda; per ulteriori informazioni, consultare MQGMO - Campo Opzioni .

Questa opzione è valida solo per le code locali, alias e modello; non è valida per le code remote, gli elenchi di distribuzione e gli oggetti che non sono code. Inoltre, non è valido se `ObjectQMgrName` è il nome di un alias del gestore code; ciò è vero anche se il valore dell'attributo **RemoteQMgrName** nella definizione locale di una coda remota utilizzata per l'alias del gestore code è il nome del gestore code locale.

CO_MQOO
Aprire come membro cooperante della serie di maniglie.

Questa opzione è valida solo con l'opzione MQOO_BROWSE. Se viene specificato senza MQOO_BROWSE, MQOPEN restituisce MQRC_OPTIONS_ERROR.

L'handle restituito viene considerato come membro di una serie di handle cooperante per le successive chiamate MQGET con una delle seguenti opzioni:

- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNMARKED_BROWSE_MSG
- MQGMO_UNMARK_BROWSE_CO_OP

Questa opzione è valida solo per le code locali, alias e modello; non è valida per le code remote, gli elenchi di distribuzione e gli oggetti che non sono code.

MQOO_INQUIRE
Aprire l'oggetto per interrogare gli attributi.

La coda, l'elenco dei nomi, la definizione del processo o il gestore code viene aperto per essere utilizzato con le chiamate MQINQ successive.

Questa opzione è valida per tutti i tipi di oggetto diversi dagli elenchi di distribuzione. Non è valido se `ObjectQMgrName` è il nome di un alias del gestore code; ciò è vero anche se il valore dell'attributo **RemoteQMgrName** nella definizione locale di una coda remota utilizzata per l'alias del gestore code è il nome del gestore code locale.

SET MQOO
Aprire la coda per impostare gli attributi.

La coda viene aperta per essere utilizzata con le chiamate MQSET successive.

Questa opzione è valida per tutti i tipi di coda diversi dagli elenchi di distribuzione. Non è valido se `ObjectQMgrName` è il nome di una definizione locale di una coda remota; ciò è vero anche se il valore dell'attributo **RemoteQMgrName** nella definizione locale di una coda remota utilizzata per l'alias del gestore code è il nome del gestore code locale.

Opzioni di bind: le seguenti opzioni si applicano quando l'oggetto che si sta aprendo è una coda del cluster; queste opzioni controllano il collegamento dell'handle della coda a una istanza della coda del cluster:

MQOO_BIND_ON_OPEN
Il gestore code locale collega l'handle della coda a un'istanza della coda di destinazione quando la coda viene aperta. Di conseguenza, tutti i messaggi inseriti utilizzando questo handle vengono inviati alla stessa istanza della coda di destinazione e allo stesso instradamento.

Questa opzione è valida solo per le code e interessa solo le code cluster. Se specificata per una coda che non è una coda cluster, l'opzione viene ignorata.

MQOO_BIND_NON_FISSO
Questo arresta il gestore code locale che esegue il bind dell'handle della coda a un'istanza della coda di destinazione. Di conseguenza, le successive chiamate MQPUT che utilizzano questo handle inviano i messaggi a istanze differenti della coda di destinazione o alla stessa istanza ma tramite instradamenti differenti. Inoltre, consente all'istanza selezionata di essere modificata successivamente dal gestore code locale, da un gestore code remoto o da un agent MCA (message channel agent), in base alle condizioni di rete.

Nota: Le applicazioni client e server che devono scambiare una serie di messaggi per completare una transazione, non devono utilizzare MQOO_BIND_NOT_FIXED (o MQOO_BIND_AS_Q_DEF quando `DefBind` ha il valore MQBND_BIND_NOT_FIXED), poiché i messaggi successivi nella serie potrebbero essere inviati a istanze differenti dell'applicazione server.

Se MQOO_BROWSE o una delle opzioni MQOO_INPUT_ * è specificata per una coda cluster, il gestore code viene forzato a selezionare l'istanza locale della coda cluster. Di conseguenza, il bind dell'handle della coda è fisso, anche se è specificato MQOO_BIND_NOT_FIXED.

Se MQOO_INQUIRE è specificato con MQOO_BIND_NOT_FIXED, le chiamate MQINQ successive che utilizzano tale handle potrebbero richiedere istanze differenti della coda del cluster, anche se in genere tutte le istanze hanno gli stessi valori di attributo.

MQOO_BIND_NOT_FIXED è valido solo per le code e interessa solo le code cluster. Se specificata per una coda che non è una coda cluster, l'opzione viene ignorata.

Gruppo_BIND_MQOO
Consente a una applicazione di richiedere che un gruppo di messaggi sia assegnato alla stessa istanza di destinazione.

Questa opzione è valida solo per le code e interessa solo le code cluster. Se specificata per una coda che non è una coda cluster, l'opzione viene ignorata.

MQOO_BIND_AS_Q_DEF
Il gestore code locale esegue il bind dell'handle di coda nel modo definito dall'attributo della coda **`DefBind`** . Il valore di questo attributo è MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXED o MQBND_BIND_ON_GROUP.

MQOO_BIND_AS_Q_DEF è il valore predefinito quando MQOO_BIND_ON_OPEN, MQOO_BIND_NOT_FIXED o MQOO_BIND_ON_GROUP non è specificato.

MQOO_BIND_AS_Q_DEF aiuta la documentazione del programma. Non è previsto che questa opzione venga utilizzata con una delle altre due opzioni di collegamento, ma poiché il relativo valore è zero tale utilizzo non può essere rilevato.

Opzioni di contesto: le seguenti opzioni controllano l'elaborazione del contesto del messaggio:

MQOO_SAVE_ALL_CONTEXT
Le informazioni di contesto sono associate a questo handle di coda. Queste informazioni vengono impostate dal contesto di qualsiasi messaggio richiamato utilizzando questo handle. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Queste informazioni di contesto possono essere inoltrate a un messaggio che viene quindi inserito in una coda utilizzando le chiamate MQPUT o MQPUT1 . Consultare le opzioni MQPMO_PASS_IDENTITY_CONTEXT e MQPMO_PASS_ALL_CONTEXT descritte in “MQPMO - Opzioni inserimento messaggio” a pagina 512.

Finché un messaggio non viene richiamato correttamente, il contesto non può essere passato a un messaggio inserito in una coda.

Un messaggio richiamato utilizzando una delle opzioni MQGMO_BROWSE_ * browse non ha le informazioni di contesto salvate (anche se i campi di contesto nel parametro **`MsgDesc`** sono impostati dopo una ricerca).

Questa opzione è valida solo per le code locali, alias e modello; non è valida per le code remote, gli elenchi di distribuzione e gli oggetti che non sono code. È necessario specificare una delle opzioni MQOOO_INPUT_ *.

MQOO_PASS_IDENTITY_CONTEXT
Ciò consente all'opzione MQPMO_PASS_IDENTITY_CONTEXT di essere specificata nel parametro **PutMsgOpts** quando un messaggio viene inserito in una coda; ciò fornisce al messaggio le informazioni sul contesto di identità da una coda di input aperta con l'opzione MQOO_SAVE_ALL_CONTEXT. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

È necessario specificare l'opzione MQOO_OUTPUT.

Questa opzione è valida per tutti i tipi di coda, inclusi gli elenchi di distribuzione.

MQOO_PASS_ALL_CONTEXT
Ciò consente all'opzione MQPMO_PASS_ALL_CONTEXT di essere specificata nel parametro **PutMsgOpts** quando un messaggio viene inserito su una coda; ciò fornisce al messaggio le informazioni sul contesto di origine e identità da una coda di input aperta con l'opzione MQOO_SAVE_ALL_CONTEXT. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Questa opzione implica MQOO_PASS_IDENTITY_CONTEXT, che non è necessario specificare. È necessario specificare l'opzione MQOO_OUTPUT.

Questa opzione è valida per tutti i tipi di coda, inclusi gli elenchi di distribuzione.

MQOO_SET_IDENTITY_CONTEXT
Ciò consente all'opzione MQPMO_SET_IDENTITY_CONTEXT di essere specificata nel parametro **PutMsgOpts** quando un messaggio viene inserito in una coda; ciò fornisce al messaggio le informazioni sul contesto identità contenute nel parametro **MsgDesc** specificato nella chiamata MQPUT o MQPUT1 . Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Questa opzione implica MQOO_PASS_IDENTITY_CONTEXT, che non è necessario specificare. È necessario specificare l'opzione MQOO_OUTPUT.

Questa opzione è valida per tutti i tipi di coda, inclusi gli elenchi di distribuzione.

MQOO_SET_ALL_CONTEXT
Ciò consente all'opzione MQPMO_SET_ALL_CONTEXT di essere specificata nel parametro **PutMsgOpts** quando un messaggio viene inserito in una coda; ciò fornisce al messaggio le informazioni sul contesto di origine e di identità contenute nel parametro **MsgDesc** specificato nella chiamata MQPUT o MQPUT1 . Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Questa opzione implica le opzioni seguenti, che non è necessario specificare:

- MQOO_PASS_IDENTITY_CONTEXT
- MQOO_PASS_ALL_CONTEXT
- MQOO_SET_IDENTITY_CONTEXT

È necessario specificare l'opzione MQOO_OUTPUT.

Questa opzione è valida per tutti i tipi di coda, inclusi gli elenchi di distribuzione.

Opzioni di lettura anticipata

Quando si richiama MQOPEN con MQOO_READ_AHEAD, il client IBM MQ abilita la lettura anticipata solo se sono soddisfatte determinate condizioni. Queste condizioni includono:

- L'applicazione client deve essere compilata e collegata rispetto alle librerie client MQI IBM MQ in thread.
- Il canale del client deve utilizzare il protocollo TCP/IP

- Il canale deve avere un'impostazione SharingConversations (SHARECNV) diversa da zero nelle definizioni di canale del client e del server.

Le seguenti opzioni controllano se i messaggi non persistenti vengono inviati al client prima che un'applicazione li richieda. Le seguenti note si applicano alle opzioni di lettura anticipata:

- È possibile specificare solo una di queste opzioni.
- Queste opzioni sono valide solo per code locali, alias e modello. Non sono validi per code remote, elenchi di distribuzione, argomenti o gestori code.
- Queste opzioni sono applicabili solo quando viene specificata anche una delle opzioni MQOO_BROWSE, MQOO_INPUT_SHARED e MQOO_INPUT_EXCLUSIVE, sebbene non sia un errore specificare queste opzioni con MQOO_INQUIRE o MQOO_SET.
- Se l'applicazione non è in esecuzione come client IBM MQ , queste opzioni vengono ignorate.

MQOO_NO_READ_AHEAD
I messaggi non persistenti non vengono inviati al client prima che un'applicazione li richieda.

MQOO_READ_AHEAD
I messaggi non persistenti vengono inviati al client prima che un'applicazione li richieda.

MQOO_READ_AHEAD_AS_Q_DEF
Il comportamento di lettura anticipata è determinato dall'attributo di lettura anticipata predefinito della coda aperta. Questo è il valore predefinito.

Altre opzioni: le seguenti opzioni controllano il controllo dell'autorizzazione, cosa accade quando il gestore code è in fase di sospensione, se risolvere il nome della coda locale e multicast:

MQOO_ALTERNATE_USER_AUTHORITY
Il campo *AlternateUserId* nel parametro **ObjDesc** contiene un identificativo utente da utilizzare per convalidare questa chiamata MQOPEN. La chiamata può avere esito positivo solo se questo *AlternateUserId* è autorizzato ad aprire l'oggetto con le opzioni di accesso specificate, indipendentemente dal fatto che l'identificativo utente con cui l'applicazione è in esecuzione sia autorizzato a farlo. Ciò non si applica alle opzioni di contesto specificate, tuttavia, che sono sempre controllate rispetto all'identificativo utente con cui è in esecuzione l'applicazione.

Questa opzione è valida per tutti i tipi di oggetto.

MQOO_FAIL_IF_QUIESCING
La chiamata MQOPEN ha esito negativo se il gestore code è in stato di sospensione.

z/OS Su z/OS, per un'applicazione CICS o IMS , questa opzione forza anche l'esito negativo della chiamata MQOPEN se la connessione è in stato di inattività.

Questa opzione è valida per tutti i tipi di oggetto.

Per informazioni sui canali client, consultare IBM MQ MQI clients.

MQOO_RESOLVE_LOCAL_Q
Riempire il campo ResolvedQName nella struttura MQOD con il nome della coda locale che è stata aperta. Allo stesso modo, il nome ResolvedQMgrviene riempito con il nome del gestore code locale che ospita la coda locale. Se la struttura MQOD è inferiore alla Versione 3, MQOO_RESOLVE_LOCAL_Q viene ignorato senza che venga restituito alcun errore.

La coda locale viene sempre restituita quando viene aperta una coda locale, alias o modello, ma ciò non si verifica quando, ad esempio, una coda remota o una coda cluster non locale viene aperta senza l'opzione MQOO_RESOLVE_LOCAL_Q; il nome ResolvedQName e ResolvedQMgrvengono riempiti con il nome RemoteQName e RemoteQMgrtrovato nella definizione della coda remota o in modo simile con la coda cluster remota scelta.

Se si specifica MQOO_RESOLVE_LOCAL_Q quando si apre, ad esempio, una coda remota, ResolvedQName è la coda di trasmissione in cui vengono inseriti i messaggi. Il nome ResolvedQMgrviene riempito con il nome del gestore code locale che ospita la coda di trasmissione.

Se si è autorizzati per la ricerca, l'input o l'output su una coda, si dispone dell'autorità richiesta per specificare questo indicatore sulla chiamata MQOPEN. Non è necessaria alcuna autorizzazione speciale.

Questa opzione è valida solo per code e gestori code.

MQOO_RESOLVE_LOCAL_TOPIC
Compilare il campo ResolvedQName nella struttura MQOD con il nome dell'argomento amministrativo aperto.

MQOO_NO_MULTICAST
I messaggi di pubblicazione non vengono inviati utilizzando multicast.

Questa opzione è valida solo con l'opzione MQOO_OUTPUT. Se viene specificato senza MQOO_OUTPUT, MQOPEN restituisce MQRC_OPTIONS_ERROR.

Questa opzione è valida solo per un argomento.

HOBJ
Tipo: MQHOBJ - output

Questo handle rappresenta l'accesso stabilito all'oggetto. Deve essere specificato nelle successive chiamate IBM MQ che operano sull'oggetto. Cessa di essere valida quando viene emessa la chiamata MQCLOSE o quando termina l'unità di elaborazione che definisce l'ambito dell'handle.

L'ambito dell'handle dell'oggetto restituito è lo stesso dell'ambito dell'handle di collegamento specificato nella chiamata. Consultare parametro MQCONN - Hconn per informazioni sull'ambito della gestione.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Il codice di errore che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

MQRC_MULTIPLE_MOTIVI
(2136, X'858 ') Sono stati restituiti più codici di errore.

Se *CompCode* è MQCC_FAILED:

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'89C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

ERRORE TIPO_Q_ALIAS_MQRC
(2001, X'7D1') La coda di base dell'alias non è un tipo valido.

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita.

ERRORE USCITA MQRC_API
(2183, X'887 ') Impossibile caricare l'uscita API.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

MQRC_CF_NOT_AVAILABLE
(2345, X' 929 ') La funzione di accoppiamento non è disponibile.

MQRC_CF_STRUC_AUTH_NON RIUSCITO
(2348, X'92C') Il controllo dell'autorizzazione della struttura CFS non è riuscito.

ERRORE MQRC_CF_STRUC_
(2349, X'92D') Struttura CF non valida.

MQRC_CF_STRUC_NON RIUSCITO
(2373, X' 945 ') La struttura della funzione di accoppiamento non è riuscita.

MQRC_CF_STRUC_IN_USO
(2346, X'92A') Struttura CF in uso.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') L'elenco - intestazione della struttura CFS (Coupling Facility Structure) è in uso.

MQRC_CICS_WAIT_NON RIUSCITO
(2140, X'85C') Richiesta di attesa rifiutata da CICS.

ERRORE DI USCITA MQRC_CLUSTER_
(2266, X'8DA') Uscita carico di lavoro cluster non riuscita.

MQRC_CLUSTER_PUT_INIBITO
(2268, X'8DC') Chiamate Put inibite per tutte le code nel cluster.

MQRC_CLUSTER_RESOLUTION_ERRORE
(2189, X'88D') Risoluzione del nome cluster non riuscita.

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Errore di risorsa cluster.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Non autorizzato per la connessione.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Connessione in fase di sospensione.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Chiusura della connessione.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') Db2 sottosistema non disponibile.

MQRC_DEF_XMIT_Q_TYPE_ERROR
(2198, X'896 ') La coda di trasmissione predefinita non è locale.

MQRC_DEF_XMIT_Q_USAGE_ERROR
(2199, X'897 ') Errore di utilizzo della coda di trasmissione predefinita.

MQRC_DYNAMIC_Q_NAME_ERROR
(2011, X'7DB') Nome della coda dinamica non valido.

MQRC_HANDLE_NON_DISPONIBILE
(2017, X'7E1') Nessun ulteriore handle disponibile.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

ERRORE MQRC_HOBJ_R
(2019, X'7E3') Handle oggetto non valido.

MQRC_MULTIPLE_MOTIVI
(2136, X'858 ') Sono stati restituiti più codici di errore.

NAME_MQRC_IN_USE
(2201, X'899 ') Nome in uso.

MQRC_NAME_NOT_VALID_FOR_TYPE
(2194, X'892 ') Nome oggetto non valido per il tipo di oggetto.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

MQRC_OBJECT_ALREADY_EXISTS
(2100, X'834 ') L'oggetto esiste.

MQRC_OBJECT_DAMAGED
(2101, X'835 ') Oggetto danneggiato.

MQRC_OBJECT_IN_USE
(2042, X'7FA') Oggetto già aperto con opzioni in conflitto.

MQRC_OBJECT_LEVEL_INCOMPATIBILE
(2360, X' 938 ') Livello oggetto non compatibile.

ERRORE MQRC_OBJECT_NAME_ERROR
(2152, X'868 ') Nome oggetto non valido.

MQRC_OBJECT_NOT_UNIQUE
(2343, X' 927 ') Oggetto non univoco.

MQRC_OBJECT_Q_MGR_NAME_ERROR
(2153, X'869 ') Nome gestore code oggetto non valido.

ERRORE MQRC_OBJECT_RECORDS
(2155, X'86B') Record oggetto non validi.

ERRORE STRINGA MQRC_OBJECT_
(2441, X'0989 ') Campo Objectstring non valido

ERRORE TIPO_OGGETTO_MQRC
(2043, X'7FB') Tipo oggetto non valido.

ERRORE MQRC_O
(2044, X'7FC') Struttura descrittore oggetto non valida.

MQRC_OPTION_NOT_VALID_FOR_TYPE
(2045, X'7FD') Opzione non valida per il tipo di oggetto.

ERRORE MQRC_OPTIONS_
(2046, X'7FE') Opzioni non valide o non congruenti.

ERRORE MQRC_PAGESET_
(2193, X'891 ') Errore durante l'accesso al dataset della serie di pagine.

MQRC_PAGESET_FULL
(2192, X'890 ') Il supporto di memoria esterna è pieno.

MQRC_Q_XX_ENCODE_CASE_ONE eliminato
(2052, X'804 ') La coda è stata eliminata.

ERRORE MQRC_Q_MGR_NAME_
(2058, X'80A') Nome gestore code non valido o sconosciuto.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Gestore code non disponibile per la connessione.

MQRC_Q_MGR_QUIESCING
(2161, X'871 ') Gestore code in fase di sospensione.

MQRC_Q_MGR_STOPPING
(2162, X'872 ') Chiusura del gestore code.

ERRORE MQRC_Q_TYPE_
(2057, X'809 ') Tipo coda non valido.

ERRORE MQRC_RECS_PRESENT_
(2154, X'86A') Numero di record presenti non valido.

MQRC_REMOTE_Q_NAME_ERROR
(2184, X'888 ') Nome coda remota non valido.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

ERRORE DI RISPOSTA MQR_RESPONSE_RECORDS
(2156, X'86C') Record di risposta non validi.

ERRORE MQRC_SECURITY_ERROR
(2063, X'80F') Si è verificato un errore di sicurezza.

ERRORE MQRC_SELECTOR_SYNTAX_ERROR
2459 (X'099B') È stata emessa una chiamata MQOPEN, MQPUT1 o MQSUB ma è stata specificata una stringa di selezione contenente un errore di sintassi.

MQRC_STOPPED_BY_CLUSTER_EXIT
(2188, X'88C') Chiamata rifiutata dall'uscita del carico di lavoro del cluster.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890 ') Il supporto di memoria esterna è pieno.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Chiamata eliminata dal programma di uscita.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

MQRC_UNKNOWN_ALIAS_BASE_Q
(2082, X'822 ') Coda di base alias sconosciuta.

MQRC_UNKNOWN_DEF_XMIT_Q
(2197, X'895 ') Coda di trasmissione predefinita sconosciuta.

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825 ') Nome oggetto sconosciuto.

MQRC_UNKNOWN_OBJECT_Q_MGR
(2086, X'826 ') Gestore code oggetti sconosciuto.

MQRC_UNKNOWN_REMOTE_Q_MGR
(2087, X'827 ') Gestore code remoto sconosciuto.

MQRC_UNKNOWN_XMIT_Q
(2196, X'894 ') Coda di trasmissione sconosciuta.

MQRC_WRONG_CF_LEVEL
(2366, X'93E') La struttura della CF (Coupling Facility) è di livello errato.

MQRC_XMIT_Q_TYPE_ERROR
(2091, X'82B') Coda di trasmissione non locale.

MQRC_XMIT_Q_USAGE_ERRORE
(2092, X'82C') Coda di trasmissione con utilizzo errato.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note generali sull'utilizzo

1. L'oggetto aperto è uno dei seguenti:
   - Una coda per:
     - Richiamare o sfogliare i messaggi (utilizzando la chiamata MQGET)
     - Inserire i messaggi (utilizzando la chiamata MQPUT)

– Interrogare sugli attributi della coda (utilizzando la chiamata MQINQ)
– Impostare gli attributi della coda (utilizzando la chiamata MQSET)

Se la coda denominata è una coda modello, viene creata una coda locale dinamica. Consultare il parametro **ObjDesc** descritto in “MQOPEN - Apri oggetto” a pagina 751.

Un elenco di distribuzione è un tipo speciale di oggetto coda che contiene un elenco di code. Può essere aperto per inserire messaggi, ma non per ottenere o sfogliare messaggi o per interrogare o impostare attributi. Consultare la nota di utilizzo 8 per ulteriori dettagli.

Una coda che ha QSGDISP(GROUP) è un tipo speciale di definizione della coda che non può essere utilizzato con le chiamate MQOPEN o MQPUT1 .

- Un elenco nomi per richiedere informazioni sui nomi delle code nell'elenco (utilizzando la chiamata MQINQ).
- Una definizione di processo per analizzare gli attributi del processo (utilizzando la chiamata MQINQ).
- Il gestore code per richiedere informazioni sugli attributi del gestore code locale (utilizzando la chiamata MQINQ).
- Un argomento per pubblicare un messaggio (utilizzando la chiamata MQPUT)

2. Un'applicazione può aprire lo stesso oggetto più di una volta. Viene restituito un handle di oggetto diverso per ogni apertura. Ogni handle restituito può essere utilizzato per le funzioni per cui è stato eseguito il corrispondente open.
3. Se l'oggetto che si sta aprendo è una coda diversa da una coda cluster, la risoluzione di tutti i nomi all'interno del gestore code locale avviene al momento della chiamata MQOPEN. Questo può includere:
   - Risoluzione del nome di una definizione locale di una coda remota sul nome del gestore code remoto e il nome con cui la coda è nota sul gestore code remoto
   - Risoluzione del nome del gestore code remoto sul nome di una coda di trasmissione locale
   - z/OS Solo su z/OS , la risoluzione del nome del gestore code remoto al nome della coda di trasmissione condivisa utilizzata dall'agente IGQ (si applica solo se i gestori code locali e remoti appartengono allo stesso gruppo di condivisione code)
   - Risoluzione alias per il nome di una coda di base o di un oggetto argomento.

   Tuttavia, tenere presente che le chiamate MQINQ o MQSET successive per l'handle si riferiscono esclusivamente al nome che è stato aperto e non all'oggetto risultante dopo la risoluzione del nome. Ad esempio, se l'oggetto aperto è un alias, gli attributi restituiti dalla chiamata MQINQ sono gli attributi dell'alias, non gli attributi della coda di base o un oggetto argomento in cui l'alias si risolve.

   Se l'oggetto che si sta aprendo è una coda cluster, la risoluzione del nome può verificarsi al momento della chiamata MQOPEN o essere rinviata fino a un momento successivo. Il punto in cui si verifica la risoluzione è controllato dalle opzioni MQOO_BIND_ * specificate sulla chiamata MQOPEN:

   - MQOO_BIND_ON_OPEN
   - MQOO_BIND_NON_FISSO
   - MQOO_BIND_AS_Q_DEF
   - Gruppo_BIND_MQOO

   Per ulteriori informazioni sulla risoluzione dei nomi per le code cluster, consultare Risoluzione dei nomi .
4. Una chiamata MQOPEN con l'opzione MQOO_BROWSE stabilisce un cursore di esplorazione, da utilizzare con le chiamate MQGET che specificano l'handle dell'oggetto e una delle opzioni. Ciò consente di eseguire la scansione della coda senza modificarne il contenuto. Un messaggio che è stato trovato dalla ricerca può essere rimosso dalla coda utilizzando l'opzione MQGMO_MSG_UNDER_CURSOR.

   Più cursori di esplorazione possono essere attivi per una singola applicazione emettendo diverse richieste MQOPEN per la stessa coda.

5. Alle applicazioni avviate da un controllo trigger viene passato il nome della coda associata all'applicazione quando questa viene avviata. Questo nome coda può essere specificato nel parametro **ObjDesc** per aprire la coda. Consultare "MQTMC2 - Messaggio di attivazione 2 (formato carattere)" a pagina 622 per ulteriori dettagli.

Opzioni di lettura anticipata

Quando si richiama MQOPEN con MQOO_READ_AHEAD, il client IBM MQ abilita la lettura anticipata solo se sono soddisfatte determinate condizioni. Queste condizioni includono:

- L'applicazione client deve essere compilata e collegata rispetto alle librerie client MQI IBM MQ in thread.
- Il canale del client deve utilizzare il protocollo TCP/IP
- Il canale deve avere un'impostazione SharingConversations (SHARECNV) diversa da zero nelle definizioni di canale del client e del server.

Le seguenti note si applicano all'utilizzo delle opzioni di lettura anticipata.

1. Le opzioni di lettura anticipata sono applicabili solo quando vengono specificate una e solo una delle opzioni MQOO_BROWSE, MQOO_INPUT_SHARED e MQOO_INPUT_EXCLUSIVE. Non viene generato un errore se le opzioni di lettura anticipata sono specificate con le opzioni MQOO_ INQUIRE o MQOO_SET.
2. La lettura anticipata non è abilitata quando viene richiesta se le opzioni utilizzate nella prima chiamata MQGET non sono supportate per l'utilizzo con la lettura anticipata. Inoltre, la lettura anticipata è disabilitata quando il client si connette a un gestore code che non supporta la lettura anticipata.
3. Se l'applicazione non è in esecuzione come client IBM MQ , le opzioni di lettura anticipata vengono ignorate.

Code cluster

Le seguenti note si applicano all'utilizzo delle code cluster.

1. Quando una coda del cluster viene aperta per la prima volta e il gestore code locale non è un gestore code del repository completo, il gestore code locale ottiene le informazioni sulla coda del cluster da un gestore code del repository completo. Quando la rete è occupata, il gestore code locale può impiegare diversi secondi per ricevere le informazioni necessarie dal gestore code del repository. Di conseguenza, l'applicazione che emette la chiamata MQOPEN potrebbe dover attendere fino a 10 secondi prima che il controllo ritorni dalla chiamata MQOPEN. Se il gestore code locale non riceve le informazioni necessarie sulla coda del cluster entro questo periodo di tempo, la chiamata non riesce con il codice motivo MQRC_CLUSTER_RESOLUTION_ERROR.
2. Quando una coda cluster viene aperta e sono presenti più istanze della coda nel cluster, l'istanza aperta dipende dalle opzioni specificate sulla chiamata MQOPEN:
   - Se le opzioni specificate includono una delle seguenti:
     - MQOO_SFOGLIA
     - MQOO_INPUT_AS_Q_DEF
     - MQOO_INPUT_EXCLUSIVE
     - MQOO_INPUT_SHARED
     - SET MQOO

     l'istanza della coda cluster aperta deve essere l'istanza locale. Se non è presente alcuna istanza locale della coda, la chiamata MQOPEN non riesce.
   - Se le opzioni specificate non includono alcuna delle opzioni descritte in precedenza, ma includono una o entrambe le seguenti:
     - MQOO_INQUIRE
     - OUTPUT MQOO

l'istanza aperta è l'istanza locale, se presente, e un'istanza remota, altrimenti (se si utilizzano i valori predefiniti di CLWLUSEQ). Tuttavia, l'istanza scelta dal gestore code può essere modificata da un'uscita del carico di lavoro del cluster (se presente).

3. Se è presente una sottoscrizione per la coda, ma non è riconosciuta da un repository completo, l'oggetto non è presente nel cluster e la chiamata ha esito negativo con codice motivo MQRC_OBJECT_NAME.

Per ulteriori informazioni sulle code cluster, consultare Code cluster.

Liste di distribuzione

Le seguenti note si applicano all'uso degli elenchi di distribuzione.

Gli elenchi di distribuzione sono supportati nei seguenti ambienti:

- 

AIX
- 

IBM i
- 

Linux
- 

Windows

e per IBM MQ MQI clients collegati a questi sistemi.

1. I campi nella struttura MQOD devono essere impostati come segue quando si apre un elenco di distribuzione:
   - `Version` deve essere MQOD_VERSION_2 o superiore.
   - `ObjectType` deve essere MQOT_Q.
   - `ObjectName` deve essere vuoto o la stringa null.
   - `ObjectQMgrName` deve essere vuoto o la stringa null.
   - `RecsPresent` Deve essere maggiore di zero.
   - Uno tra `ObjectRecOffset` e `ObjectRecPtr` deve essere zero e l'altro diverso da zero.
   - Non più di uno tra `ResponseRecOffset` e `ResponseRecPtr` può essere diverso da zero.
   - Devono essere presenti `RecsPresent` record oggetto, indirizzati da `ObjectRecOffset` o `ObjectRecPtr`. I record oggetto devono essere impostati sui nomi delle code di destinazione da aprire.
   - Se uno tra `ResponseRecOffset` e `ResponseRecPtr` è diverso da zero, devono essere presenti `RecsPresent` record di risposta. Vengono impostati dal gestore code se la chiamata viene completata con il codice motivo MQRC_MULTIPLE_REASON.

   Un MQOD version-2 può essere utilizzato anche per aprire una singola coda che non si trova in un elenco di distribuzione, assicurandosi che `RecsPresent` sia zero.
2. Nel parametro **`Options`** sono valide solo le seguenti opzioni di apertura:
   - OUTPUT MQOO
   - PASSO_MQOO_ * _CONTESTO
   - MQOO_SET_ * _CONTESTO
   - MQOO_ALTERNATE_USER_AUTHORITY
   - MQOO_FAIL_IF_QUIESCING
3. Le code di destinazione nell'elenco di distribuzione possono essere code locali, alias o remote, ma non possono essere code modello. Se viene specificata una coda modello, l'apertura di tale coda ha esito negativo, con codice motivo MQRC_Q_TYPE_ERROR. Tuttavia, ciò non impedisce che altre code nell'elenco vengano aperte correttamente.
4. I parametri del codice di completamento e del codice di errore sono impostati come segue:

- Se le operazioni di apertura per le code nell'elenco di distribuzione hanno avuto esito positivo o negativo nello stesso modo, i parametri del codice di completamento e del codice motivo vengono impostati per descrivere il risultato comune. I record di risposta MQRR (se forniti dall'applicazione) non sono impostati in questo caso.

  Ad esempio, se ogni apertura ha esito positivo, il codice di completamento è impostato su MQCC_OK e il codice motivo è impostato su MQRC_NONE; se ogni apertura ha esito negativo perché non esiste alcuna coda, i parametri sono impostati su MQCC_FAILED e MQRC_UNKNOWN_OBJECT_NAME.
- Se le operazioni di apertura per le code nell'elenco di distribuzione non hanno esito positivo o negativo nello stesso modo:
  - Il parametro del codice di completamento è impostato su MQCC_WARNING se almeno un'apertura ha avuto esito positivo e su MQCC_FAILED se tutti hanno avuto esito negativo.
  - Il parametro del codice di errore è impostato su MQRC_MULTIPLE_REASON.
  - I record di risposta (se forniti dall'applicazione) sono impostati sui codici di completamento individuali e sui codici motivo per le code nell'elenco di distribuzione.

5. Quando un elenco di distribuzione è stato aperto correttamente, l'handle `Hobj` restituito dalla chiamata può essere utilizzato nelle chiamate MQPUT successive per inserire i messaggi nelle code nell'elenco di distribuzione e in una chiamata MQCLOSE per rinunciare all'accesso all'elenco di distribuzione. L'unico opzione di chiusura valida per un elenco di distribuzione è MQCO_NONE.

   La chiamata MQPUT1 può essere utilizzata anche per inserire un messaggio in un elenco di distribuzione; la struttura MQOD che definisce le code nell'elenco viene specificata come parametro su tale chiamata.
6. Ogni destinazione aperta correttamente nell'elenco di distribuzione viene contata come un handle separato quando si verifica se l'applicazione ha superato il numero massimo consentito di handle (consultare l'attributo del gestore code **MaxHandles**). Ciò è vero anche quando due o più destinazioni nell'elenco di distribuzione si risolvono nella stessa coda fisica. Se la chiamata MQOPEN o MQPUT1 per un elenco di distribuzione fa sì che il numero di handle utilizzati dall'applicazione superi `MaxHandles`, la chiamata ha esito negativo con codice motivo MQRC_HANDLE_NOT_AVAILABLE.
7. Ogni destinazione aperta correttamente ha il valore del relativo attributo **OpenOutputCount** incrementato di uno. Se due o più destinazioni nell'elenco di distribuzione si risolvono nella stessa coda fisica, il relativo attributo **OpenOutputCount** viene incrementato del numero di destinazioni nell'elenco di distribuzione che si risolvono in tale coda.
8. Qualsiasi modifica alle definizioni di coda che avrebbe causato la non validità di un handle se le code fossero state aperte singolarmente (ad esempio, una modifica nel percorso di risoluzione), non fa sì che l'handle dell'elenco di distribuzione diventi non valido. Tuttavia, si verifica un errore per quella particolare coda quando l'handle dell'elenco di distribuzione viene utilizzato su una successiva chiamata MQPUT.
9. Un elenco di distribuzione può contenere una sola destinazione.

Code remote

Le seguenti note si applicano all'utilizzo delle code remote.

Una coda remota può essere specificata in due modi nel parametro **ObjDesc** di questa chiamata.

- Specificando per `ObjectName` il nome di una definizione locale della coda remota. In questo caso, `ObjectQMgrName` si riferisce al gestore code locale e può essere specificato come spazio vuoto o (nel linguaggio di programmazione C) come una stringa nulla.

  La convalida di sicurezza eseguita dal gestore code locale verifica che l'utente sia autorizzato ad aprire la definizione locale della coda remota.
- Specificando per `ObjectName` il nome della coda remota come è noto al gestore code remoto. In questo caso `ObjectQMgrName` è il nome del gestore code remoto.

  La convalida di sicurezza eseguita dal gestore code locale verifica che l'utente sia autorizzato a inviare i messaggi alla coda di trasmissione risultante dal processo di risoluzione dei nomi.

In entrambi i casi:

- Nessun messaggio viene inviato dal gestore code locale al gestore code remoto per controllare che l'utente sia autorizzato a inserire messaggi nella coda.
- Quando un messaggio arriva al gestore code remoto, il gestore code remoto potrebbe rifiutarlo perché l'utente che ha originato il messaggio non è autorizzato.

Per ulteriori informazioni, consultare i campi `ObjectName` e `ObjectQMgrName` descritti in "MQOD - Descrittore oggetto" a pagina 492 .

Oggetti

Sicurezza

Le seguenti note si riferiscono agli aspetti di sicurezza dell'utilizzo di MQOPEN.

Il gestore code esegue i controlli di sicurezza quando viene emessa una chiamata MQOPEN, per verificare che l'identificativo utente con cui l'applicazione è in esecuzione disponga del livello di autorizzazione appropriato prima che sia consentito l'accesso. Il controllo dell'autorizzazione viene eseguito sul nome dell'oggetto che si sta aprendo e non sul nome o sui nomi, come risultato dopo che un nome è stato risolto.

Se l'oggetto aperto è una coda alias che punta a un oggetto argomento, il gestore code esegue un controllo di sicurezza sul nome della coda alias, prima di eseguire un controllo di sicurezza per l'argomento come se l'oggetto argomento fosse stato utilizzato direttamente.

Se l'oggetto aperto è un oggetto argomento, con `ObjectName` da solo o utilizzando `ObjectString` (con o senza un `ObjectName`di base), il gestore code esegue il controllo di sicurezza utilizzando la stringa di argomenti risultante, ricavata dall'oggetto argomento specificato in `ObjectName`e, se necessario, concatenandolo con quello fornito in `ObjectString`, e quindi individuando l'oggetto argomento più vicino o superiore a tale punto nella struttura ad albero dell'argomento per eseguire il controllo di sicurezza. Potrebbe non essere lo stesso oggetto argomento specificato in `ObjectName`.

Se l'oggetto che si sta aprendo è una coda modello, il gestore code esegue un controllo di sicurezza completo rispetto sia al nome della coda modello che al nome della coda dinamica che viene creata. Se la coda dinamica risultante viene aperta esplicitamente, viene eseguito un ulteriore controllo di sicurezza della risorsa rispetto al nome della coda dinamica.

z/OS In z/OS, il gestore code esegue i controlli di sicurezza solo se la sicurezza è abilitata. Per ulteriori informazioni sul controllo di sicurezza, consultare Impostazione della sicurezza su z/OS .

Attributi

Le seguenti note sono relative agli attributi.

Gli attributi di un oggetto possono essere modificati mentre un'applicazione ha l'oggetto aperto. In molti casi, l'applicazione non lo nota, ma per alcuni attributi il gestore code contrassegna l'handle come non più valido. Questi attributi sono:

- Qualsiasi attributo che influisce sulla risoluzione del nome dell'oggetto. Ciò si applica indipendentemente dalle opzioni aperte utilizzate e include quanto segue:
  - Una modifica all'attributo **BaseQName** di una coda alias aperta.
  - Una modifica all'attributo **TargetType** di una coda alias aperta.
  - Una modifica agli attributi della coda **RemoteQName** o **RemoteQMgrName** , per qualsiasi handle aperto per questa coda o per una coda che si risolve tramite questa definizione come alias del gestore code.
  - Qualsiasi modifica che fa sì che un handle attualmente aperto per una coda remota si risolva in una coda di trasmissione differente o che non si risolva in una coda. Ad esempio, può includere:

- Una modifica all'attributo **XmitQName** della definizione locale di una coda remota, se la definizione viene utilizzata per una coda o per un alias del gestore code.
- z/OS Solo su z/OS , una modifica al valore dell'attributo del gestore code **IntraGroupQueuing** o una modifica nella definizione della coda di trasmissione condivisa (SYSTEM.QSG.TRANSMIT.QUEUE) utilizzato da IGQ Agent.

C'è un'eccezione: la creazione di una nuova coda di trasmissione. Un handle che si sarebbe risolto in questa coda se fosse stato presente quando l'handle è stato aperto, ma invece risolto nella coda di trasmissione predefinita, non viene reso non valido.

– Una modifica all'attributo del gestore code **DefXmitQName** . In questo caso, tutti gli handle aperti che si sono risolti nella coda precedentemente denominata (che si sono risolti solo perché era la coda di trasmissione predefinita) sono contrassegnati come non validi. Gli handle risolti in questa coda per altri motivi non vengono influenzati.

- L'attributo della coda **Shareability** , se esistono due o più handle che attualmente forniscono l'accesso MQOO_INPUT_SHARED per questa coda o per una coda che si risolve in questa coda. In tal caso, tutti gli handle aperti per questa coda o per una coda che si risolve in questa coda vengono contrassegnati come non validi, indipendentemente dalle opzioni di apertura.

  z/OS Su z/OS, gli handle precedentemente descritti sono contrassegnati come non validi se uno o più handle stanno attualmente fornendo l'accesso MQOO_INPUT_SHARED o MQOO_INPUT_EXCLUSIVE alla coda.
- L'attributo della coda **Usage** , per tutti gli handle aperti per questa coda o per una coda che si risolve in questa coda, indipendentemente dalle opzioni di apertura.

Quando un handle è contrassegnato come non valido, tutte le chiamate successive (diverse da MQCLOSE) che utilizzano questo handle hanno esito negativo con codice motivo MQRC_OBJECT_CHANGED. L'applicazione deve emettere una chiamata MQCLOSE (utilizzando l'handle originale) e riaprire la coda. Qualsiasi aggiornamento non sottoposto a commit rispetto al vecchio handle delle precedenti chiamate riuscite può essere ancora sottoposto a commit o a backout, come richiesto dalla logica dell'applicazione.

Se la modifica di un attributo causa questo, utilizzare una versione forzata speciale della chiamata.

Richiamo C

```
MQOPEN (Hconn, &ObjDesc, Options, &Hobj, &CompCode,
       &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;     /* Connection handle */
MQOD     ObjDesc;   /* Object descriptor */
MQLONG   Options;   /* Options that control the action of MQOPEN */
MQHOBJ   Hobj;      /* Object handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQOPEN' USING HCONN, OBJDESC, OPTIONS, HOBJ, COMPCODE, REASON
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Object descriptor
 01  OBJDESC.
     COPY CMQODV.
**   Options that control the action of MQOPEN
```

```
 01  OPTIONS   PIC S9(9) BINARY.
**   Object handle
 01  HOBJ      PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQOPEN (Hconn, ObjDesc, Options, Hobj, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl ObjDesc   like MQOD;      /* Object descriptor */
dcl Options   fixed bin(31);  /* Options that control the action of
                                 MQOPEN */
dcl Hobj      fixed bin(31);  /* Object handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQOPEN,(HCONN,OBJDESC,OPTIONS,HOBJ,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN     DS      F  Connection handle
OBJDESC   CMQODA  ,  Object descriptor
OPTIONS   DS      F  Options that control the action of MQOPEN
HOBJ      DS      F  Object handle
COMPCODE  DS      F  Completion code
REASON    DS      F  Reason code qualifying COMPCODE
```

Richiamo Visual Basic

Windows

```
MQOPEN Hconn, ObjDesc, Options, Hobj, CompCode, Reason
```

Dichiarare i parametri come segue:

```
Dim Hconn    As Long 'Connection handle'
Dim ObjDesc  As MQOD 'Object descriptor'
Dim Options  As Long 'Options that control the action of MQOPEN'
Dim Hobj     As Long 'Object handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQPUT - Inserisci messaggio

La chiamata MQPUT inserisce un messaggio in una coda o in un elenco di distribuzione o in un argomento. La coda, l'elenco di distribuzione o l'argomento devono essere già aperti.

Sintassi

MQPUT (*Hconn*, *Hobj*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di Hconn è stato restituito da una chiamata MQCONN o MQCONNX precedente.

In applicazioni z/OS per CICS , la chiamata MQCONN può essere omessa e il seguente valore specificato per *Hconn*:

DEF_MQH_HCONN
Handle di connessione predefinito.

HOBJ
Tipo: MQHOBJ - input

Questo handle rappresenta la coda a cui viene aggiunto il messaggio o l'argomento in cui viene pubblicato il messaggio. Il valore di Hobj è stato restituito da una precedente chiamata MQOPEN che specificava l'opzione MQOO_OUTPUT.

MsgDesc
Tipo: MQMD - input/output

Questa struttura descrive gli attributi del messaggio che si sta inviando e riceve informazioni sul messaggio una volta completata la richiesta di inserimento. Consultare “MQMD - Descrittore messaggi” a pagina 431 per i dettagli.

Se l'applicazione fornisce un MQMD version-1 , ai dati del messaggio può essere anteposto un prefisso con una struttura MQMDE per specificare i valori per i campi esistenti in MQMD version-2 ma non in version-1. Il campo *Formato* in MQMD deve essere impostato su MQFMT_MD_EXTENSION per indicare che è presente un MQMDE. Consultare “MQMDE - Estensione descrittore messaggio” a pagina 484 per maggiori dettagli.

L'applicazione non deve fornire una struttura MQMD se viene fornito un handle del messaggio valido nei campi OriginalMsgHandle o NewMsgHandle della struttura MQPMO. Se non viene fornito nulla in uno di questi campi, il descrittore del messaggio viene preso dal descrittore associato agli handle del messaggio.

Se si utilizzano o si pianifica di utilizzare le uscite API, si consiglia di fornire esplicitamente una struttura MQMD e di non utilizzare i descrittori del messaggio associati agli handle del messaggio. Ciò è dovuto al fatto che l'uscita API associata alla chiamata MQPUT o MQPUT1 non è in grado di accertare quali valori MQMD vengono utilizzati dal gestore code per completare la richiesta MQPUT o MQPUT1 .

Opzioni PutMsg
Tipo: MQPMO - input/output

Vedi “MQPMO - Opzioni inserimento messaggio” a pagina 512 per i dettagli.

BufferLength
Tipo: MQLONG - input

La lunghezza del messaggio in Buffer. Zero è valido e indica che il messaggio non contiene dati dell'applicazione. Il limite superiore per BufferLength dipende da diversi fattori:

- Se la destinazione è una coda locale o si risolve in una coda locale, il limite superiore dipende dal fatto che:
  - Il gestore code locale supporta la segmentazione.
  - L'applicazione mittente specifica l'indicatore che consente al gestore code di segmentare il messaggio. Questo indicatore è MQMF_SEGMENTATION_ALLOWED e può essere specificato in un MQMD version-2 o in un MQMDE utilizzato con un MQMD version-1 .

  Se vengono soddisfatte entrambe le condizioni, BufferLength non può superare 999 999 999 meno il valore del campo Offset in MQMD. Il messaggio logico più lungo che può essere inserito

è quindi 999 999 999 byte (quando Offset è zero). Tuttavia, i vincoli delle risorse imposti dal sistema operativo o dall'ambiente in cui l'applicazione è in esecuzione potrebbero determinare un limite inferiore.

Se una o entrambe le condizioni precedenti non sono soddisfatte, BufferLength non può superare l'attributo **MaxMsgLength** della coda e l'attributo **MaxMsgLength** del gestore code.

- Se la destinazione è una coda remota o si risolve in una coda remota, si applicano le condizioni per le code locali, ma a ogni gestore code attraverso il quale il messaggio deve passare per raggiungere la coda di destinazione; in particolare:

  1. La coda di trasmissione locale utilizzata per memorizzare temporaneamente il messaggio nel gestore code locale
  2. Code di trasmissione intermedie (se presenti) utilizzate per memorizzare il messaggio nei gestori code sull'instradamento tra i gestori code locali e di destinazione
  3. La coda di destinazione sul gestore code di destinazione

  Il messaggio più lungo che può essere inserito è quindi regolato dal più restrittivo di queste code e gestori code.

  Quando un messaggio si trova su una coda di trasmissione, ulteriori informazioni si trovano con i dati del messaggio e ciò riduce la quantità di dati dell'applicazione che possono essere trasmessi. In questa situazione, sottrarre byte MQ_MSG_HEADER_LENGTH dai valori MaxMsgLength delle code di trasmissione quando si determina il limite per BufferLength.

  Nota: Quando il messaggio viene inserito, è possibile diagnosticare in modo sincrono solo gli errori di conformità alla condizione 1 (con codice di errore MQRC_MSG_TOO_BIG_FOR_Q_MGR o MQRC_MSG_BIG_FOR_FOR_Q). Se le condizioni 2 o 3 non vengono soddisfatte, il messaggio viene reindirizzato a una coda di messaggi non recapitabili (messaggi non recapitabili), su un gestore code intermedio o sul gestore code di destinazione. In questo caso, viene generato un messaggio di report se richiesto dal mittente.

Memorizza nel buffer
Tipo: MQBYTExBufferLength - input

Si tratta di un buffer contenente i dati dell'applicazione da inviare. Il buffer deve essere allineato su un limite appropriato alla natura dei dati nel messaggio. L'allineamento a 4 - byte è adatto per la maggior parte dei messaggi (inclusi i messaggi contenenti strutture di intestazione IBM MQ), ma alcuni messaggi potrebbero richiedere un allineamento più rigoroso. Ad esempio, un messaggio contenente un numero intero binario a 64 bit potrebbe richiedere un allineamento a 8 byte.

Se Buffer contiene dati numerici o di caratteri, impostare i campi CodedCharSetId e Encoding nel parametro **MsgDesc** sui valori appropriati per i dati; ciò consente al destinatario del messaggio di convertire i dati (se necessario) nella serie di caratteri e nella codifica utilizzati dal destinatario.

Nota: Tutti gli altri parametri sulla chiamata MQPUT devono essere nella serie di caratteri e nella codifica del gestore code locale (forniti dall'attributo del gestore code **CodedCharSetId** e MQENC_NATIVE).

Nel linguaggio di programmazione C, il parametro viene dichiarato come un puntatore a void; l'indirizzo di qualsiasi tipo di dati può essere specificato come parametro.

Se il parametro **BufferLength** è zero, Buffer non viene indicato; in questo caso, l'indirizzo del parametro passato dai programmi scritti nell'assembler C o System/390 può essere null.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Il codice di errore che qualifica CompCode.

Se CompCode è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se CompCode è MQCC_WARNING:

GRUPPO_INCOMPLE_MQRC
(2241, X'8C1') Gruppo di messaggi non completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Messaggio logico non completo.

MQRC_INCONSIST_PERSISTENZA
(2185, X'889 ') Specifica di persistenza incongruente.

UOW MQRC_INCONSISTENT_
(2245, X'8C5') Specifica dell'unità di lavoro non congruente.

MQRC_MULTIPLE_MOTIVI
(2136, X'858 ') Sono stati restituiti più codici di errore.

MQRC_PRIORITY_EXCEEDS_XX_ENCODE_CASE_ONE massimo
(2049, X'801 ') La priorità del messaggio supera il valore massimo supportato.

MQRC_UNKNOWN_REPORT_OPZIONE
(2104, X'838 ') Le opzioni del prospetto nel descrittore del messaggio non sono riconosciute.

Se CompCode è MQCC_FAILED:

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'89C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

MQRC_ALIAS_TARGTYPE_CHANGED
(2480, X'09B0') Il tipo di destinazione della sottoscrizione è stato modificato da coda a argomento.

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita.

ERRORE USCITA MQRC_API
(2183, X'887 ') Impossibile caricare l'uscita API.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

MQRC_BACK_OUT
(2003, X'7D3') Unità di lavoro ripristinata.

ERRORE MQRC_BUFFER_
(2004, X'7D4') Parametro del buffer non valido.

ERRORE MQRC_BUFFER_LENGTH
(2005, X'7D5') Parametro di lunghezza del buffer non valido.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

MQRC_CALL_INTERROTTO
(2549, X'9F5') MQPUT o MQCMIT è stato interrotto e l'elaborazione della riconnessione non può ristabilire un risultato definito.

MQRC_CF_NOT_AVAILABLE
(2345, X' 929 ') La funzione di accoppiamento non è disponibile.

MQRC_CF_STRUC_NON RIUSCITO
(2373, X' 945 ') La struttura della funzione di accoppiamento non è riuscita.

MQRC_CF_STRUC_IN_USO
(2346, X'92A') Struttura CF in uso.

ERRORE MQRC_CFGR
(2416, X' 970 ') La struttura del gruppo PCF MQCFGR nei dati del messaggio non è valida.

ERRORE MQRC_CFH
(2235, X'8BB') Struttura intestazione PCF non valida.

ERRORE MQRC_CFIF
(2414, X'96E') La struttura del parametro del filtro numero intero PCF nei dati del messaggio non è valida.

ERRORE FILTRO MQRC
(2236, X'8BC') Struttura parametro elenco di numeri interi PCF o struttura parametro elenco di numeri interi PCIF*64 non valida.

ERRORE MQRC_CFIN
(2237, X'8BD') Struttura del parametro numero intero PCF o struttura del parametro numero intero PCIF*64 non valida.

ERRORE MQRC_CFF
(2415, X'96F') La struttura del parametro del filtro stringa PCF nei dati del messaggio non è valida.

ERRORE MQRC_CFSL
(2238, X'8BE') Struttura parametro elenco stringhe PCF non valida.

ERRORE MQRC_CFST
(2239, X'8BF') Struttura parametro stringa PCF non valida.

MQRC_CICS_WAIT_NON RIUSCITO
(2140, X'85C') Richiesta di attesa rifiutata da CICS.

ERRORE DI USCITA MQRC_CLUSTER_
(2266, X'8DA') Uscita carico di lavoro cluster non riuscita.

MQRC_CLUSTER_RESOLUTION_ERRORE
(2189, X'88D') Risoluzione del nome cluster non riuscita.

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Errore di risorsa cluster.

MQRC_COD_NOT_VALID_FOR_XCF_Q
(2106, X'83A') Opzione report COD non valida per la coda XCF.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Non autorizzato per la connessione.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Connessione in fase di sospensione.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Chiusura della connessione.

ERRORE CONTENUTO MQRC
2554 (X'09FA') Non è stato possibile analizzare il contenuto del messaggio per stabilire se il messaggio deve essere consegnato a un sottoscrittore con un selettore di messaggi esteso.

ERRORE MQRC_CONTEXT_HANDLE_
(2097, X'831 ') L'handle di coda a cui si fa riferimento non salva il contesto.

MQRC_CONTEXT_NOT_AVAILABLE
(2098, X'832 ') Contesto non disponibile per la gestione code a cui si fa riferimento.

ERRORE MQRC_DATA_LENGTH
(2010, X'7DA') Parametro di lunghezza dati non valido.

ERRORE MQRC_DH_
(2135, X'857 ') Struttura intestazione di distribuzione non valida.

ERRORE MQRC_DLH
(2141, X'85D') Struttura intestazione lettera non instradabile non valida.

ERRORE MQRC_EPH
(2420, X' 974 ') Struttura PCF incorporata non valida.

ERRORE DI MQRC_EXPIRY_
(2013, X'7DD') Scadenza non valida.

ERRORE MQRC_FEEDBACK
(2014, X'7DE') Codice feedback non valido.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Unità globali di conflitto di lavoro.

ERRORE MQRC_GROUP_ID_
(2258, X'8D2') Identificativo gruppo non valido.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X' 931 ') Handle in uso per l'unità di lavoro globale.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

ERRORE MQRC_HEADER_
(2142, X'85E') Struttura intestazione MQ non valida.

ERRORE MQRC_HOBJ_R
(2019, X'7E3') Handle oggetto non valido.

ERRORE MQRC_IIH
(2148, X'864 ') IMS struttura intestazione informazioni non valida.

GRUPPO_INCOMPLE_MQRC
(2241, X'8C1') Gruppo di messaggi non completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Messaggio logico non completo.

MQRC_INCONSIST_PERSISTENZA
(2185, X'889 ') Specifica di persistenza incongruente.

UOW MQRC_INCONSISTENT_
(2245, X'8C5') Specifica dell'unità di lavoro non congruente.

MQRC_LOCAL_UOW_CONFLICT
(2352, X' 930 ') L'unità di lavoro globale è in conflitto con l'unità di lavoro locale.

ERRORE MQRC_MD
(2026, X'7EA') Descrittore messaggio non valido.

ERRORE MQRC_MDE
(2248, X'8C8') Estensione descrittore messaggio non valida.

MQRC_MISSING_REPLY_TO_Q
(2027, X'7EB') Manca la coda di risposta o è stato utilizzato MQPMO_SUPPRESS_REPLYTO

MQRC_MISSING_WIH
(2332, X'91C') I dati del messaggio non iniziano con MQWIH.

ERRORE MQRC_MSG_FLAGS_
(2249, X'8C9') Indicatori messaggio non validi.

ERRORE MQRC_MSG_SEQ_NUMBER_
(2250, X'8CA') Numero di sequenza messaggio non valido.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') La lunghezza del messaggio è maggiore del massimo consentito per la coda.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') La lunghezza del messaggio è maggiore del massimo consentito per il gestore code.

ERRORE MQRC_MSG_TYPE_
(2029, X'7ED') Tipo di messaggio nel descrizione del messaggio non valido.

MQRC_MULTIPLE_MOTIVI
(2136, X'858 ') Sono stati restituiti più codici di errore.

MQRC_NO_DESTINATIONS_AVAILABLE
(2270, X'8DE') Nessuna coda di destinazione disponibile.

MQRC_NOT_OPEN_FOR_OUTPUT
(2039, X'7F7') Coda non aperta per l'emissione.

MQRC_NOT_OPEN_FOR_PASS_ALL
(2093, X'82D') Coda non aperta per passare tutto il contesto.

MQRC_NOT_OPEN_FOR_PASS_IDENT
(2094, X'82E') Coda non aperta per il contesto di identità del passaggio.

MQRC_NOT_OPEN_FOR_SET_ALL
(2095, X'82F') Coda non aperta per impostare tutto il contesto.

MQRC_NOT_OPEN_FOR_SET_IDENT
(2096, X'830 ') Coda non aperta per il contesto di identità impostato.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Definizione oggetto modificata dall'apertura.

MQRC_OBJECT_DAMAGED
(2101, X'835 ') Oggetto danneggiato.

ERRORE MQRC_OFFSET_
(2251, X'8CB') Scostamento segmento messaggio non valido.

MQRC_OPEN_NON RIUSCITO
(2137, X'859 ') Oggetto non aperto correttamente.

ERRORE MQRC_OPTIONS_
(2046, X'7FE') Opzioni non valide o non congruenti.

MQRC_ORIGINAL_LENGTH_ERRORE
(2252, X'8CC') Lunghezza originale non valida.

ERRORE MQRC_PAGESET_
(2193, X'891 ') Errore durante l'accesso al dataset della serie di pagine.

MQRC_PAGESET_FULL
(2192, X'890 ') Il supporto di memoria esterna è pieno.

ERRORE MQRC_PCF
(2149, X'865 ') Strutture PCF non valide.

ERRORE MQRC_PERSISTENCE
(2047, X'7FF') Persistenza non valida.

MQRC_PERSIST_NOT_ALLOWED
(2048, X'800 ') La coda non supporta i messaggi persistenti.

ERRORE PMO_MQRC
(2173, X'87D') Struttura delle opzioni Put - message non valida.

ERRORE MQRC_PMO_RECORD_FLAGS_ERROR
(2158, X'86E') Indicatori del record del messaggio Put non validi.

ERRORE MQRC_PRIORITY_ERROR
(2050, X'802 ') Priorità messaggio non valida.

MQRC_PUBLICATION_FAILURE
(2502, X'9C6') La pubblicazione non è stata consegnata a nessuno dei sottoscrittori.

MQRC_PUT_INIBITO
(2051, X'803 ') Chiamate Put inibite per la coda, per la coda in cui si risolve questa coda o per l'argomento.

ERRORE MQRC_PUT_MSG_RECORDS_
(2159, X'86F') Inserisci record messaggio non validi.

MQRC_PUT_NON_CONSERVATO
(2479, X'09AF') Non è stato possibile conservare la pubblicazione

MQRC_Q_XX_ENCODE_CASE_ONE eliminato
(2052, X'804 ') La coda è stata eliminata.

MQRC_Q_FULL
(2053, X'805 ') La coda contiene già il numero massimo di messaggi.

ERRORE MQRC_Q_MGR_NAME_
(2058, X'80A') Nome gestore code non valido o sconosciuto.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Gestore code non disponibile per la connessione.

MQRC_Q_MGR_QUIESCING
(2161, X'871 ') Gestore code in fase di sospensione.

MQRC_Q_MGR_STOPPING
(2162, X'872 ') Chiusura del gestore code.

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808 ') Nessuno spazio disponibile sul disco per la coda.

MQRC_RECONNECT_NON RIUSCITO
(2548, X'9F4') Dopo la riconnessione, si è verificato un errore durante la reintegrazione delle maniglie per una connessione ricollegabile.

ERRORE MQRC_RECS_PRESENT_
(2154, X'86A') Numero di record presenti non valido.

ERRORE MQRC_REPORT_OPTIONS_
(2061, X'80D') Le opzioni del prospetto nel descrizione del messaggio non sono valide.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

ERRORE DI RISPOSTA MQR_RESPONSE_RECORDS
(2156, X'86C') Record di risposta non validi.

ERRORE MQRC_RFH
(2334, X'91E') Struttura MQRFH o MQRFH2 non valida.

ERRORE RMH_MQRC
(2220, X'8AC') Struttura intestazione messaggio di riferimento non valida.

MQRC_SEGMENT_LENGTH_ZERO
(2253, X'8CD') La lunghezza dei dati nel segmento del messaggio è zero.

MQRC_SEGMENTS_NOT_SUPPORTED
(2365, X'93D') Segmenti non supportati.

SELEZIONE_MQRC_NON_DISPONIBILE
2551 (X'09F7') Esiste un possibile sottoscrittore per la pubblicazione, ma il gestore code non può verificare se inviare la pubblicazione al sottoscrittore.

MQRC_STOPPED_BY_CLUSTER_EXIT
(2188, X'88C') Chiamata rifiutata dall'uscita del carico di lavoro del cluster.

ERRORE MQRC_STORAGE_CLASS_
(2105, X'839 ') Errore della classe di memoria.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890 ') Il supporto di memoria esterna è pieno.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Chiamata eliminata dal programma di uscita.

MQRC_SYNCPOINT_LIMITE_RAGGIUNTO
(2024, X'7E8') Non è possibile gestire ulteriori messaggi all'interno dell'unità di lavoro corrente.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818 ') Supporto punto di sincronizzazione non disponibile.

ERRORE TM_MQRC
(2265, X'8D9') Struttura del messaggio trigger non valida.

ERRORE MQRC_TMC_
(2191, X'88F') Struttura del messaggio trigger di caratteri non valida.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

ERRORE MQRC_UOW_ENLISTMENT_ERROR
(2354, X' 932 ') L'inserimento nell'unità di lavoro globale non è riuscito.

MQRC_UOW_MIX_NON_SUPPORTATO
(2355, X' 933 ') La miscelazione delle chiamate UOW non è supportata.

MQRC_UOW_NON_DISPONIBILE
(2255, X'8CF') Unità di lavoro non disponibile per il gestore code.

ERRORE MQRC_WIH
(2333, X'91D') Struttura MQWIH non valida.

MQRC_WRONG_MD_VERSIONE
(2257, X'8D1') Versione non corretta di MQMD fornita.

ERRORE MQRC_XQH
(2260, X'8D4') Struttura intestazione coda trasmissione non valida.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note sull'utilizzo dell'argomento

1. Le seguenti note si applicano all'uso degli argomenti:

   a. Quando si utilizza MQPUT per pubblicare i messaggi su un argomento, dove uno o più sottoscrittori a tale argomento non possono ricevere la pubblicazione a causa di un problema con la coda del sottoscrittore (ad esempio, è piena), il codice motivo restituito alla chiamata MQPUT e il comportamento di consegna dipende dall'impostazione degli attributi PMSGDLV o NPMSGDLV sull'TOPIC. Si noti che la consegna di una pubblicazione alla coda di messaggi non recapitabili quando viene specificato MQRO_DEAD_LETTER_Q o l'eliminazione del messaggio quando viene specificato MQRO_DISCARD_MSG viene considerata come una consegna corretta del messaggio. Se nessuna delle pubblicazioni viene consegnata, MQPUT restituisce MQRC_PUBLICATION_FAILURE. Ciò può verificarsi nei seguenti casi:

      - Un messaggio viene pubblicato in un TOPIC con PMSGDLV o NPMSGDLV (a seconda della persistenza del messaggio) impostato su ALL e qualsiasi sottoscrizione (durevole o meno) ha una coda che non può ricevere la pubblicazione.
      - Un messaggio viene pubblicato in un TOPIC con PMSGDLV o NPMSGDLV (in base alla persistenza del messaggio) impostato su ALLDUR e una sottoscrizione durevole ha una coda che non può ricevere la pubblicazione.

      MQPUT può restituire MQRC_NONE anche se le pubblicazioni non possono essere consegnate ad alcuni sottoscrittori nei seguenti casi:

      - Un messaggio viene pubblicato in un TOPIC con PMSGDLV o NPMSGDLV (a seconda della persistenza del messaggio) impostato su ALLAVAIL e qualsiasi sottoscrizione, durevole o meno, ha una coda che non può ricevere la pubblicazione.

- Un messaggio viene pubblicato in un TOPIC con PMSGDLV o NPMSGDLV (in base alla persistenza del messaggio) impostato su ALLDUR e una sottoscrizione non durevole ha una coda che non può ricevere la pubblicazione.

È possibile utilizzare l'attributo dell'argomento USEDLQ per stabilire se la coda di messaggi non recapitabili viene utilizzata quando i messaggi di pubblicazione non possono essere consegnati alla coda del sottoscrittore corretta. Per ulteriori informazioni sull'utilizzo di USEDLQ, consultare DEFINE TOPIC.

b. Se non sono presenti sottoscrittori per l'argomento utilizzato, il messaggio pubblicato non viene inviato ad alcuna coda e viene eliminato. Non importa se il messaggio è persistente o non persistente, o se ha una scadenza illimitata o ha un tempo di scadenza, viene ancora eliminato se non ci sono sottoscrittori. L'eccezione è se il messaggio deve essere conservato, nel qual caso, sebbene non venga inviato ad alcuna coda di sottoscrittori, viene memorizzato rispetto all'argomento per essere consegnato a qualsiasi nuova sottoscrizione o a qualsiasi sottoscrittore che richieda le pubblicazioni conservate utilizzando MQSUBRQ.

MQPUT e MQPUT1

È possibile utilizzare entrambe le chiamate MQPUT e MQPUT1 per inserire i messaggi su una coda; la chiamata da utilizzare dipende dalle circostanze

- Utilizzare la chiamata MQPUT per inserire più messaggi sulla stessa coda.

  Viene emessa prima una chiamata MQOPEN che specifica l'opzione MQOO_OUTPUT, seguita da una o più richieste MQPUT per aggiungere messaggi alla coda; infine, la coda viene chiusa con una chiamata MQCLOSE. Ciò fornisce prestazioni migliori rispetto all'utilizzo ripetuto della chiamata di MQPUT1 .
- Utilizzare la chiamata MQPUT1 per inserire solo un messaggio su una coda.

  Questa chiamata incapsula le chiamate MQOPEN, MQPUT e MQCLOSE in una singola chiamata, riducendo il numero di chiamate che devono essere emesse.

Code di destinazione

Le seguenti note si applicano all'utilizzo delle code di destinazione:

1. Se un'applicazione inserisce una sequenza di messaggi nella stessa coda senza utilizzare i gruppi di messaggi, l'ordine di tali messaggi viene conservato se sono soddisfatte le condizioni dettagliate. Alcune condizioni si applicano alle code di destinazione locali e remote; altre condizioni si applicano solo alle code di destinazione remote.

   Condizioni applicabili alle code di destinazione locali e remote

   - Tutte le chiamate MQPUT si trovano all'interno della stessa unità di lavoro o nessuna di esse si trova all'interno di un'unità di lavoro.

     Tenere presente che quando i messaggi vengono inseriti in una particolare coda all'interno di una singola unità di lavoro, i messaggi provenienti da altre applicazioni potrebbero essere intervallati dalla sequenza di messaggi nella coda.
   - Tutte le chiamate MQPUT vengono effettuate utilizzando lo stesso handle di oggetto *Hobj*.

     In alcuni ambienti, la sequenza dei messaggi viene conservata anche quando vengono utilizzati diversi handle di oggetto, se le chiamate vengono effettuate dalla stessa applicazione. Il significato di *stessa applicazione* è determinato dall'ambiente:

     – z/OS Su z/OS, l'applicazione è:

       - Per CICS, l'attività CICS
       - Per IMS, l'attività
       - Per il batch z/OS , l'attività

     – IBM i Su IBM i, l'applicazione è il lavoro.

– ALW Su AIX, Linux, and Windows, l'applicazione è il thread.

- Tutti i messaggi hanno la stessa priorità.
- I messaggi non vengono inseriti in una coda del cluster con MQOO_BIND_NOT_FIXED specificato (o con MQOO_BIND_AS_Q_DEF attivo quando l'attributo della coda DefBind ha il valore MQBND_BIND_NOT_FIXED).

Ulteriori condizioni che si applicano alle code di destinazione remote

- Esiste un solo percorso dal gestore code di invio al gestore code di destinazione.

  Se alcuni messaggi nella sequenza potrebbero trovarsi in un percorso differente (ad esempio, a causa della riconfigurazione, del bilanciamento del traffico o della selezione del percorso in base alla dimensione del messaggio), non è possibile garantire l'ordine dei messaggi nel gestore code di destinazione.
- I messaggi non vengono inseriti temporaneamente nelle code di messaggi non recapitabili nei gestori code di invio, intermedi o di destinazione.

  Se uno o più messaggi vengono inseriti temporaneamente in una coda di messaggi non recapitabili (ad esempio, perché una coda di trasmissione o la coda di destinazione è temporaneamente piena), i messaggi possono arrivare sulla coda di destinazione fuori sequenza.
- I messaggi sono tutti persistenti o non persistenti.

  Se un canale sulla rotta tra i gestori code di invio e di destinazione ha il suo attributo **NonPersistentMsgSpeed** impostato su MQNPMS_FAST, i messaggi non persistenti possono saltare in avanti rispetto ai messaggi persistenti, determinando l'ordine dei messaggi persistenti relativi ai messaggi non persistenti non conservati. Tuttavia, l'ordine dei messaggi persistenti relativi l'uno all'altro e dei messaggi non persistenti relativi l'uno all'altro, viene conservato.

Se queste condizioni non vengono soddisfatte, è possibile utilizzare i gruppi di messaggi per conservare l'ordine dei messaggi, ma ciò richiede che le applicazioni di invio e di ricezione utilizzino il supporto di raggruppamento dei messaggi. Per ulteriori informazioni sui gruppi di messaggi, consultare:

- Campo MQMD - MsgFlags
- MQPMO_LOGICAL_ORDER
- MQGMO_LOGICAL_ORDER

Elenchi di distribuzione

Le seguenti note si applicano all'uso degli elenchi di distribuzione.

Gli elenchi di distribuzione sono supportati nei seguenti ambienti:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Windows Windows

e per IBM MQ MQI clients collegati a questi sistemi.

1. È possibile inserire i messaggi in un elenco di distribuzione utilizzando version-1 o version-2 MQPMO. Se si utilizza un MQPM version-1 (o un MQPMO version-2 con `RecsPresent` uguale a 0), l'applicazione non può fornire record di messaggi di inserimento o record di risposta. Non è possibile identificare le code che rilevano errori se il messaggio viene inviato correttamente ad alcune code nell'elenco di distribuzione e non ad altre.

   Se l'applicazione fornisce record di messaggi di inserimento o record di risposta, impostare il campo `Version` su MQPMO_VERSION_2.

È anche possibile utilizzare un MQPMO version-2 per inviare messaggi a una coda singola che non si trova in un elenco di distribuzione, verificando che `RecsPresent` sia zero.

2. I parametri del codice di completamento e del codice di errore sono impostati come segue:

   - Se gli inserimenti nelle code nell'elenco di distribuzione hanno esito positivo o negativo nello stesso modo, i parametri del codice di completamento e del codice motivo vengono impostati per descrivere il risultato comune. I record di risposta MQRR (se forniti dall'applicazione) non sono impostati in questo caso.

     Ad esempio, se ogni operazione di inserimento ha esito positivo, il codice di completamento e il codice motivo sono impostati su MQCC_OK e MQRC_NONE; se ogni operazione ha esito negativo perché tutte le code non sono abilitate per le operazioni di inserimento, i parametri sono impostati su MQCC_FAILED e MQRC_PUT_INIITED.

   - Se gli inserimenti nelle code nell'elenco di distribuzione non hanno esito positivo o negativo nello stesso modo:

     – Il parametro del codice di completamento è impostato su MQCC_WARNING se almeno un inserimento ha avuto esito positivo e su MQCC_FAILED se tutti hanno avuto esito negativo.

     – Il parametro del codice di errore è impostato su MQRC_MULTIPLE_REASON.

     – I record di risposta (se forniti dall'applicazione) sono impostati sui codici di completamento individuali e sui codici motivo per le code nell'elenco di distribuzione.

     Se l'operazione di inserimento in una destinazione non riesce perché l'apertura per tale destinazione non è riuscita, i campi nel record di risposta sono impostati su MQCC_FAILED e MQRC_OPEN_FAILED; tale destinazione è inclusa in `InvalidDestCount`.

3. Se una destinazione nell'elenco di distribuzione si risolve in una coda locale, il messaggio viene posizionato su tale coda in formato normale (cioè, non come un messaggio dell'elenco di distribuzione). Se più di una destinazione si risolve nella stessa coda locale, viene inserito un messaggio nella coda per ciascuna di tali destinazioni.

   Se una destinazione nell'elenco di distribuzione si risolve in una coda remota, un messaggio viene inserito nella coda di trasmissione appropriata. Quando più destinazioni si risolvono nella stessa coda di trasmissione, è possibile inserire nella coda di trasmissione un singolo messaggio dell'elenco di distribuzione contenente tali destinazioni, anche se tali destinazioni non erano adiacenti nell'elenco di destinazione fornito dall'applicazione. Tuttavia, questa operazione può essere eseguita solo se la coda di trasmissione supporta i messaggi dell'elenco di distribuzione (consultare DistLists).

   Se la coda di trasmissione non supporta gli elenchi di distribuzione, una copia del messaggio in formato normale viene inserita nella coda di trasmissione per ogni destinazione che utilizza tale coda di trasmissione.

   Se un elenco di distribuzione con i dati del messaggio dell'applicazione è troppo grande per una coda di trasmissione, il messaggio dell'elenco di distribuzione viene suddiviso in messaggi dell'elenco di distribuzione più piccoli, ognuno dei quali contiene un numero minore di destinazioni. Se i dati del messaggio dell'applicazione si adattano solo alla coda, i messaggi dell'elenco di distribuzione non possono essere utilizzati e il gestore code genera una copia del messaggio in formato normale per ciascuna destinazione che utilizza tale coda di trasmissione.

   Se destinazioni diverse hanno priorità o persistenza del messaggio differenti (ciò può verificarsi quando l'applicazione specifica MQPRI_PRIORITY_AS_Q_DEF o MQPER_PERSISTENCE_AS_Q_DEF), i messaggi non vengono conservati nello stesso messaggio dell'elenco di distribuzione. Invece, il gestore code genera tutti i messaggi dell'elenco di distribuzione necessari per soddisfare i diversi valori di priorità e persistenza.

4. Un inserimento in un elenco di distribuzione può risultare in:

   - Un singolo messaggio dell'elenco di distribuzione oppure
   - Un numero di messaggi di elenco di distribuzione più piccoli, oppure
   - Una combinazione di messaggi dell'elenco di distribuzione e messaggi normali oppure
   - Solo messaggi normali.

Quale di questi si verifica dipende dal fatto che:

- Le destinazioni nell'elenco sono locali, remote o miste.
- Le destinazioni hanno la stessa priorità e persistenza del messaggio.
- Le code di trasmissione possono contenere i messaggi dell'elenco di distribuzione.
- La lunghezza massima dei messaggi delle code di trasmissione è sufficiente per contenere il messaggio in formato di elenco di distribuzione.

Tuttavia, indipendentemente da quale di queste ricorre, ogni messaggio *fisico* risultante (ovvero, ogni messaggio normale o messaggio dell'elenco di distribuzione risultante dall'inserimento) conta come solo *un* messaggio quando:

- Verifica se l'applicazione ha superato il numero massimo consentito di messaggi in un'unità di lavoro (consultare l'attributo del gestore code **MaxUncommittedMsgs**).
- Verifica se le condizioni di attivazione sono soddisfatte.
- Incrementando le profondità della coda e verificando se la profondità massima della coda delle code viene superata.

5. Qualsiasi modifica alle definizioni di coda che avrebbe causato la non validità di un handle se le code fossero state aperte singolarmente (ad esempio, una modifica nel percorso di risoluzione), non fa sì che l'handle dell'elenco di distribuzione diventi non valido. Tuttavia, si verifica un errore per quella particolare coda quando l'handle dell'elenco di distribuzione viene utilizzato su una successiva chiamata MQPUT.

Intestazioni

Se un messaggio viene inserito con una o più strutture di intestazione IBM MQ all'inizio dei dati del messaggio dell'applicazione, il gestore code esegue determinati controlli sulle strutture di intestazione per verificarne la validità. Se il gestore code rileva un errore, la chiamata ha esito negativo con un codice motivo appropriato. I controlli effettuati variano in base alle particolari strutture presenti:

- I controlli vengono eseguiti solo se viene utilizzato un MQMD version-2 o successivo sulla chiamata MQPUT o MQPUT1 . I controlli non vengono eseguiti se viene utilizzato un MQMD version-1 , anche se è presente un MQMDE all'inizio dei dati del messaggio.
- Le strutture non supportate dal gestore code locale e le strutture che seguono il primo MQDLH nel messaggio non vengono convalidate.
- Le strutture MQDH e MQMDE vengono convalidate completamente dal gestore code.
- Altre strutture vengono convalidate parzialmente dal gestore code (non tutti i campi sono controllati).

I controlli generali eseguiti dal gestore code includono:

- Il campo `StrucId` deve essere valido.
- Il campo `Version` deve essere valido.
- Il campo `StrucLength` deve specificare un valore sufficientemente grande da includere la struttura più eventuali dati a lunghezza variabile che fanno parte della struttura.
- Il campo `CodedCharSetId` non deve essere zero o un valore negativo non valido (MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR e MQCCSI_UNDEFINED non sono validi nella maggior parte delle strutture di intestazione IBM MQ).
- Il parametro **BufferLength** della chiamata deve indicare un valore sufficientemente grande da includere la struttura (la struttura non deve estendersi oltre la fine del messaggio).

Oltre ai controlli generali sulle strutture, devono essere soddisfatte le condizioni seguenti:

- La somma delle lunghezze delle strutture in un messaggio PCF deve essere uguale alla lunghezza specificata dal parametro **BufferLength** nella chiamata MQPUT o MQPUT1 . Un messaggio PCF è un messaggio con un nome formato MQFMT_ADMIN, MQFMT_EVENT o MQFMT_PCF.
- Una struttura IBM MQ non deve essere troncata, tranne nelle seguenti situazioni in cui sono consentite le strutture troncate:

  - Messaggi che sono messaggi di report.
  - Messaggi PCF.
  - Messaggi contenenti una struttura di MQDLH. (Le strutture che seguono il primo MQDLH possono essere troncate; le strutture che precedono MQDLH non possono.)
- Una struttura IBM MQ non deve essere suddivisa su due o più segmenti; la struttura deve essere contenuta interamente in un segmento.

Memorizza nel buffer

Per il linguaggio di programmazione Visual Basic, si applicano i punti seguenti:

- Se la dimensione del parametro **Buffer** è inferiore alla lunghezza specificata dal parametro **BufferLength** , la chiamata ha esito negativo con codice motivo MQRC_BUFFER_LENGTH_ERROR.
- Il parametro **Buffer** viene dichiarato come di tipo `String`. Se i dati da inserire nella coda non sono di tipo `String`, utilizzare ilChiamata MQPUTAny al posto di MQPUT.

  La chiamata MQPUTAny ha gli stessi parametri della chiamata MQPUT, tranne per il fatto che il parametro **Buffer** è dichiarato come di tipo `Any`, consentendo l'inserimento di qualsiasi tipo di dati nella coda. Tuttavia, ciò significa che `Buffer` non può essere controllato per garantire che abbia una dimensione di almeno `BufferLength` byte.

Richiamo C

```
MQPUT (Hconn, Hobj, &MsgDesc, &PutMsgOpts, BufferLength, Buffer,
      &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;         /* Connection handle */
MQHOBJ   Hobj;          /* Object handle */
MQMD     MsgDesc;       /* Message descriptor */
MQPMO    PutMsgOpts;    /* Options that control the action of MQPUT */
MQLONG   BufferLength;  /* Length of the message in Buffer */
MQBYTE   Buffer[n];     /* Message data */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
     CALL 'MQPUT' USING HCONN, HOBJ, MSGDESC, PUTMSGOPTS, BUFFERLENGTH,
                        BUFFER, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Object handle
 01  HOBJ          PIC S9(9) BINARY.
**   Message descriptor
 01  MSGDESC.
     COPY CMQMDV.
**   Options that control the action of MQPUT
 01  PUTMSGOPTS.
     COPY CMQPMOV.
**   Length of the message in BUFFER
 01  BUFFERLENGTH  PIC S9(9) BINARY.
**   Message data
 01  BUFFER        PIC X(n).
**   Completion code
 01  COMPCODE      PIC S9(9) BINARY.
```

```
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQPUT (Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer,
           CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn         fixed bin(31);  /* Connection handle */
dcl Hobj          fixed bin(31);  /* Object handle */
dcl MsgDesc       like MQMD;      /* Message descriptor */
dcl PutMsgOpts    like MQPMO;     /* Options that control the action of
                                     MQPUT */
dcl BufferLength  fixed bin(31);  /* Length of the message in Buffer */
dcl Buffer        char(n);        /* Message data */
dcl CompCode      fixed bin(31);  /* Completion code */
dcl Reason        fixed bin(31);  /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQPUT,(HCONN,HOBJ,MSGDESC,PUTMSGOPTS,BUFFERLENGTH, X
               BUFFER,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN         DS       F      Connection handle
HOBJ          DS       F      Object handle
MSGDESC       CMQMDA   ,      Message descriptor
PUTMSGOPTS    CMQPMOA  ,      Options that control the action of MQPUT
BUFFERLENGTH  DS       F      Length of the message in BUFFER
BUFFER        DS       CL(n)  Message data
COMPCODE      DS       F      Completion code
REASON        DS       F      Reason code qualifying COMPCODE
```

Richiamo Visual Basic

Windows

```
MQPUT Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer, CompCode,
      Reason
```

Dichiarare i parametri come segue:

```
Dim Hconn        As Long   'Connection handle'
Dim Hobj         As Long   'Object handle'
Dim MsgDesc      As MQMD   'Message descriptor'
Dim PutMsgOpts   As MQPMO  'Options that control the action of MQPUT'
Dim BufferLength As Long   'Length of the message in Buffer'
Dim Buffer       As String 'Message data'
Dim CompCode     As Long   'Completion code'
Dim Reason       As Long   'Reason code qualifying CompCode'
```

MQPUT1 - Inserire un messaggio

La chiamata MQPUT1 inserisce un messaggio in una coda o in un elenco di distribuzione o in un argomento.

Non è necessario che la coda, l'elenco di distribuzione o l'argomento siano aperti.

Sintassi

MQPUT1 (*Hconn*, *ObjDesc*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

z/OS In applicazioni z/OS per CICS , la chiamata MQCONN può essere omessa e il seguente valore specificato per *Hconn*:

DEF_MQH_HCONN
Handle di connessione predefinito.

ObjDesc
Tipo: MQOD - input/output

Si tratta di una struttura che identifica la coda a cui viene aggiunto il messaggio o l'argomento in cui viene pubblicato il messaggio. Consultare "MQOD - Descrittore oggetto" a pagina 492 per i dettagli.

Se la struttura è una coda, l'utente deve essere autorizzato ad aprire la coda per l'emissione. La coda non deve essere una coda modello.

MsgDesc
Tipo: MQMD - input/output

Questa struttura descrive gli attributi del messaggio che si sta inviando e riceve le informazioni di feedback una volta completata la richiesta di inserimento. Consultare "MQMD - Descrittore messaggi" a pagina 431 per i dettagli.

Se l'applicazione fornisce un MQMD version-1 , ai dati del messaggio può essere anteposto un prefisso con una struttura MQMDE per specificare i valori per i campi esistenti in MQMD version-2 ma non in version-1. Impostare il campo `Format` in MQMD su MQFMT_MD_EXTENSION per indicare che è presente un MQMDE. Consultare "MQMDE - Estensione descrittore messaggio" a pagina 484 per maggiori dettagli.

L'applicazione non ha bisogno di fornire una struttura MQMD se viene fornito un handle del messaggio valido nel campo `MsgHandle` della struttura MQGMO o nei campi `OriginalMsgHandle` o `NewMsgHandle` della struttura MQPMO. Se non viene fornito nulla in uno di questi campi, il descrittore del messaggio viene preso dal descrittore associato agli handle del messaggio.

Opzioni PutMsg
Tipo: MQPMO - input/output

Vedi "MQPMO - Opzioni inserimento messaggio" a pagina 512 per i dettagli.

BufferLength
Tipo: MQLONG - input

La lunghezza del messaggio in `Buffer`. Zero è valido e indica che il messaggio non contiene dati dell'applicazione. Il limite superiore dipende da vari fattori; consultare "MQPUT - Inserisci messaggio" a pagina 768 per la descrizione del parametro **`BufferLength`** .

Memorizza nel buffer
Tipo: MQBYTExBufferLength - input

Si tratta di un buffer contenente i dati del messaggio dell'applicazione da inviare. Allineare il buffer su un limite appropriato alla natura dei dati nel messaggio. L'allineamento a 4 - byte è adatto per la maggior parte dei messaggi (inclusi i messaggi contenenti strutture di intestazione IBM MQ), ma alcuni messaggi potrebbero richiedere un allineamento più rigoroso. Ad esempio, un messaggio contenente un numero intero binario a 64 bit potrebbe richiedere un allineamento a 8 byte.

Se Buffer contiene dati numerici o di caratteri, impostare i campi CodedCharSetId e Encoding nel parametro **MsgDesc** sui valori appropriati per i dati; ciò consente al destinatario del messaggio di convertire i dati (se necessario) nella serie di caratteri e nella codifica utilizzati dal destinatario.

Nota: Tutti gli altri parametri sulla chiamata MQPUT1 devono essere nella serie di caratteri e nella codifica del gestore code locale (forniti dall'attributo del gestore code **CodedCharSetId** e MQENC_NATIVE).

Nel linguaggio di programmazione C, il parametro viene dichiarato come un puntatore a void; l'indirizzo di qualsiasi tipo di dati può essere specificato come parametro.

Se il parametro **BufferLength** è zero, Buffer non viene indicato; in questo caso, l'indirizzo del parametro passato dai programmi scritti nell'assembler C o System/390 può essere null.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Il codice di errore che qualifica CompCode.

Se CompCode è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se CompCode è MQCC_WARNING:

MQRC_MULTIPLE_MOTIVI
(2136, X'858 ') Sono stati restituiti più codici di errore.

GRUPPO_INCOMPLE_MQRC
(2241, X'8C1') Gruppo di messaggi non completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Messaggio logico non completo.

MQRC_PRIORITY_EXCEEDS_XX_ENCODE_CASE_ONE massimo
(2049, X'801 ') La priorità del messaggio supera il valore massimo supportato.

MQRC_UNKNOWN_REPORT_OPZIONE
(2104, X'838 ') Opzioni prospetto nel descrittore del messaggio non riconosciute.

Se CompCode è MQCC_FAILED:

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'89C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

ERRORE TIPO_Q_ALIAS_MQRC
(2001, X'7D1') La coda di base dell'alias non è un tipo valido.

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita.

ERRORE USCITA MQRC_API
(2183, X'887 ') Impossibile caricare l'uscita API.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

MQRC_BACK_OUT
(2003, X'7D3') Unità di lavoro ripristinata.

ERRORE MQRC_BUFFER_
(2004, X'7D4') Parametro del buffer non valido.

ERRORE MQRC_BUFFER_LENGTH
(2005, X'7D5') Parametro di lunghezza del buffer non valido.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

MQRC_CF_NOT_AVAILABLE
(2345, X' 929 ') CF (coupling facility) non disponibile.

MQRC_CF_STRUC_AUTH_NON RIUSCITO
(2348, X'92C') Il controllo dell'autorizzazione della struttura CFS non è riuscito.

ERRORE MQRC_CF_STRUC_
(2349, X'92D') Struttura CF non valida.

MQRC_CF_STRUC_NON RIUSCITO
(2373, X' 945 ') La struttura della funzione di accoppiamento non è riuscita.

MQRC_CF_STRUC_IN_USO
(2346, X'92A') Struttura CF in uso.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') L'elenco - intestazione della struttura CFS (Coupling Facility Structure) è in uso.

ERRORE MQRC_CFGR
(2416, X' 970 ') La struttura del gruppo PCF MQCFGR nei dati del messaggio non è valida.

ERRORE MQRC_CFH
(2235, X'8BB') Struttura intestazione PCF non valida.

ERRORE MQRC_CFIF
(2414, X'96E') La struttura del parametro del filtro numero intero PCF nei dati del messaggio non è valida.

ERRORE FILTRO MQRC
(2236, X'8BC') Struttura parametro elenco di numeri interi PCF o struttura parametro elenco di numeri interi PCIF*64 non valida.

ERRORE MQRC_CFIN
(2237, X'8BD') Struttura del parametro numero intero PCF o struttura del parametro numero intero PCIF*64 non valida.

ERRORE MQRC_CFF
(2415, X'96F') La struttura del parametro del filtro stringa PCF nei dati del messaggio non è valida.

ERRORE MQRC_CFSL
(2238, X'8BE') Struttura parametro elenco stringhe PCF non valida.

ERRORE MQRC_CFST
(2239, X'8BF') Struttura parametro stringa PCF non valida.

MQRC_CICS_WAIT_NON RIUSCITO
(2140, X'85C') Richiesta di attesa rifiutata da CICS.

ERRORE DI USCITA MQRC_CLUSTER_
(2266, X'8DA') Uscita carico di lavoro cluster non riuscita.

MQRC_CLUSTER_RESOLUTION_ERRORE
(2189, X'88D') Risoluzione del nome cluster non riuscita.

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Errore di risorsa cluster.

MQRC_COD_NOT_VALID_FOR_XCF_Q
(2106, X'83A') Opzione report COD non valida per la coda XCF.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Non autorizzato per la connessione.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Connessione in fase di sospensione.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Chiusura della connessione.

ERRORE CONTENUTO MQRC
2554 (X'09FA') Impossibile analizzare il contenuto del messaggio per determinare se il messaggio può essere consegnato a un sottoscrittore con un selettore di messaggi esteso.

ERRORE MQRC_CONTEXT_HANDLE_
(2097, X'831 ') L'handle di coda a cui si fa riferimento non salva il contesto.

MQRC_CONTEXT_NOT_AVAILABLE
(2098, X'832 ') Contesto non disponibile per la gestione code a cui si fa riferimento.

ERRORE MQRC_DATA_LENGTH
(2010, X'7DA') Parametro di lunghezza dati non valido.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') Db2 sottosistema non disponibile.

MQRC_DEF_XMIT_Q_TYPE_ERROR
(2198, X'896 ') La coda di trasmissione predefinita non è locale.

MQRC_DEF_XMIT_Q_USAGE_ERROR
(2199, X'897 ') Errore di utilizzo della coda di trasmissione predefinita.

ERRORE MQRC_DH_
(2135, X'857 ') Struttura intestazione di distribuzione non valida.

ERRORE MQRC_DLH
(2141, X'85D') Struttura intestazione lettera non instradabile non valida.

ERRORE MQRC_EPH
(2420, X' 974 ') Struttura PCF incorporata non valida.

ERRORE DI MQRC_EXPIRY_
(2013, X'7DD') Scadenza non valida.

ERRORE MQRC_FEEDBACK
(2014, X'7DE') Codice feedback non valido.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Unità globali di conflitto di lavoro.

ERRORE MQRC_GROUP_ID_
(2258, X'8D2') Identificativo gruppo non valido.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X' 931 ') Handle in uso per l'unità di lavoro globale.

MQRC_HANDLE_NON_DISPONIBILE
(2017, X'7E1') Nessun ulteriore handle disponibile.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

ERRORE MQRC_HEADER_
(2142, X'85E') IBM MQ struttura intestazione non valida.

ERRORE MQRC_IIH
(2148, X'864 ') IMS struttura intestazione informazioni non valida.

MQRC_LOCAL_UOW_CONFLICT
(2352, X' 930 ') L'unità di lavoro globale è in conflitto con l'unità di lavoro locale.

ERRORE MQRC_MD
(2026, X'7EA') Descrittore messaggio non valido.

ERRORE MQRC_MDE
(2248, X'8C8') Estensione descrittore messaggio non valida.

MQRC_MISSING_REPLY_TO_Q
(2027, X'7EB') Coda reply - to mancante.

MQRC_MISSING_WIH
(2332, X'91C') I dati del messaggio non iniziano con MQWIH.

ERRORE MQRC_MSG_FLAGS_
(2249, X'8C9') Indicatori messaggio non validi.

ERRORE MQRC_MSG_SEQ_NUMBER_
(2250, X'8CA') Numero di sequenza messaggio non valido.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') La lunghezza del messaggio è maggiore del massimo consentito per la coda.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') La lunghezza del messaggio è maggiore del massimo consentito per il gestore code.

ERRORE MQRC_MSG_TYPE_
(2029, X'7ED') Tipo di messaggio nel descrizione del messaggio non valido.

MQRC_MULTIPLE_MOTIVI
(2136, X'858 ') Sono stati restituiti più codici di errore.

MQRC_NO_DESTINATIONS_AVAILABLE
(2270, X'8DE') Nessuna coda di destinazione disponibile.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

MQRC_OBJECT_DAMAGED
(2101, X'835 ') Oggetto danneggiato.

MQRC_OBJECT_IN_USE
(2042, X'7FA') Oggetto già aperto con opzioni in conflitto.

MQRC_OBJECT_LEVEL_INCOMPATIBILE
(2360, X' 938 ') Livello oggetto non compatibile.

ERRORE MQRC_OBJECT_NAME_ERROR
(2152, X'868 ') Nome oggetto non valido.

MQRC_OBJECT_NOT_UNIQUE
(2343, X' 927 ') Oggetto non univoco.

MQRC_OBJECT_Q_MGR_NAME_ERROR
(2153, X'869 ') Nome gestore code oggetto non valido.

ERRORE MQRC_OBJECT_RECORDS
(2155, X'86B') Record oggetto non validi.

ERRORE TIPO_OGGETTO_MQRC
(2043, X'7FB') Tipo oggetto non valido.

ERRORE MQRC_O
(2044, X'7FC') Struttura descrittore oggetto non valida.

ERRORE MQRC_OFFSET_
(2251, X'8CB') Scostamento segmento messaggio non valido.

ERRORE MQRC_OPTIONS_
(2046, X'7FE') Opzioni non valide o non congruenti.

MQRC_ORIGINAL_LENGTH_ERRORE
(2252, X'8CC') Lunghezza originale non valida.

ERRORE MQRC_PAGESET_
(2193, X'891 ') Errore durante l'accesso al dataset della serie di pagine.

MQRC_PAGESET_FULL
(2192, X'890 ') Il supporto di memoria esterna è pieno.

ERRORE MQRC_PCF
(2149, X'865 ') Strutture PCF non valide.

ERRORE MQRC_PERSISTENCE
(2047, X'7FF') Persistenza non valida.

MQRC_PERSIST_NOT_ALLOWED
(2048, X'800 ') La coda non supporta i messaggi persistenti.

ERRORE PMO_MQRC
(2173, X'87D') Struttura delle opzioni Put - message non valida.

ERRORE MQRC_PMO_RECORD_FLAGS_ERROR
(2158, X'86E') Indicatori del record del messaggio Put non validi.

ERRORE MQRC_PRIORITY_ERROR
(2050, X'802 ') Priorità messaggio non valida.

MQRC_PUBLICATION_FAILURE
(2502, X'9C6') La pubblicazione non è stata consegnata a nessuno dei sottoscrittori.

MQRC_PUT_INIBITO
(2051, X'803 ') Chiamate Put inibite per la coda.

ERRORE MQRC_PUT_MSG_RECORDS_
(2159, X'86F') Inserisci record messaggio non validi.

MQRC_Q_XX_ENCODE_CASE_ONE eliminato
(2052, X'804 ') La coda è stata eliminata.

MQRC_Q_FULL
(2053, X'805 ') La coda contiene già il numero massimo di messaggi.

ERRORE MQRC_Q_MGR_NAME_
(2058, X'80A') Nome gestore code non valido o sconosciuto.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Gestore code non disponibile per la connessione.

MQRC_Q_MGR_QUIESCING
(2161, X'871 ') Gestore code in fase di sospensione.

MQRC_Q_MGR_STOPPING
(2162, X'872 ') Chiusura del gestore code.

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808 ') Nessuno spazio disponibile sul disco per la coda.

ERRORE MQRC_Q_TYPE_
(2057, X'809 ') Tipo coda non valido.

ERRORE MQRC_RECS_PRESENT_
(2154, X'86A') Numero di record presenti non valido.

MQRC_REMOTE_Q_NAME_ERROR
(2184, X'888 ') Nome coda remota non valido.

ERRORE MQRC_REPORT_OPTIONS_
(2061, X'80D') Le opzioni del prospetto nel descrizione del messaggio non sono valide.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

ERRORE DI RISPOSTA MQR_RESPONSE_RECORDS
(2156, X'86C') Record di risposta non validi.

ERRORE MQRC_RFH
(2334, X'91E') Struttura MQRFH o MQRFH2 non valida.

ERRORE RMH_MQRC
(2220, X'8AC') Struttura intestazione messaggio di riferimento non valida.

ERRORE MQRC_SECURITY_ERROR
(2063, X'80F') Si è verificato un errore di sicurezza.

MQRC_SEGMENT_LENGTH_ZERO
(2253, X'8CD') La lunghezza dei dati nel segmento del messaggio è zero.

SELEZIONE_MQRC_NON_DISPONIBILE
2551 (X'09F7') Esiste un possibile sottoscrittore per la pubblicazione, ma il gestore code non può verificare se inviare la pubblicazione al sottoscrittore.

MQRC_STOPPED_BY_CLUSTER_EXIT
(2188, X'88C') Chiamata rifiutata dall'uscita del carico di lavoro del cluster.

ERRORE MQRC_STORAGE_CLASS_
(2105, X'839 ') Errore della classe di memoria.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890 ') Il supporto di memoria esterna è pieno.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Chiamata eliminata dal programma di uscita.

MQRC_SYNCPOINT_LIMITE_RAGGIUNTO
(2024, X'7E8') Non è possibile gestire ulteriori messaggi all'interno dell'unità di lavoro corrente.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818 ') Supporto punto di sincronizzazione non disponibile.

ERRORE TM_MQRC
(2265, X'8D9') Struttura del messaggio trigger non valida.

ERRORE MQRC_TMC_
(2191, X'88F') Struttura del messaggio trigger di caratteri non valida.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

MQRC_UNKNOWN_ALIAS_BASE_Q
(2082, X'822 ') Coda di base alias sconosciuta.

MQRC_UNKNOWN_DEF_XMIT_Q
(2197, X'895 ') Coda di trasmissione predefinita sconosciuta.

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825 ') Nome oggetto sconosciuto.

MQRC_UNKNOWN_OBJECT_Q_MGR
(2086, X'826 ') Gestore code oggetti sconosciuto.

MQRC_UNKNOWN_REMOTE_Q_MGR
(2087, X'827 ') Gestore code remoto sconosciuto.

MQRC_UNKNOWN_XMIT_Q
(2196, X'894 ') Coda di trasmissione sconosciuta.

ERRORE MQRC_UOW_ENLISTMENT_ERROR
(2354, X' 932 ') L'inserimento nell'unità di lavoro globale non è riuscito.

MQRC_UOW_MIX_NON_SUPPORTATO
(2355, X' 933 ') La miscelazione delle chiamate UOW non è supportata.

MQRC_UOW_NON_DISPONIBILE
(2255, X'8CF') Unità di lavoro non disponibile per il gestore code.

ERRORE MQRC_WIH
(2333, X'91D') Struttura MQWIH non valida.

MQRC_WRONG_CF_LEVEL
(2366, X'93E') La struttura della CF (Coupling Facility) è di livello errato.

MQRC_WRONG_MD_VERSIONE
(2257, X'8D1') Versione non corretta di MQMD fornita.

MQRC_XMIT_Q_TYPE_ERROR
(2091, X'82B') Coda di trasmissione non locale.

MQRC_XMIT_Q_USAGE_ERRORE
(2092, X'82C') Coda di trasmissione con utilizzo errato.

ERRORE MQRC_XQH
(2260, X'8D4') Struttura intestazione coda trasmissione non valida.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note d'utilizzo

1. Entrambe le chiamate MQPUT e MQPUT1 possono essere utilizzate per inserire i messaggi su una coda; la chiamata da utilizzare dipende dalle circostanze:
   - Utilizzare la chiamata MQPUT per inserire più messaggi nella stessa *coda* .

     Viene emessa prima una chiamata MQOPEN che specifica l'opzione MQOO_OUTPUT, seguita da una o più richieste MQPUT per aggiungere messaggi alla coda; infine, la coda viene chiusa con una chiamata MQCLOSE. Ciò fornisce prestazioni migliori rispetto all'utilizzo ripetuto della chiamata di MQPUT1 .
   - Utilizzare la chiamata MQPUT1 per inserire solo *un* messaggio su una coda.

     Questa chiamata incapsula le chiamate MQOPEN, MQPUT e MQCLOSE in una singola chiamata, riducendo il numero di chiamate che devono essere emesse.
2. Se un'applicazione inserisce una sequenza di messaggi nella stessa coda senza utilizzare i gruppi di messaggi, l'ordine di tali messaggi viene conservato se sono soddisfatte determinate condizioni. Tuttavia, nella maggior parte degli ambienti la chiamata MQPUT1 non soddisfa tali condizioni e quindi non conserva l'ordine dei messaggi. La chiamata MQPUT deve essere utilizzata in questi ambienti. Consultare Note sull'utilizzo di MQPUT per i dettagli.
3. La chiamata MQPUT1 può essere utilizzata per inserire i messaggi negli elenchi di distribuzione. Per informazioni generali su questo argomento, consultare le note di utilizzo per le chiamate MQOPEN e MQPUT.

   Gli elenchi di distribuzione sono supportati nei seguenti ambienti:
   - AIX
   - IBM i
   - Linux
   - Windows

   e per i client IBM MQ connessi a questi sistemi.

   Le seguenti differenze si applicano quando si utilizza la chiamata MQPUT1 :

   a. Se l'applicazione fornisce record di risposta MQRR, devono essere forniti utilizzando la struttura MQOD; non possono essere forniti utilizzando la struttura MQPMO.

   b. Il codice di errore MQRC_OPEN_FAILED non viene mai restituito da MQPUT1 nei record di risposta; se una coda non si apre, il record di risposta per tale coda contiene il codice di errore risultante dall'operazione di apertura.

      Se un'operazione aperta per una coda ha esito positivo con un codice di completamento MQCC_WARNING, il codice di completamento e il codice motivo nel record di risposta per tale coda vengono sostituiti dai codici di completamento e motivo risultanti dall'operazione di inserimento.

Come per le chiamate MQOPEN e MQPUT, il gestore code imposta i record di risposta (se forniti) solo quando il risultato della chiamata non è lo stesso per tutte le code nell'elenco di distribuzione; ciò è indicato dalla chiamata che si completa con il codice motivo MQRC_MULTIPLE_REASON.

4. Se la chiamata MQPUT1 viene utilizzata per inserire un messaggio in una coda cluster, la chiamata si comporta come se MQOO_BIND_NOT_FIXED fosse stato specificato nella chiamata MQOPEN.
5. Se un messaggio viene inserito con una o più strutture di intestazione IBM MQ all'inizio dei dati del messaggio dell'applicazione, il gestore code esegue determinati controlli sulle strutture di intestazione per verificarne la validità. Per ulteriori informazioni, consultare le note di utilizzo per la chiamata MQPUT.
6. Se si verifica più di una delle situazioni di avvertenza (consultare il parametro **CompCode**), il codice di errore restituito è il primo nel seguente elenco che si applica:
   a. MQRC_MULTIPLE_MOTIVI
   b. MQRC_INCOMPLETE_MSG
   c. GRUPPO_INCOMPLE_MQRC
   d. MQRC_PRIORITY_EXCEEDS_MAXIMUM o MQRC_UNKNOWN_REPORT_OPTION
7. Per il linguaggio di programmazione Visual Basic, si applicano i punti seguenti:
   - Se la dimensione del parametro **Buffer** è inferiore alla lunghezza specificata dal parametro **BufferLength** , la chiamata ha esito negativo con codice motivo MQRC_BUFFER_LENGTH_ERROR.
   - Il parametro **Buffer** viene dichiarato come di tipo `String`. Se i dati da inserire nella coda non sono di tipo `String`, utilizzare ilMQPUT1Any al posto di MQPUT1.

     La chiamata MQPUT1Any ha gli stessi parametri della chiamata MQPUT1 , ad eccezione del fatto che il parametro **Buffer** viene dichiarato come di tipo `Any`, consentendo l'inserimento di qualsiasi tipo di dati nella coda. Tuttavia, ciò significa che `Buffer` non può essere controllato per garantire che abbia una dimensione di almeno `BufferLength` byte.
8. Quando una chiamata MQPUT1 viene emessa con MQPMO_SYNCPOINT, il funzionamento predefinito cambia, in modo che l'operazione di inserimento venga completata in modo asincrono. Ciò potrebbe causare una modifica nel comportamento di alcune applicazioni che si basano su determinati campi nelle strutture MQOD e MQMD che vengono restituiti, ma che ora contengono valori non definiti. Un'applicazione può specificare MQPMO_SYNC_RESPONSE per assicurarsi che l'operazione di inserimento venga eseguita in modo sincrono e che tutti i valori di campo appropriati vengano completati.

Richiamo C

```
MQPUT1 (Hconn, &ObjDesc, &MsgDesc, &PutMsgOpts,
       BufferLength, Buffer, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;         /* Connection handle */
MQOD     ObjDesc;       /* Object descriptor */
MQMD     MsgDesc;       /* Message descriptor */
MQPMO    PutMsgOpts;    /* Options that control the action of MQPUT1 */
MQLONG   BufferLength;  /* Length of the message in Buffer */
MQBYTE   Buffer[n];     /* Message data */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
     CALL 'MQPUT1' USING HCONN, OBJDESC, MSGDESC, PUTMSGOPTS,
                         BUFFERLENGTH, BUFFER, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Object descriptor
 01  OBJDESC.
     COPY CMQODV.
**   Message descriptor
 01  MSGDESC.
     COPY CMQMDV.
**   Options that control the action of MQPUT1
 01  PUTMSGOPTS.
     COPY CMQPMOV.
**   Length of the message in BUFFER
 01  BUFFERLENGTH  PIC S9(9) BINARY.
**   Message data
 01  BUFFER        PIC X(n).
**   Completion code
 01  COMPCODE      PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQPUT1 (Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
            CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn         fixed bin(31);  /* Connection handle */
dcl ObjDesc       like MQOD;      /* Object descriptor */
dcl MsgDesc       like MQMD;      /* Message descriptor */
dcl PutMsgOpts    like MQPMO;     /* Options that control the action of
                                     MQPUT1 */
dcl BufferLength  fixed bin(31);  /* Length of the message in Buffer */
dcl Buffer        char(n);        /* Message data */
dcl CompCode      fixed bin(31);  /* Completion code */
dcl Reason        fixed bin(31);  /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
         CALL MQPUT1,(HCONN,OBJDESC,MSGDESC,PUTMSGOPTS,BUFFERLENGTH,   X
               BUFFER,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN         DS       F      Connection handle
OBJDESC       CMQODA   ,      Object descriptor
MSGDESC       CMQMDA   ,      Message descriptor
PUTMSGOPTS    CMQPMOA  ,      Options that control the action of MQPUT1
BUFFERLENGTH  DS       F      Length of the message in BUFFER
BUFFER        DS       CL(n)  Message data
COMPCODE      DS       F      Completion code
REASON        DS       F      Reason code qualifying COMPCODE
```

Richiamo Visual Basic

Windows

```
MQPUT1 Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
       CompCode, Reason
```

Dichiarare i parametri come segue:

```
Dim Hconn        As Long   'Connection handle'
Dim ObjDesc      As MQOD   'Object descriptor'
Dim MsgDesc      As MQMD   'Message descriptor'
Dim PutMsgOpts   As MQPMO  'Options that control the action of MQPUT1'
Dim BufferLength As Long   'Length of the message in Buffer'
Dim Buffer       As String 'Message data'
Dim CompCode     As Long   'Completion code'
Dim Reason       As Long   'Reason code qualifying CompCode'
```

MQSET - Imposta attributi oggetto

Utilizzare la chiamata MQSET per modificare gli attributi di un oggetto rappresentato da un handle. L'oggetto deve essere una coda.

Sintassi

MQSET (*Hconn*, *Hobj*, *SelectorCount*, *Selectors*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *Compcode*, *Causa*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di Hconn è stato restituito da una chiamata MQCONN o MQCONNX precedente.

z/OS In applicazioni z/OS per CICS , la chiamata MQCONN può essere omessa e il seguente valore specificato per *Hconn*:

DEF_MQH_HCONN
Handle di connessione predefinito.

HOBJ
Tipo: MQHOBJ - input

Questo handle rappresenta l'oggetto coda con gli attributi da impostare. L'handle è stato restituito da una chiamata MQOPEN precedente che specificava l'opzione MQOO_SET.

SelectorCount
Tipo: MQLONG - input

Questo è il conteggio dei selettori forniti nell'array Selectors . È il numero di attributi che devono essere impostati. Zero è un valore valido. Il numero massimo consentito è 256.

Selettori
Tipo: MQLONGxSelectorCount - input

Si tratta di un array di selettori di attributi **SelectorCount** ; ogni selettore identifica un attributo (numero intero o carattere) con un valore che deve essere impostato.

Ogni selettore deve essere valido per il tipo di coda rappresentato da Hobj . Solo alcuni valori MQIA_ * e MQCA_ * sono consentiti; come elencato in seguito.

I selettori possono essere specificati in qualsiasi ordine. I valori degli attributi che corrispondono ai selettori di attributi interi (selettori MQIA_ *) devono essere specificati in IntAttrs nello stesso ordine in cui tali selettori si verificano in Selectors. I valori di attributo che corrispondono ai selettori di attributi carattere (selettori MQCA_ *) devono essere specificati in CharAttrs nello stesso ordine in cui si verificano tali selettori. I selettori MQIA_ * possono essere interconnessi con i selettori MQCA_ *; è importante solo l'ordine relativo all'interno di ciascun tipo.

È possibile specificare lo stesso selettore più di una volta; se lo si fa, l'ultimo valore specificato per un particolare selettore è quello che diventa effettivo.

Nota:

1. I selettori degli attributi integer e character sono assegnati in due intervalli differenti; i selettori MQIA_ * si trovano nell'intervallo MQIA_FIRST - MQIA_LAST e i selettori MQCA_ * nell'intervallo MQCA_FIRST - MQCA_LAST.

   Per ogni intervallo, le costanti MQIA_LAST_USED e MQCA_LAST_USED definiscono il valore massimo accettato dal gestore code.
2. Se tutti i selettori MQIA_ * si verificano per primi, è possibile utilizzare gli stessi numeri di elemento per indirizzare gli elementi corrispondenti negli array `Selectors` e `IntAttrs` .
3. Se il parametro **`SelectorCount`** è zero, non si fa riferimento a `Selectors` ; in questo caso, l'indirizzo del parametro passato dai programmi scritti nell'assembler C o System/390 potrebbe essere null.

Gli attributi che è possibile impostare sono elencati nella seguente tabella. Non è possibile impostare altri attributi utilizzando questa chiamata. Per i selettori di attributi MQCA_ *, la costante che definisce la lunghezza in byte della stringa richiesta in `CharAttrs` viene fornita tra parentesi.

Tabella 554. Selettori di attributi MQSET per code

Selettore	Descrizione	Nota
DATI MQCA_TRIGGER_	Dati trigger (MQ_TRIGGER_DATA_LENGTH).	
MQIA_DIST_LISTS	Supporto elenco di distribuzione.	1
MQIA_INIBITORI_GET	Indica se le operazioni get sono consentite.	
MQIA_INIB_PUT	Se le operazioni di inserimento sono consentite.	
CONTROL MQIA_TRIGGER_	Controllo trigger.	
PROFONDITÀ trigger MQIA_	La lunghezza del trigger.	
MQIA_TRIGGER_MSG_PRIORITY	La priorità dei messaggi per i trigger.	
TIPO_TRIGGER_MQI	Il tipo di trigger.	

Nota:

1. Supportato solo sulle piattaforme seguenti:
   - AIX
   - IBM i
   - Linux
   - Windows

   e per IBM MQ MQI clients collegati a questi sistemi.

IntAttrCount
Tipo: MQLONG - input

Questo è il numero di elementi nell'array `IntAttrs` e deve essere almeno il numero di selettori MQIA_ * nel parametro **`Selectors`** . Zero è un valore valido se non ce ne sono.

IntAttrs
Tipo: MQLONGxIntAttrCount - input

Questo è un array di `IntAttrCount` valori di attributo integer. Questi valori di attributo devono essere nello stesso ordine dei selettori MQIA_ * nell'array `Selectors` .

Se il parametro **`IntAttrCount`** o **`SelectorCount`** è zero, `IntAttrs` non viene indicato; in questo caso, l'indirizzo del parametro passato dai programmi scritti nell'assembler C o System/390 potrebbe essere null.

CharAttrLunghezza
Tipo: MQLONG - input

Questa è la lunghezza in byte del parametro **CharAttrs** e deve essere almeno la somma delle lunghezze degli attributi carattere specificati nell'array `Selectors` . Zero è un valore valido se non sono presenti selettori MQCA_ * in `Selectors`.

CharAttrs
Tipo: MQCHAR x CharAttrLunghezza - input

Questo è il buffer contenente i valori di attributo carattere, concatenati insieme. La lunghezza del buffer viene fornita dal parametro **CharAttrLength** .

Gli attributi dei caratteri devono essere specificati nello stesso ordine dei selettori MQCA_ * nell'array `Selectors` . La lunghezza di ciascun attributo carattere è fissa (consultare Selettori). Se il valore da impostare per un attributo contiene un numero di caratteri non vuoti inferiore alla lunghezza definita dell'attributo, riempire il valore in `CharAttrs` a destra con spazi per far sì che il valore dell'attributo corrisponda alla lunghezza definita dell'attributo.

Se il parametro **CharAttrLength** o **SelectorCount** è zero, `CharAttrs` non viene indicato; in questo caso, l'indirizzo del parametro passato dai programmi scritti nell'assembler C o System/390 potrebbe essere null.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Il codice di errore che qualifica `CompCode`.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se `CompCode` è MQCC_FAILED:

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'89C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita.

ERRORE USCITA MQRC_API
(2183, X'887 ') Impossibile caricare l'uscita API.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

MQRC_CF_NOT_AVAILABLE
(2345, X' 929 ') La funzione di accoppiamento non è disponibile.

MQRC_CF_STRUC_NON RIUSCITO
(2373, X' 945 ') La struttura della funzione di accoppiamento non è riuscita.

MQRC_CF_STRUC_IN_USO
(2346, X'92A') Struttura CF in uso.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') L'elenco - intestazione della struttura CFS (Coupling Facility Structure) è in uso.

MQRC_CHAR_ATTR_LENGTH_ERROR
(2006, X'7D6') Lunghezza degli attributi dei caratteri non valida.

ERRORE MQRC_CHAR_ATTRS_
(2007, X'7D7') Stringa di attributi carattere non valida.

MQRC_CICS_WAIT_NON RIUSCITO
(2140, X'85C') Richiesta di attesa rifiutata da CICS.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Non autorizzato per la connessione.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Chiusura della connessione.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') Db2 sottosistema non disponibile.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

ERRORE MQRC_HOBJ_R
(2019, X'7E3') Handle oggetto non valido.

ERRORE MQRC_INHIBIT_VALUE_
(2020, X'7E4') Valore per l'attributo della coda di inibizione - ricezione o inibizione - inserimento non valido.

ERRORE MQRC_INT_ATTR_COUNT_
(2021, X'7E5') Conteggio di attributi interi non valido.

ERRORE - MQRC_INT_ATTRS_ARRAY_ERROR
(2023, X'7E7') Array di attributi interi non valido.

MQRC_NOT_OPEN_FOR_SET
(2040, X'7F8') Coda non aperta per il set.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Definizione oggetto modificata dall'apertura.

MQRC_OBJECT_DAMAGED
(2101, X'835 ') Oggetto danneggiato.

ERRORE MQRC_PAGESET_
(2193, X'891 ') Errore durante l'accesso al dataset della serie di pagine.

MQRC_Q_XX_ENCODE_CASE_ONE eliminato
(2052, X'804 ') La coda è stata eliminata.

ERRORE MQRC_Q_MGR_NAME_
(2058, X'80A') Nome gestore code non valido o sconosciuto.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Gestore code non disponibile per la connessione.

MQRC_Q_MGR_STOPPING
(2162, X'872 ') Chiusura del gestore code.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

ERRORE MQRC_SELECTOR_COUNT
(2065, X'811 ') Conteggio dei selettori non valido.

ERRORE DI MQRC_SELECTOR_ERROR
(2067, X'813 ') Selettore attributo non valido.

MQRC_SELECTOR_LIMIT_EXCEEDED
(2066, X'812 ') Conteggio dei selettori troppo grande.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Chiamata eliminata dal programma di uscita.

ERRORE MQRC_TRIGGER_CONTROL_
(2075, X'81B') Valore per l'attributo trigger - control non valido.

ERRORE MQRC_TRIGGER_DEPTH_ERROR
(2076, X'81C') Valore per l'attributo trigger - depth non valido.

MQRC_TRIGGER_MSG_PRIORITY_ERR
(2077, X'81D') Valore per l'attributo trigger - message - priority non valido.

ERRORE MQRC_TRIGGER_TIPO
(2078, X'81E') Valore per l'attributo di tipo trigger non valido.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note d'utilizzo

1. Utilizzando questa chiamata, l'applicazione può specificare un array di attributi interi, una raccolta di stringhe di attributi di caratteri o entrambi. Se non si verifica alcun errore, gli attributi specificati vengono tutti impostati simultaneamente. Se si verifica un errore (ad esempio, se un selettore non è valido o se si tenta di impostare un attributo su un valore non valido), la chiamata ha esito negativo e non viene impostato alcun attributo.
2. I valori degli attributi possono essere determinati utilizzando il richiamo MQINQ; consultare “MQINQ - Richiedi attributi oggetto” a pagina 723 per i dettagli.

   Nota: Non tutti gli attributi con valori che possono essere interrogati utilizzando la chiamata MQINQ possono avere i valori modificati utilizzando la chiamata MQSET. Ad esempio, non è possibile impostare alcun oggetto del processo o attributo del gestore code con questa chiamata.
3. Le modifiche agli attributi vengono conservate durante i riavvii del gestore code (tranne le modifiche alle code dinamiche temporanee, che non sopravvivono ai riavvii del gestore code).
4. Non è possibile modificare gli attributi di una coda modello utilizzando la chiamata MQSET. Tuttavia, se si apre una coda modello utilizzando la chiamata MQOPEN con l'opzione MQOO_SET, è possibile utilizzare la chiamata MQSET per impostare gli attributi della coda locale dinamica creata dalla chiamata MQOPEN.
5. Se l'oggetto impostato è una coda cluster, è necessario che sia presente un'istanza locale della coda cluster affinché l'apertura abbia esito positivo.

Per ulteriori informazioni sugli attributi dell'oggetto, consultare:

- “Attributi per le code” a pagina 858
- “Attributi per gli elenchi nomi” a pagina 892
- “Attributi per le definizioni di processi” a pagina 895
- “Attributi per il gestore code” a pagina 818

Richiamo C

```
MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;           /* Connection handle */
MQHOBJ   Hobj;            /* Object handle */
MQLONG   SelectorCount;   /* Count of selectors */
MQLONG   Selectors[n];    /* Array of attribute selectors */
MQLONG   IntAttrCount;    /* Count of integer attributes */
MQLONG   IntAttrs[n];     /* Array of integer attributes */
MQLONG   CharAttrLength;  /* Length of character attributes buffer */
MQCHAR   CharAttrs[n];    /* Character attributes */
MQLONG   CompCode;        /* Completion code */
MQLONG   Reason;          /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQSET' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,
                   INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,
                   CHARATTRS, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN            PIC S9(9) BINARY.
**   Object handle
 01  HOBJ             PIC S9(9) BINARY.
**   Count of selectors
 01  SELECTORCOUNT    PIC S9(9) BINARY.
**   Array of attribute selectors
 01  SELECTORS-TABLE.
  02  SELECTORS        PIC S9(9) BINARY OCCURS n TIMES.
**   Count of integer attributes
 01  INTATTRCOUNT     PIC S9(9) BINARY.
**   Array of integer attributes
 01  INTATTRS-TABLE.
  02  INTATTRS         PIC S9(9) BINARY OCCURS n TIMES.
**   Length of character attributes buffer
 01  CHARATTRLENGTH   PIC S9(9) BINARY.
**   Character attributes
 01  CHARATTRS        PIC X(n).
**   Completion code
 01  COMPCODE         PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON           PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
           IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn           fixed bin(31);  /* Connection handle */
dcl Hobj            fixed bin(31);  /* Object handle */
dcl SelectorCount   fixed bin(31);  /* Count of selectors */
dcl Selectors(n)    fixed bin(31);  /* Array of attribute selectors */
dcl IntAttrCount    fixed bin(31);  /* Count of integer attributes */
dcl IntAttrs(n)     fixed bin(31);  /* Array of integer attributes */
dcl CharAttrLength  fixed bin(31);  /* Length of character attributes
                                       buffer */
dcl CharAttrs       char(n);        /* Character attributes */
dcl CompCode        fixed bin(31);  /* Completion code */
```

```
dcl Reason          fixed bin(31);  /* Reason code qualifying
                                       CompCode */
```

Chiamata High Level Assembler

```
CALL MQSET,(HCONN,HOBJ,SELECTORCOUNT,SELECTORS,INTATTRCOUNT,  X
             INTATTRS,CHARATTRLENGTH,CHARATTRS,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN           DS  F      Connection handle
HOBJ            DS  F      Object handle
SELECTORCOUNT   DS  F      Count of selectors
SELECTORS       DS  (n)F   Array of attribute selectors
INTATTRCOUNT    DS  F      Count of integer attributes
INTATTRS        DS  (n)F   Array of integer attributes
CHARATTRLENGTH  DS  F      Length of character attributes buffer
CHARATTRS       DS  CL(n)  Character attributes
COMPCODE        DS  F      Completion code
REASON          DS  F      Reason code qualifying COMPCODE
```

Richiamo Visual Basic

```
MQSET Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, CompCode, Reason
```

Dichiarare i parametri come segue:

```
Dim Hconn          As Long   'Connection handle'
Dim Hobj           As Long   'Object handle'
Dim SelectorCount  As Long   'Count of selectors'
Dim Selectors      As Long   'Array of attribute selectors'
Dim IntAttrCount   As Long   'Count of integer attributes'
Dim IntAttrs       As Long   'Array of integer attributes'
Dim CharAttrLength As Long   'Length of character attributes buffer'
Dim CharAttrs      As String 'Character attributes'
Dim CompCode       As Long   'Completion code'
Dim Reason         As Long   'Reason code qualifying CompCode'
```

MQSETMP - Imposta proprietà messaggio

Utilizzare la chiamata MQSETMP per impostare o modificare una proprietà di un handle del messaggio.

Sintassi

MQSETMP (*Hconn*, *Hmsg*, *SetPropOpts*, *Nome*, *PropDesc*, *Tipo*, *ValueLength*, *Valore*, *Compcode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code.

Il valore deve corrispondere all'handle di connessione utilizzato per creare l'handle del messaggio specificato nel parametro **Hmsg** . Se l'handle del messaggio è stato creato utilizzando MQHC_UNASSOCIATED_HCONN, è necessario stabilire una connessione valida sul thread impostando una proprietà dell'handle del messaggio, altrimenti la chiamata non riesce con codice motivo MQRC_CONNECTION_BROKEN.

Msg
Tipo: MQHMSG - input

Questo è l'handle del messaggio da modificare. Il valore è stato restituito da una precedente chiamata MQCRTMH.

Opzioni SetProp
Tipo: MQSMPO - input

Controllare come sono impostate le proprietà del messaggio.

Questa struttura permette alle applicazioni di specificare le opzioni che controllano la modalità di impostazione delle proprietà del messaggio. La struttura è un parametro di input nella chiamata MQSETMP. Per ulteriori informazioni, consultare MQSMPO .

Nome
Tipo: MQCHARV - input

Questo è il nome della proprietà da impostare.

Consultare Nomi proprietà e Limitazioni nome proprietà per ulteriori informazioni sull'utilizzo dei nomi proprietà.

PropDesc
Tipo: MQPD - input/output

Questa struttura viene utilizzata per definire gli attributi di una proprietà, inclusi:

- cosa succede se la proprietà non è supportata
- a quale contesto di messaggio appartiene la propriet ...
- in quali messaggi viene copiata la propriet ... man mano che fluisce

Per ulteriori informazioni su questa struttura, consultare MQPD .

Tipo
Tipo: MQLONG - input

Il tipo di dati della proprietà impostata. Può essere uno dei seguenti valori:

BOOLEAN MQTIPO
Un booleano. *ValueLength* deve essere 4.

MQTYPE_BYTE_STRING
Una stringa di byte. *ValueLength* deve essere uguale o maggiore di zero.

MQTYPE_INT8
Un numero intero con segno a 8 bit. *ValueLength* deve essere 1.

MQTYPE_INT16
Un numero intero con segno a 16 bit. *ValueLength* deve essere 2.

MQTYPE_INT32
Un numero intero con segno a 32 bit. *ValueLength* deve essere 4.

MQTYPE_INT64
Un numero intero con segno a 64 bit. *ValueLength* deve essere 8.

MQTYPE_FLOAT32
Un numero a virgola mobile a 32 bit. *ValueLength* deve essere 4.

Nota: questo tipo non è supportato con applicazioni che utilizzano IBM COBOL per z/OS.

MQTYPE_FLOAT64
Un numero a virgola mobile a 64 bit. *ValueLength* deve essere 8.

Nota: questo tipo non è supportato con applicazioni che utilizzano IBM COBOL per z/OS.

MQTYPE_STRING
Una stringa di caratteri. *ValueLength* deve essere uguale o maggiore di zero o il valore speciale MQVL_NULL_TERMINATED.

MQTYPE_NULL
La proprietà esiste ma ha un valore null. *ValueLength* deve essere zero.

ValueLength
Tipo: MQLONG - input

La lunghezza in byte del valore della proprietà nel parametro *Valore* . Zero è valido solo per valori null o per stringhe o stringhe di byte. Zero indica che la proprietà esiste ma che il valore non contiene caratteri o byte.

Il valore deve essere maggiore o uguale a zero o al seguente valore speciale se il parametro *Type* ha MQTYPE_STRING impostato:

MQVL_NULL_TERMINATO
Il valore è delimitato dal primo null rilevato nella stringa. Il valore null non è incluso come parte della stringa. Questo valore non è valido se MQTYPE_STRING non è impostato.

Nota: il carattere null utilizzato per terminare una stringa se MQVL_NULL_TERMINATED è impostato su un valore null dalla serie di caratteri del valore.

Valore
Tipo: MQBYTExValueLength - input

Il valore della proprietà da impostare. Il buffer deve essere allineato su un limite appropriato alla natura dei dati nel valore.

Nel linguaggio di programmazione C, il parametro viene dichiarato come un puntatore a void; l'indirizzo di qualsiasi tipo di dati può essere specificato come parametro.

Se *ValueLength* è zero, non si fa riferimento a *Valore* . In questo caso, l'indirizzo del parametro inoltrato dai programmi scritti nell'assembler C o System/390 può essere null.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Il codice di errore che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

ERRORE MQRC_RFH_FORMATO
(2421, X'0975 ') Impossibile analizzare una cartella MQRFH2 contenente le proprietà.

Se *CompCode* è MQCC_FAILED:

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'089C') Adattatore non disponibile.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

MQRC_ASID_MISMATCH
(2157, X'86D') Gli ASID principale e home differiscono.

ERRORE MQRC_BUFFER_
(2004, X'07D4') Parametro valore non valido.

ERRORE MQRC_BUFFER_LENGTH
(2005, X'07D5') Parametro di lunghezza valore non valido.

MQRC_CALL_IN_PROVERDE
(2219, X'08AB') Chiamata MQI immessa prima del completamento della chiamata precedente.

ERRORE MQRC_HMSG_
(2460, X'099C') Il puntatore della gestione messaggi non è valido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') handle del messaggio già in uso.

ERRORE MQRC_OPTIONS_
(2046, X'07FE') Opzioni non valide o non congruenti.

ERRORE MQRC_PD
(2482, X'09B2') Struttura descrittore proprietà non valido.

ERRORE MQRC_PROPERTY_NAME_
(2442, X'098A') Nome proprietà non valido.

ERRORE TIPO_PROFILO_XX_ENCODE_CASE_CAPS_LOCK_OFF MQRC_
(2473, X'09A9') Tipo dati proprietà non valido.

MQRC_PROP_NUMBER_FORMAT_ERRORE
(2472, X'09A8') Errore di formato numero rilevato nei dati del valore.

ERRORE MQRC_SMPO
(2463, X'099F') Struttura delle opzioni della proprietà del messaggio impostata non valida.

ERRORE CCSID DI MQRC_SOURCE_
(2111, X'083F') Identificativo serie di caratteri codificato del nome proprietà non valido.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Richiamo C

```
MQSETMP (Hconn, Hmsg, &SetPropOpts, &Name, &PropDesc, Type,
ValueLength, &Value, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;       /* Connection handle */
MQHMSG   Hmsg;        /* Message handle */
MQSMPO   SetPropOpts; /* Options that control the action of MQSETMP */
MQCHARV  Name;        /* Property name */
MQPD     PropDesc;    /* Property descriptor */
MQLONG   Type;        /* Property data type */
MQLONG   ValueLength; /* Length of property value in Value */
MQBYTE   Value[n];    /* Property value */
MQLONG   CompCode;    /* Completion code */
MQLONG   Reason;      /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
     CALL 'MQSETMP' USING HCONN, HMSG, SETMSGOPTS, NAME, PROPDESC, TYPE,
                          VALUELENGTH, VALUE, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN     PIC S9(9) BINARY.
**   Message handle
 01  HMSG    PIC S9(18) BINARY.
```

```
**   Options that control the action of MQSETMP
 01  SETMSGOPTS.
     COPY CMQSMPOV.
**   Property name
 01  NAME
    COPY CMQCHRVV.
**   Property descriptor
 01  PROPDESC.
     COPY CMQPDV.
**   Property data type
 01  TYPE          PIC S9(9) BINARY.
**   Length of property value in VALUE
 01  VALUELENGTH   PIC S9(9) BINARY.
**   Property value
 01  VALUE         PIC X(n).
**   Completion code
 01  COMPCODE  PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON    PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQSETMP (Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength,
              Value, CompCode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn       fixed bin(31);  /* Connection handle */
dcl Hmsg        fixed bin(63);  /* Message handle */
dcl SetPropOpts like MQSMPO;    /* Options that control the action of MQSETMP */
dcl Name        like MQCHARV;   /* Property name */
dcl PropDesc    like MQPD;      /* Property descriptor */
dcl Type        fixed bin(31);  /* Property data type */
dcl ValueLength fixed bin(31);  /* Length of property value in Value */
dcl Value       char(n);        /* Property value */
dcl CompCode    fixed bin(31);  /* Completion code */
dcl Reason      fixed bin(31);  /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
         CALL MQSETMP,(HCONN,HMSG,SETMSGHOPTS,NAME,PROPDESC,TYPE,VALUELENGTH,
               VALUE,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN        DS         F     Connection handle
HMSG         DS         D     Message handle
SETMSGOPTS   CMQSMPOA   ,     Options that control the action of MQSETMP
NAME         CMQCHRVA   ,     Property name
PROPDESC     CMQPDA     ,     Property descriptor
TYPE         DS         F     Property data type
VALUELENGTH  DS         F     Length of property value in VALUE
VALUE        DS         CL(n) Property value
COMPCODE     DS         F     Completion code
REASON       DS         F     Reason code qualifying COMPCODE
```

MQSTAT - Richiamo delle informazioni di stato

Utilizzare la chiamata MQSTAT per richiamare le informazioni sullo stato. Il tipo di informazioni di stato restituite è determinato dal valore Tipo specificato nella chiamata.

Sintassi

MQSTAT (*Hconn*, *Tipo*, *Stat*, *Compcode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

In applicazioni z/OS per CICS , la chiamata MQCONN può essere omessa e il seguente valore specificato per *Hconn*:

DEF_MQH_HCONN
Handle di connessione predefinito.

Tipo
Tipo: MQLONG - input

Tipo di informazioni di stato richieste. I valori> validi sono:

MQSTAT_TYPE_ASYNC_ERROR
Restituisce le informazioni sulle precedenti operazioni di inserimento asincrone.

RICONNESSIONE_TIPO_MQSTAT

Restituisce le informazioni sulla riconnessione. Se la connessione è in fase di riconnessione o non è riuscita, le informazioni descrivono l'errore che ha causato l'avvio della riconnessione.

Questo valore è valido solo per connessioni client. Per altri tipi di connessione, la chiamata ha esito negativo con codice motivo **MQRC_ENVIRONMENT_ERROR**

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Restituisce informazioni su un errore precedente relativo alla riconnessione. Se la connessione non è riuscita a riconnettersi, le informazioni descrivono l'errore che ha causato il malfunzionamento della riconnessione.

Questo valore è valido solo per connessioni client. Per altri tipi di connessione, la chiamata ha esito negativo con codice di errore **MQRC_ENVIRONMENT_ERROR**.

stat
Tipo: MQSTS - input/output

Struttura delle informazioni sullo stato. Vedi "MQSTS - Struttura di report di stato" a pagina 607 per i dettagli.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Il codice di errore che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE USCITA MQRC_API
(2374, X' 946 ') Uscita API non riuscita

ERRORE USCITA MQRC_API
(2183, X'887 ') Impossibile caricare l'uscita API.

MQRC_CALL_IN_PROVERDE
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Connessione al gestore code persa.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Chiusura della connessione.

MQRC_FUNZIONE_NON_SUPPORTATA
(2298, X'8FA') La funzione richiesta non è disponibile nell'ambiente corrente.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

MQRC_Q_MGR_STOPPING
(2162,X'872' - Arresto del gestore code

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

ERRORE TIPO_STATO_MQRC
(2430, X'97E' Errore con tipo MQSTAT

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

ERRORE MQRC_STS_
(2426, X'97A') Errore con la struttura MQSTS

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note d'utilizzo

1. Una chiamata a MQSTAT che specifica un tipo di MQSTAT_TYPE_ASYNC_ERROR restituisce informazioni sulle precedenti operazioni asincrone MQPUT e MQPUT1 . La struttura MQSTS restituita dalla chiamata MQSTAT contiene le prime informazioni di errore o di avvertenza asincrone registrate per tale connessione. Se ulteriori errori o avvertenze seguono il primo, normalmente non modificano questi valori. Tuttavia, se si verifica un errore con un codice di completamento MQCC_WARNING, viene restituito un errore successivo con un codice di completamento MQCC_FAILED .
2. Se non si sono verificati errori dal momento in cui è stata stabilita la connessione o dall'ultima chiamata a MQSTAT , nella struttura MQSTS vengono restituiti un CompCode di MQCC_OK e un motivo di MQRC_NONE .
3. I conteggi del numero di chiamate asincrone che sono state elaborate sotto l'handle di connessione vengono restituiti per mezzo di tre campi contatore; PutSuccessCount, PutWarningCount e PutFailureCount. Questi contatori vengono incrementati dal gestore code ogni volta che un'operazione asincrona viene elaborata correttamente, ha un'avvertenza o ha esito negativo (notare che, per motivi di account, un inserimento in un elenco di distribuzione viene contato una volta per coda di destinazione anziché una volta per elenco di distribuzione). Un contatore non viene incrementato oltre il valore positivo massimo AMQ_LONG_MAX.
4. Una chiamata riuscita a MQSTAT comporta la reimpostazione di eventuali precedenti informazioni di errore o conteggi.
5. Il funzionamento di MQSTAT dipende dal valore del parametro **MQSTAT Type** fornito.
6. **MQSTAT_TYPE_ASYNC_ERROR**
    a. Una chiamata a MQSTAT che specifica un tipo di MQSTAT_TYPE_ASYNC_ERROR restituisce informazioni sulle precedenti operazioni asincrone MQPUT e MQPUT1 . La struttura MQSTS restituita dalla chiamata MQSTAT contiene le prime informazioni di errore o di avvertenza

asincrone registrate per tale connessione. Se ulteriori errori o avvertenze seguono il primo, normalmente non modificano questi valori. Tuttavia, se si verifica un errore con un codice di completamento MQCC_WARNING, viene restituito un errore successivo con un codice di completamento MQCC_FAILED .

b. Se non si sono verificati errori dal momento in cui è stata stabilita la connessione o dall'ultima chiamata a MQSTAT , nella struttura MQSTS vengono restituiti un CompCode di MQCC_OK e un motivo di MQRC_NONE .

c. I conteggi del numero di chiamate asincrone che sono state elaborate sotto l'handle di connessione vengono restituiti per mezzo di tre campi contatore; PutSuccessCount, PutWarningCount e PutFailureCount. Questi contatori vengono incrementati dal gestore code ogni volta che un'operazione asincrona viene elaborata correttamente, ha un'avvertenza o ha esito negativo (notare che, per motivi di account, un inserimento in un elenco di distribuzione viene contato una volta per coda di destinazione anziché una volta per elenco di distribuzione). Un contatore non viene incrementato oltre il valore positivo massimo AMQ_LONG_MAX.

d. Una chiamata riuscita a MQSTAT comporta la reimpostazione di eventuali precedenti informazioni di errore o conteggi.

RICONNESSIONE_TIPO_MQSTAT

Si supponga di richiamare MQSTAT con Type impostato su MQSTAT_TYPE_RECONNECTION all'interno di un gestore eventi durante la riconnessione. Considerare questi esempi.

Il client sta tentando la riconnessione o non è riuscito a riconnettersi.
CompCode nella struttura MQSTS è MQCC_FAILED e Reason potrebbe essere MQRC_CONNECTION_BROKEN o MQRC_Q_MGR_QUIESCING. ObjectType è MQOT_Q_MGR, ObjectName è il nome del gestore code e ObjectQMgrName è vuoto.

Il client ha completato correttamente la riconnessione o non è mai stato disconnesso.
CompCode nella struttura MQSTS è MQCC_OK e Reason è MQRC_NONE

Le chiamate successive a MQSTAT restituiscono gli stessi risultati.

ERRORE_RICONNESSIONE_TIPO_MQSTAT_

Si supponga di richiamare MQSTAT con Type impostato a MQSTAT_TYPE_RECONNECTION_ERROR in risposta alla ricezione di MQRC_RECONNECT_FAILED di una chiamata MQI. Considerare questi esempi.

Si è verificato un errore di autorizzazione durante la riapertura di una coda durante la riconnessione a un gestore code differente.
CompCode nella struttura MQSTS è MQCC_FAILED e Reason è il motivo per cui la riconnessione ha avuto esito negativo, ad esempio MQRC_NOT_AUTHORIZED. ObjectType è il tipo di oggetto che ha causato il problema, come ad esempio MQOT_QUEUE, ObjectName è il nome della coda e ObjectQMgrName il nome del gestore code proprietario della coda.

Si è verificato un errore di connessione socket durante la riconnessione.
CompCode nella struttura MQSTS è MQCC_FAILED e Reason è il motivo per cui la riconnessione ha avuto esito negativo, ad esempio MQRC_HOST_NOT_AVAILABLE. ObjectType è MQOT_Q_MGR, ObjectName è il nome del gestore code e ObjectQMgrName è vuoto.

Le chiamate successive a MQSTAT restituiscono gli stessi risultati.

Richiamo C

```
MQSTAT (Hconn, StatType, &Stat, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN Hconn;        /* Connection Handle */
MQLONG StatType;    /* Status type */
MQSTS Stat;           /* Status information structure */
```

```
MQLONG CompCode;    /* Completion code */
MQLONG Reason;        /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQSTAT' USING HCONN, STATTYPE, STAT, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**    Connection handle
 01    HCONN      PIC S9(9)      BINARY.
**    Status type
 01    STATTYPE   PIC S9(9)      BINARY.
**    Status information
 01    STAT.
    COPY CMQSTSV.
**    Completion code
 01    COMPCODE   PIC S9(9)      BINARY.
**    Reason code qualifying COMPCODE
 01    REASON     PIC S9(9)      BINARY.
```

Chiamata PL/I

```
call MQSTAT (Hconn, StatType, Stat, Compcode, Reason);
```

Dichiarare i parametri come segue:

```
dcl Hconn       fixed bin(31); /* Connection handle */
dcl StatType    fixed bin(31); /* Status type */
dcl Stat        like MQSTS;    /* Status information structure */
dcl CompCode    fixed bin(31); /* Completion code */
dcl Reason      fixed bin(31); /* Reason code qualifying CompCode */
```

System/390 Richiamo Assembler

```
CALL MQSTAT,(HCONN,STATTYPE,STAT,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN     DS     F    Connection handle
STATTYPE  DS     F    Status type
STAT      CMQSTSA,    Status information structure
COMPCODE  DS     F    Completion code
REASON    DS     F    Reason code qualifying COMPCODE
```

MQSUB - Registrazione sottoscrizione

Utilizzare la chiamata MQSUB per registrare la sottoscrizione delle applicazioni a un particolare argomento.

Sintassi

MQSUB (*Hconn*, *SubDesc*, *Hobj*, *Hsub*, *Compcode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

In applicazioni z/OS per CICS , la chiamata MQCONN può essere omessa e il seguente valore specificato per *Hconn*:

DEF_MQH_HCONN
Handle di connessione predefinito.

SubDesc
Tipo: MQSD - input/output

Si tratta di una struttura che identifica l'oggetto in uso registrato dall'applicazione. Per ulteriori informazioni, fare riferimento a "MQSD - Descrittore sottoscrizione" a pagina 582.

HOBJ
Tipo: MQHOBJ - input/output

Questo handle rappresenta l'accesso stabilito per ottenere i messaggi inviati a questa sottoscrizione. Questi messaggi possono essere memorizzati su una coda specifica oppure il gestore code può gestire la propria memoria senza utilizzare una coda specifica.

Per utilizzare una coda specifica, è necessario associarla alla sottoscrizione quando viene creata. Ciò può essere fatto in due modi:

- Utilizzando il comando DEFINE SUB MQSC e fornito tale comando con il nome di un oggetto coda.
- Fornendo questo handle quando si richiama MQSUB con MQSO_CREATE

  Se questo handle viene fornito come parametro di input sulla chiamata, deve essere un handle di oggetto valido restituito da una precedente chiamata MQOPEN di una coda utilizzando almeno una delle seguenti opzioni:

  – MQOO_INPUT_ *
  – MQOO_SFOGLIA
  – MQOO_OUTPUT (se la coda è una coda remota)

  In caso contrario, la chiamata ha esito negativo con MQRC_HOBJ_ERROR. Non può essere un handle di oggetto per una coda alias che si risolve in un oggetto argomento. In caso affermativo, la chiamata ha esito negativo con MQRC_HOBJ_ERROR.

Se il gestore code deve gestire la memoria dei messaggi inviati a questa sottoscrizione, questo valore deve essere impostato quando si crea la sottoscrizione, utilizzando l'opzione MQSO_MANAGED. Il gestore code restituisce quindi questo handle come parametro di output sulla chiamata. L'handle restituito è noto come handle gestito. Se MQHO_NONE è specificato ma MQSO_MANAGED non è stato specificato, la chiamata ha esito negativo con MQRC_HOBJ_ERROR.

Quando un gestore code restituisce un handle gestito all'utente, è possibile utilizzarlo su una chiamata MQGET o MQCB con o senza opzioni di esplorazione, su una chiamata MQINQ o su MQCLOSE. Non è possibile utilizzarlo in MQPUT, MQSUB, MQSET; il tentativo di eseguire tale operazione non riesce con MQRC_NOT_OPEN_FOR_OUTPUT, MQRC_HOBJ_ERROR o MQRC_NOT_OPEN_FOR_SET.

Se questa sottoscrizione viene ripresa utilizzando l'opzione MQSO_RESUME nella struttura MQSD, l'handle può essere restituito all'applicazione in questo parametro impostando MQSO_MANAGED su MQHO_NONE. È possibile eseguire questa operazione se la sottoscrizione sta utilizzando un handle gestito o meno e può essere utile fornire le sottoscrizioni create utilizzando DEFINE SUB con l'handle per la coda di sottoscrizione definita su tale comando. Nel caso in cui venga ripresa una sottoscrizione creata amministrativamente, la coda si apre con MQOO_INPUT_AS_Q_DEF e MQOO_BROWSE. Se è necessario specificare altre opzioni, l'applicazione deve aprire esplicitamente la coda di sottoscrizione e fornire l'handle dell'oggetto sulla chiamata. Se si verifica un problema durante l'apertura della coda, la chiamata ha esito negativo con MQRC_INVALID_DESTINATION. Se viene fornito *Hobj* , deve essere equivalente a *Hobj* nella chiamata MQSUB originale. Ciò significa che se viene fornita una gestione oggetto restituita da una chiamata MQOPEN, la gestione deve essere nella stessa coda utilizzata in precedenza. Se non è la stessa coda, la chiamata ha esito negativo con MQRC_HOBJ_ERROR.

Se questa sottoscrizione viene modificata utilizzando l'opzione MQSO_ALTER nella struttura MQSD, è possibile fornire un *Hobj* differente. Tutte le pubblicazioni che sono state consegnate alla coda e che sono state precedentemente identificate tramite questo parametro restano su quella coda ed è responsabilità dell'applicazione richiamare tali messaggi se il parametro **Hobj** ora rappresenta una coda diversa.

Tabella 555. Utilizzo di hobj con varie opzioni di sottoscrizione

Opzioni	***Hobj***	**Descrizione**
MQSO_CREATE + MQSO_MANAGED	Ignorato all'immissione	Crea una sottoscrizione con memoria dei messaggi gestiti dal gestore code
CREA_MQSO	Un handle di oggetto valido	Crea una sottoscrizione fornendo una coda specifica come destinazione per i messaggi.
RESUME MQSO	MQHO_NONE	Ripristina una sottoscrizione creata in precedenza, indipendentemente dal fatto che sia stata gestita o meno, e fa in modo che il gestore code restituisca l'handle dell'oggetto per l'utilizzo da parte dell'applicazione.
RESUME MQSO	Un handle di oggetto valido, corrispondente	Riprende una sottoscrizione precedentemente creata che utilizza una coda specifica come destinazione per i messaggi e utilizza un handle di oggetto con opzioni di apertura specifiche.
MQSO_ALTER + MQSO_MANAGED	MQHO_NONE	Modifica una sottoscrizione esistente che in precedenza utilizzava una coda specifica, quindi ora è una sottoscrizione gestita. La classe di destinazione (gestita o meno) non può essere modificata.
ALTER MQSO	Un handle di oggetto valido	Modifica una sottoscrizione esistente, che sia stata gestita o meno, in modo che ora utilizzi una coda specifica. Quando l'opzione MQSO_MANAGED non viene utilizzata, la coda fornita può essere modificata, ma la classe di destinazione (gestita o meno) non può essere modificata.

Se è stato fornito o restituito, *Hobj* deve essere specificato nelle successive chiamate MQGET o MQCB che desiderano ricevere i messaggi di pubblicazione inviati a questa sottoscrizione.

L'handle *Hobj* non è più valido quando viene emessa la chiamata MQCLOSE o quando l'unità di elaborazione che definisce l'ambito dell'handle termina (fino a quando l'applicazione non si disconnette). L'ambito dell'handle dell'oggetto restituito è uguale a quello dell'handle di collegamento specificato nella chiamata. Consultare Hconn (MQHCONN) - output per informazioni sull'ambito della gestione. Un MQCLOSE dell'handle *Hobj* non influisce sull'handle *Hsub* .

HSub
Tipo: MQHOBJ - output

Questo handle rappresenta la sottoscrizione effettuata. Può essere utilizzato per altre due operazioni:

- Può essere utilizzato su una chiamata MQSUBRQ successiva per richiedere che le pubblicazioni vengano inviate quando è stata utilizzata l'opzione MQSO_PUBLICATIONS_ON_REQUEST durante l'esecuzione della sottoscrizione.
- Può essere utilizzato su una chiamata MQCLOSE successiva per rimuovere la sottoscrizione effettuata. L'handle *Hsub* cessa di essere valido quando viene emessa la chiamata MQCLOSE o quando l'unità di elaborazione che definisce l'ambito dell'handle termina. L'ambito dell'handle dell'oggetto restituito è uguale a quello dell'handle di collegamento specificato nella chiamata. Un MQCLOSE dell'handle *Hsub* non influisce sull'handle *Hobj* .

Questo handle non può essere passato a una chiamata MQGET o MQCB. È necessario utilizzare il parametro **Hobj** . Non è possibile utilizzare questo handle su qualsiasi chiamata IBM MQ diversa da MQCLOSE o MQSUBRQ. La trasmissione di questo handle a qualsiasi altra chiamata IBM MQ risulta in MQRC_HOBJ_ERROR.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento riuscito

MQCC_AVVERTENZA
Avvertenza (completamento parziale)

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo
Tipo: MQLONG - output

Il codice di errore che qualifica *CompCode*.

Se *CompCode* è MQCC_OK, il codice motivo è il seguente:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED, il codice motivo è uno dei seguenti:

MQRC_CLUSTER_RESOLUTION_ERRORE
(2189, X'88D') Risoluzione del nome cluster non riuscita.

MQRC_DURABILITY_NOT_ALLOWED
2436 (X'0984 ') Una chiamata MQSUB che utilizza l'opzione MQSO_DURABLE non è riuscita.

MQRC_FUNZIONE_NON_SUPPORTATA
2298 (X'08FA') La funzione richiesta non è disponibile nell'ambiente corrente.

ERRORE MQRC_HOBJ_R
2019 (X'07E3') Handle Hobj non valido.

MQRC_IDENTITY_MISMATCH
2434 (X'0982 ') Il nome della sottoscrizione corrisponde alla sottoscrizione esistente.

MQRC_NOT_AUTHORIZED
2035 (X'07F3') L'utente non è autorizzato ad eseguire l'operazione.

MQRC_NO_SUBSCRIPZIONE
2428 (X'097C') Il nome della sottoscrizione identificato non esiste.

ERRORE STRINGA MQRC_OBJECT_
2441 (X'0989 ') Campo stringa oggetto non valido.

ERRORE MQRC_OPTIONS_
2046 (X'07FE') Il campo o il parametro Opzioni contiene opzioni non valide o una combinazione di opzioni non valide.

MQRC_Q_MGR_QUIESCING
2161 (X'0871 ') Gestore code in fase di sospensione.

MQRC_RECONNECT_Q_MGR_REQD
2555 (X'09FB' X) L'opzione MQCNO_RECONNECT_Q_MGR è obbligatoria.

MQRC_RETAINED_MSG_Q_ERROR
2525 (X'09DD') Le pubblicazioni conservate che esistono per la stringa di argomenti sottoscritta non possono essere richiamate.

MQRC_RETAINED_NOT_DELIVERED
2526 (X'09DE') Le pubblicazioni conservate che esistono per la stringa di argomenti sottoscritta, non possono essere consegnate alla coda di destinazione della sottoscrizione e non possono essere consegnate alla coda di messaggi non recapitabili.

ERRORE MQRC_S
2424 (X'0978 ') Descrittore sottoscrizione (MQSD) non valido.

SELEZIONE_MQRC_NON_DISPONIBILE
2551 (X'09F7') La stringa di selezione non segue la sintassi del selettore IBM MQ e non era disponibile alcun fornitore di selezione dei messaggi estesi.

ERRORE MQRC_SELECTION_STRING_
2519 (X'09D7') La stringa di selezione deve essere specificata come descritto nella documentazione della struttura MQCHARV.

ERRORE MQRC_SELECTOR_SYNTAX_ERROR
2459 (X'099B') È stata emessa una chiamata MQOPEN, MQPUT1o MQSUB, ma è stata specificata una stringa di selezione contenente un errore di sintassi.

ERRORE MQRC_SUB_USER_DATA_
2431 (X'097F') SubUserCampo dati non valido.

ERRORE MQRC_SUB_NAME_
2440 (X'0988 ') Campo SubName non valido.

MQRC_SUB_ALREADY_EXISTS
2432 (X'0980 ') La sottoscrizione esiste già.

ERRORE MQRC_SUB_USER_DATA_
2431 (X'097F') SubUserCampo dati non valido.

ERRORE STRINGA MQRC_TOPIC_
2425 (X'0979 ') La stringa argomento non è valida.

MQRC_UNKNOWN_OBJECT_NAME
2085 (X'0825 ') Impossibile trovare l'oggetto identificato nel campo MQSD ObjectName .

MQRC_SUB_JOIN_NOT_ALTERABLE
29440 (X'7300 ') La modalità di condivisione della sottoscrizione è incompatibile con la sottoscrizione esistente. Questo errore potrebbe essere restituito quando si tenta di riprendere una sottoscrizione condivisa JMS 2.0 in una applicazione non JMS.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note d'utilizzo

- La sottoscrizione viene effettuata a un argomento, denominato utilizzando il nome breve di un oggetto argomento predefinito, il nome completo della stringa argomento o è formato dalla concatenazione di due parti. Consultare la descrizione di *ObjectName* e *ObjectString* in “MQSD - Descrittore sottoscrizione” a pagina 582.
- Il gestore code esegue i controlli di sicurezza quando viene emessa una chiamata MQSUB, per verificare che l'identificativo utente con cui l'applicazione è in esecuzione disponga del livello di autorizzazione appropriato prima che sia consentito l'accesso. L'oggetto argomento appropriato si trova nella gerarchia degli argomenti e viene effettuato un controllo di autorizzazione su questo oggetto argomento per garantire che sia impostata l'autorizzazione alla sottoscrizione. Se l'opzione MQSO_MANAGED non viene utilizzata, viene eseguito un controllo dell'autorizzazione sulla coda di destinazione per assicurarsi

che l'autorizzazione per l'output sia impostata. Se viene utilizzata l'opzione MQSO_MANAGED, non viene eseguito alcun controllo di autorizzazione sulla coda gestita per l'emissione o l'accesso all'interrogazione.

- Se non si fornisce un Hobj come input, la chiamata MQSUB assegna due handle, un handle di oggetto (Hobj) e un handle di sottoscrizione (Hsub).
- L'Hobj restituito sulla chiamata MQSUB quando viene utilizzata l'opzione MQSO_MANAGED, può essere interrogato per trovare attributi come la soglia di backout e il nome della riaccodamento di backout eccessivo. È anche possibile analizzare il nome della coda gestita, ma non si deve tentare di aprire direttamente questa coda.
- Le sottoscrizioni possono essere raggruppate consentendo la distribuzione di una sola pubblicazione al gruppo di sottoscrizioni anche se più di un gruppo corrispondeva alla pubblicazione. Le sottoscrizioni sono raggruppate utilizzando l'opzione MQSO_GROUP_SUB e per raggruppare le sottoscrizioni devono essere
  - utilizzando la stessa coda denominata (che non sta utilizzando l'opzione MQSO_MANAGED) sullo stesso gestore code - rappresentato dal parametro Hobj nella chiamata MQSUB
  - condividere lo stesso ID SubCorrel
  - essere dello stesso SubLevel

  Questi attributi definiscono la serie di sottoscrizioni considerate nel gruppo e sono anche gli attributi che non possono essere modificati se una sottoscrizione è raggruppata. La modifica di SubLevel risulta in MQRC_SUBLEVEL_NOT_ALTERABLE e la modifica di uno degli altri (che può essere modificato se una sottoscrizione non è raggruppata) risulta in MQRC_GROUPING_NOT_ALTERABLE.
- Il corretto completamento della chiamata MQSUB non significa che l'azione sia stata completata. Per controllare che questa chiamata sia stata completata, consultare il passo DEFINE SUB in Verifica del completamento dei comandi asincroni per le reti distribuite.
- I campi in MQSD vengono compilati al ritorno da una chiamata MQSUB che utilizza l'opzione MQSO_RESUME. L'MQSD restituito può essere inoltrato direttamente in una chiamata MQSUB che utilizza l'opzione MQSO_ALTER con tutte le modifiche che è necessario apportare alla sottoscrizione applicata all'MQSD. Alcuni campi hanno considerazioni speciali come indicato nella tabella.

Tabella 556. Considerazioni speciali per i campi in MQDS

Nome campo in MQSD	**Considerazioni speciali**
Opzioni di accesso o di creazione	Alcune delle opzioni possono essere reimpostate al ritorno dalla chiamata MQSUB. Se si riutilizza MQSD in una chiamata MQSUB, l'opzione richiesta deve essere impostata esplicitamente.
Opzioni di durata, Opzioni di destinazione, Opzioni di registrazione & Opzioni Wildcard	Queste opzioni vengono impostate in base alle esigenze
Opzioni di pubblicazione	Queste opzioni sono impostate in base alle esigenze, ad eccezione di MQSO_NEW_PUBLICATIONS_ONLY, applicabile solo a MQSO_CREATE.
Altre opzioni	Queste opzioni non vengono modificate al ritorno da una chiamata MQSUB. Controllano come viene emessa la chiamata API e non vengono memorizzati con la sottoscrizione. Devono essere impostati come richiesto in qualsiasi successiva chiamata MQSUB che riutilizzi MQSD.
ObjectName	Questo campo di sola immissione non viene modificato al ritorno da una chiamata MQSUB.

Tabella 556. Considerazioni speciali per i campi in MQDS (Continua)

Nome campo in MQSD	**Considerazioni speciali**
ObjectString	Questo campo di sola immissione non viene modificato al ritorno da una chiamata MQSUB. Il nome completo dell'argomento utilizzato viene restituito nel campo *ResObjectString* , se viene fornito un buffer.
ID AlternateUsere ID AlternateSecurity	Questi campi di solo input non vengono modificati al ritorno da una chiamata MQSUB. Controllano come viene emessa la chiamata API e non vengono memorizzati con la sottoscrizione. Devono essere impostati come richiesto su qualsiasi chiamata MQSUB successiva che riutilizzi MQSD.
SubExpiry	Al ritorno da una chiamata MQSUB utilizzando l'opzione MQSO_RESUME, questo campo è impostato sulla scadenza originale della sottoscrizione e non sul tempo di scadenza rimanente. Se si riutilizza MQSD in una chiamata MQSUB utilizzando l'opzione MQSO_ALTER, si reimposta la scadenza della sottoscrizione per avviare di nuovo il conto alla rovescia.
SubName	Questo campo è un campo di input su una chiamata MQSUB e non viene modificato sull'output.
SubUserData e SelectionString	Questi campi a lunghezza variabile vengono restituiti in output da una chiamata MQSUB utilizzando l'opzione MQSO_RESUME, se viene fornito un buffer, e anche una lunghezza del buffer positiva in *VSBufSize*. Se non viene fornito alcun buffer, viene restituita solo la lunghezza nel campo *VSLength* di MQCHARV. Se il buffer fornito è più piccolo dello spazio richiesto per restituire il campo, nel buffer fornito vengono restituiti solo *VSBufSize* byte. Se si riutilizza MQSD in una chiamata MQSUB utilizzando l'opzione MQSO_ALTER e non viene fornito un buffer, ma viene fornito un *VSLength* diverso da zero, se tale lunghezza corrisponde alla lunghezza esistente del campo, non viene apportata alcuna modifica al campo.
Token SubCorrelId e PubAccounting	Se non si utilizza MQSO_SET_CORREL_ID, il *SubCorrelId* viene generato dal gestore code. Se non si utilizza MQSO_SET_IDENTITY_CONTEXT, il *PubAccountingToken* viene generato dal gestore code. Questi campi vengono restituiti in MQSD da una chiamata MQSUB utilizzando l'opzione MQSO_RESUME. Se sono generati dal gestore code, il valore generato viene restituito su una chiamata MQSUB utilizzando l'opzione MQSO_CREATE o MQSO_ALTER.
PubPriority, SubLevel & PubApplIdentityData	Questi campi vengono restituiti in MQSD.

Tabella 556. Considerazioni speciali per i campi in MQDS (Continua)	
Nome campo in MQSD	**Considerazioni speciali**
Stringa ResObject	Questo campo di output viene restituito solo in MQSD se viene fornito un buffer.

Richiamo C

```
MQSUB (Hconn, &SubDesc, &Hobj, &Hsub, &CompCode, &Reason)
```

Dichiarare i parametri come segue:

```
MQHCONN Hconn;    /* Connection handle */
MQSD    SubDesc;  /* Subscription descriptor */
MQHOBJ  Hobj;     /* Object handle */
MQHOBJ  Hsub;     /* Subscription handle */
MQLONG  CompCode; /* Completion code */
MQLONG  Reason;   /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQSUB' USING HCONN, SUBDESC, HOBJ, HSUB, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN    PIC S9(9) BINARY.
**   Subscription descriptor
 01  SUBDESC.
     COPY CMQSDV.
**   Object handle
 01  HOBJ     PIC S9(9) BINARY.
**   Subscription handle
 01  HSUB     PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON   PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQSUB (Hconn, SubDesc, Hobj, Hsub, CompCode, Reason)
```

Dichiarare i parametri come segue:

```
dcl Hconn    fixed bin(31); /* Connection handle */
dcl SubDesc  like MQSD;     /* Subscription descriptor */
dcl Hobj     fixed bin(31); /* Object handle */
dcl Hsub     fixed bin(31); /* Subscription handle */
dcl CompCode fixed bin(31); /* Completion code */
dcl Reason   fixed bin(31); /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQSUB,(HCONN,SUBDESC,HOBJ,HSUB,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN    DS      F  Connection handle
SUBDESC  CMQSDA  ,  Subscription descriptor
HOBJ     DS      F  Object handle
HSUB     DS      F  Subscription handle
COMPCODE DS      F  Completion code
REASON   DS      F  Reason code qualifying COMPCODE
```

MQSUBRQ - Richiesta di sottoscrizione

Utilizzare la chiamata MQSUBRQ per effettuare una richiesta per la pubblicazione conservata, quando il sottoscrittore è stato registrato con MQSO_PUBLICATIONS_ON_REQUEST.

Sintassi

MQSUBRQ (*Hconn*, *Hsub*, *Action*, *SubRqOpts*, *Compcode*, *Motivo*)

Parametri

Hconn
Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

In applicazioni z/OS per CICS , la chiamata MQCONN può essere omessa e il seguente valore specificato per *Hconn*:

DEF_MQH_HCONN
Handle di connessione predefinito.

HSub
Tipo: MQHOBJ - input

Questo handle rappresenta la sottoscrizione per cui è necessario richiedere un aggiornamento. Il valore di *Hsub* è stato restituito da una chiamata MQSUB precedente.

Azione
Tipo: MQLONG - input

Questo parametro controlla la particolare azione richiesta sulla sottoscrizione. È necessario specificare il seguente valore:

PUBLICAZIONE MQSR_ACTION_

Questa azione richiede che una pubblicazione di aggiornamento venga inviata per l'argomento specificato. Può essere utilizzato solo se il sottoscrittore (subscriber) ha specificato l'opzione MQSO_PUBLICATIONS_ON_REQUEST sulla chiamata MQSUB quando ha effettuato la sottoscrizione. Se il gestore code dispone di una pubblicazione conservata per l'argomento, questa viene inviata al sottoscrittore. In caso contrario, la chiamata non riesce. Se a un'applicazione viene inviata una pubblicazione che è stata conservata, ciò viene indicato dalla proprietà del messaggio MQIsRetained di tale pubblicazione.

Poiché l'argomento nella sottoscrizione esistente rappresentato dal parametro Hsub può contenere caratteri jolly, il sottoscrittore potrebbe ricevere più pubblicazioni conservate.

Opzioni SubRq
Tipo: MQSRO - input/output

Queste opzioni controllano l'azione di MQSUBRQ, consultare “MQSRO - Opzioni di richieste di sottoscrizione” a pagina 605 per dettagli.

Se non sono richieste opzioni, i programmi scritti in C o nell'assembler S/390 possono specificare un indirizzo di parametro null invece di specificare l'indirizzo di una struttura MQSRO.

CompCode
Tipo: MQLONG - output

Il codice di completamento; è uno dei seguenti:

MQCC_OK
Completamento riuscito

MQCC_AVVERTENZA
Avvertenza (completamento parziale)

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo
Tipo: MQLONG - output

Il codice di errore che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_FUNZIONE_NON_SUPPORTATA
2298 (X'08FA') La funzione richiesta non è disponibile nell'ambiente corrente.

MQRC_NO_RETAINED_MSG
2437 (X'0985 ') Non ci sono pubblicazioni conservate attualmente memorizzate per questo argomento.

ERRORE MQRC_OPTIONS_
2046 (X'07FE') Il campo o il parametro Opzioni contiene opzioni non valide o una combinazione di opzioni non valide.

MQRC_Q_MGR_QUIESCING
2161 (X'0871 ') Gestore code in fase di sospensione.

ERRORE_ERRORE_MQRC
2438 (X'0986 ') Nella chiamata MQSUBRQ, le opzioni MQSRO della richiesta di sottoscrizione non è valido.

MQRC_RETAINED_MSG_Q_ERROR
2525 (X'09DD') Le pubblicazioni conservate che esistono per la stringa di argomenti sottoscritta non possono essere richiamate.

MQRC_RETAINED_NOT_DELIVERED
2526 (X'09DE') Le pubblicazioni conservate che esistono per la stringa di argomenti sottoscritta, non possono essere consegnate alla coda di destinazione della sottoscrizione e non possono essere consegnate alla coda di messaggi non recapitabili.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Note d'utilizzo

Le seguenti note di utilizzo si applicano all'utilizzo del codice azione MQSR_ACTION_PUBLICATION:

1. Se questo comando viene completato correttamente, le pubblicazioni conservate corrispondenti alla sottoscrizione specificata sono state inviate alla sottoscrizione e possono essere ricevute utilizzando MQGET o MQCB utilizzando l'Hobj restituito sul comando MQSUB originale che ha creato la sottoscrizione.
2. Se l'argomento sottoscritto dal verbo MQSUB originale che ha creato la sottoscrizione conteneva un carattere jolly, è possibile inviare più di una pubblicazione conservata. Il numero di pubblicazioni inviate come risultato di questa chiamata viene registrato nel campo NumPubs nella struttura SubRqOpts.

3. Se questo verbo viene completato con un codice motivo MQRC_NO_RETAINED_MSG, non vi erano pubblicazioni attualmente conservate per l'argomento specificato. #
4. Se questo verbo viene completato con un codice motivo di MQRC_RETAINED_MSG_Q_ERROR o MQRC_RETAINED_NOT_DELIVERED, ci sono attualmente pubblicazioni conservate per l'argomento specificato, ma si è verificato un errore che indica che non è stato possibile consegnarle.
5. L'applicazione deve disporre di una sottoscrizione corrente all'argomento prima di poter effettuare questa chiamata. Se la sottoscrizione è stata effettuata in un'istanza precedente dell'applicazione e non è disponibile un handle valido per la sottoscrizione, l'applicazione deve prima richiamare MQSUB con l'opzione MQSO_RESUME per ottenere un handle da utilizzare in questa chiamata.
6. Le pubblicazioni vengono inviate alla destinazione registrata per l'utilizzo con la sottoscrizione corrente di questa applicazione. Se le pubblicazioni devono essere inviate altrove, la sottoscrizione deve essere prima modificata utilizzando la chiamata MQSUB con l'opzione MQSO_ALTER.

Richiamo C

```
MQSUB (Hconn, Hsub, Action, &SubRqOpts, &CompCode, &Reason)
```

Dichiarare i parametri come segue:

```
MQHCONN Hconn;     /* Connection handle */
MQHOBJ  Hsub;      /* Subscription handle */
MQLONG  Action;    /* Action requested by MQSUBRQ */
MQSRO   SubRqOpts; /* Options that control the action of MQSUBRQ */
MQLONG  CompCode;  /* Completion code */
MQLONG  Reason;    /* Reason code qualifying CompCode */
```

Richiamo COBOL

```
CALL 'MQSUBRQ' USING HCONN, HSUB, ACTION, SUBRQOPTS, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
** Connection handle
01 HCONN PIC S9(9) BINARY.
** Subscription handle
01 HSUB PIC S9(9) BINARY.
** Action requested by MQSUBRQ
01 ACTION PIC S9(9) BINARY.
** Options that control the action of MQSUBRQ
01 SUBRQOPTS.
COPY CMQSROV.
** Completion code
01 COMPCODE PIC S9(9) BINARY.
** Reason code qualifying COMPCODE
01 REASON PIC S9(9) BINARY.
```

Chiamata PL/I

```
call MQSUBRQ (Hconn, Hsub, Action, SubRqOpts, CompCode, Reason)
```

Dichiarare i parametri come segue:

```
dcl Hconn fixed bin(31);    /* Connection handle */
dcl Hsub fixed bin(31);     /* Subscription handle */
dcl Action fixed bin(31);   /* Action requested by MQSUBRQ */
dcl SubRqOpts like MQSRO;   /* Options that control the action of MQSUBRQ */
dcl CompCode fixed bin(31); /* Completion code */
dcl Reason fixed bin(31);   /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQSUBRQ,(HCONN, HSUB, ACTION, SUBRQOPTS,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN DS F Connection handle
HSUB DS F Subscription handle
ACTION DS F Action requested by MQSUBRQ
SUBRQOPTS CMQSROA , Options that control the action of MQSUBRQ
COMPCODE DS F Completion code
REASON DS F Reason code qualifying COMPCODE
```

Attributi degli oggetti

Questa raccolta di argomenti elenca solo gli oggetti IBM MQ che possono essere oggetto di una chiamata di funzione MQINQ e fornisce dettagli sugli attributi che è possibile interrogare e sui selettori da utilizzare.

Attributi per il gestore code

Alcuni attributi del gestore code sono corretti per particolari implementazioni; altri possono essere modificati utilizzando il comando MQSC `ALTER QMGR`.

Gli attributi possono essere visualizzati anche utilizzando il comando `DISPLAY QMGR`. La maggior parte degli attributi del gestore code può essere interrogata aprendo un oggetto MQOT_Q_MGR speciale e utilizzando la chiamata MQINQ con l'handle restituito.

La seguente tabella riepiloga gli attributi specifici del gestore code. Gli attributi sono descritti in ordine alfabetico.

Nota: I nomi degli attributi visualizzati in questa sezione sono nomi descrittivi utilizzati con la chiamata MQINQ; i nomi sono gli stessi dei comandi PCF. Quando i comandi MQSC vengono utilizzati per definire, modificare o visualizzare gli attributi, vengono utilizzati nomi brevi alternativi; per ulteriori informazioni, consultare Comandi MQSC .

Tabella 557. Attributi per il gestore code

Attributo	Descrizione
AccountingConnOverride	Sovrascrivere le impostazioni di account.
AccountingInterval	La frequenza di scrittura dei record di contabilità intermedi.
ActivityConnOverride	Sovrascrivi impostazioni attività.
ActivityTrace	Controlla la raccolta della traccia dell'attività dell'applicazione IBM MQ MQI.
AdoptNewMCACheck	Elementi controllati per determinare se adottare un nuovo MCA.
z/OS AdoptNewMCAType	Indica se riavviare automaticamente un'istanza orfana di un MCA di un particolare tipo di canale.
AlterationDate	Data dell'ultima modifica della definizione
AlterationTime	Ora dell'ultima modifica della definizione
AuthorityEvent	Controlla se vengono generati eventi di autorizzazione (non autorizzati)
z/OS BridgeEvent	Attributo di controllo per gli eventi bridge.
ChannelAutoDef	Controlla se la definizione di canale automatica è consentita
ChannelAutoDefEvent	Controlla se vengono generati eventi di definizione automatica del canale
ChannelAutoDefExit	Nome dell'uscita utente per la definizione di canale automatica
ChannelEvent	Attributo di controllo per gli eventi canale.
ChannelInitiatorControl	Attributo di controllo per l'iniziatore di canali
ChannelMonitoring	Dati di monitoraggio online per i canali

Tabella 557. Attributi per il gestore code (Continua)	
Attributo	**Descrizione**
ChannelStatistics	Controlla la raccolta dei dati statistici per canali.
z/OS ChinitAdapters	Numero di attività secondarie dell'adattatore per l'elaborazione delle chiamate IBM MQ .
z/OS ChinitDispatchers	Numero di dispatcher da usare per l'iniziatore di canali.
	Riservato per uso IBM .
z/OS ChinitTraceAutoStart	Indica se la traccia dell'iniziatore di canali deve essere avviata automaticamente.
z/OS ChinitTraceTableSize	Dimensione dello spazio dati di traccia dell'iniziatore del canale.
ClusterSenderMonitoringDefault	Valori predefiniti dei dati di controllo in linea per i canali mittenti del cluster
ClusterSender	Controlla la raccolta delle informazioni di monitoraggio delle statistiche per i canali mittente del cluster.
ClusterWorkloadData	Dati utente per uscita carico di lavoro cluster
ClusterWorkloadExit	Nome dell'uscita utente per la gestione del carico di lavoro del cluster
ClusterWorkloadLength	Lunghezza massima dei dati del messaggio passati all'exit del carico di lavoro cluster
CLWLMRUChannels	Numero di canali utilizzati più di recente per il bilanciamento del carico di lavoro del cluster
CLWLUseQ	Il carico di lavoro del cluster utilizza la coda remota.
CodedCharSetId	CCSID (Coded character set identifier)
CommandEvent	Attributo di controllo per gli eventi comando.
CommandInputattributo QName	Nome coda di input comandi
CommandLevel	Livello di comandi
CommandServerattributo Controllo	Attributo di controllo per il server dei comandi.
Attributo Evento di configurazione	Attributo di controllo per gli eventi di configurazione.
DeadLetterQName	Nome della coda di messaggi non recapitabili
DefClusterXmitQueuecoda trasmissione	Tipo coda di trasmissione cluster predefinito
DefXmitQName	Nome coda di trasmissione predefinita
DistLists	Supporto elenco di distribuzione
z/OS Gruppo DNS	Nome del gruppo per il listener TCP quando si utilizza il supporto DNS (Dynamic Domain Name Services) di Workload Manager.
z/OS DNSWLM	Indica se il listener TCP esegue la registrazione con Workload Manager per DDNS (Dynamic Domain Name Services).
ExpiryInterval	Intervallo tra le scansioni per i messaggi scaduti
IGQPutAuthority	Autorizzazione di inserimento nella coda all'interno del gruppo
IGQUserId	Identificativo utente accodamento all'interno del gruppo
InhibitEvent	Controlla se vengono generati eventi di inibizione (inibisci ricezione e inibisci immissione)
InitialKey 1	La chiave iniziale per il sistema di protezione password.
IPAddressVersion	Versione dell'indirizzo Internet Protocol
z/OS IntraGroupAccodamento	Supporto di accodamento all'interno del gruppo
z/OS ListenerTimer	Intervallo di tempo tra i tentativi di riavviare il listener dopo un errore APPC o TCP/IP.
LocalEvent	Controlla se vengono generati eventi di errori locali
LoggerEvent	Controlla se vengono generati eventi del programma di registrazione
z/OS LUGroupName	Nome LU generico per il listener LU 6.2 che gestisce le trasmissioni in entrata per il gruppo di condivisione code.
z/OS Nome LU	Nome della LU da utilizzare per le trasmissioni LU in uscita 6.2 .

Tabella 557. Attributi per il gestore code (Continua)	
Attributo	**Descrizione**
LU62ARMSuffix	Suffisso di SYS1.PARMLIB membro APPCPMxx, che designa LUADD per questo iniziatore di canali.
z/OS LU62Channels	Numero massimo di canali correnti o di client connessi che utilizzano LU 6.2.
MaxActiveChannels	Numero massimo di canali che possono essere attivi in qualsiasi momento.
MaxChannels	Numero massimo di canali correnti.
MaxHandles	Il numero massimo di puntatori
MaxMsgLength	La lunghezza massima del messaggio in byte
Attributo MaxPriority	Priorità massima
MaxPropertiesLength	Lunghezza massima dei dati delle proprietà in byte
MaxUncommittedMsgs	Numero massimo di messaggi di cui non è stato eseguito il commit in un'unità di lavoro
MQIAccounting	Controlla la raccolta delle informazioni di account per i dati MQI.
MQIStatistics	Controlla la raccolta di informazioni di controllo statistiche per il gestore code.
MsgMarkBrowseInterval	Intervallo dopo il quale il gestore code può rimuovere il contrassegno dai messaggi visualizzati.
z/OS OutboundPortMin	Con *OutboundPortMin*, definisce l'intervallo di numeri di porta da utilizzare durante il bind dei canali in uscita.
z/OS OutboundPortMin	Con *OutboundPortMax*, definisce l'intervallo di numeri di porta da utilizzare durante il bind dei canali in uscita.
PerformanceEvent	Controlla se vengono generati eventi relativi alle prestazioni
Piattaforma	La piattaforma su cui è in esecuzione il gestore code
PubSubNPInputMsg	Se eliminare (o conservare) un messaggio di input non consegnato
PubSubNPResponse	Controlla il comportamento di non consegnate
PubSubMaxMsgRetryCount	Il numero di tentativi durante l'elaborazione (nel punto di sincronizzazione) di un messaggio di comando non riuscito
PubSubSyncPoint	Indica se solo i messaggi persistenti (o tutti) devono essere elaborati nel syncpoint
PubSubMode	Se l'interfaccia di pubblicazione / sottoscrizione in coda è in esecuzione
QMgrDesc	La descrizione del gestore code
QMgrIdentifier	Identificativo univoco generato internamente del gestore code
QMgrName	Nome del gestore code
QSGName	Nome del gruppo di condivisione code
QueueAccounting	Controlla la raccolta delle informazioni di account per le code.
QueueMonitoring	Dati di monitoraggio in linea per le code
QueueStatistics	Controlla la raccolta dei dati statistici per le code.
z/OS ReceiveTimeout	Il tempo di attesa dei dati da parte del canale TCP/IP prima di ritornare allo stato inattivo.
z/OS ReceiveTimeoutMin	Qualificatore per *ReceiveTimeout*.
z/OS ReceiveTimeoutReceiveTimeout	Tempo minimo durante il quale il canale TCP/IP attende i dati prima di tornare allo stato inattivo.
RemoteEvent	Controlla se vengono generati eventi di errore remoti
RepositoryName	Nome del cluster per il quale questo gestore code fornisce i servizi del repository
RepositoryNamelist	Nome dell'oggetto elenco nomi contenente i nomi dei cluster per i quali questo gestore code fornisce i servizi del repository
ScyCase	Caso dei profili di sicurezza
SharedQMgrNome	Nome gestore code condiviso
“SPLCAP” a pagina 854	IBM MQ Protezione di sicurezza dei messaggi avanzata per un gestore code attivato o disattivato.
Elenco nomi SSLCRL [1]	Nome dell'oggetto elenco nomi contenente i nomi degli oggetti delle informazioni di autenticazione.

Tabella 557. Attributi per il gestore code (Continua)	
Attributo	**Descrizione**
SSLCryptoHardware [1]	Stringa di configurazione hardware crittografica.
SSLEvent	Attributo di controllo per gli eventi TLS.
SSLFIPSRequired	Utilizzare solo algoritmi certificati FIPS per la crittografia.
SSLKeyRepository [1]	Ubicazione del repository chiavi TLS.
SSLKeyRepositoryPassword [1]	La password per il repository chiavi TLS.
SSLKeyResetConteggio	Conteggio reimpostazioni chiave TLS.
Attività SSL [1]	Numero di attività secondarie del server per l'elaborazione di chiamate TLS.
StatisticsInterval	Frequenza di scrittura dei dati di monitoraggio delle statistiche.
StartStopEvent	Controlla se vengono generati eventi di avvio e arresto
SyncPoint	Disponibilità Syncpoint
z/OS Canali TCP	Numero massimo di canali correnti o client connessi che utilizzano TCP/IP.
z/OS TCPKeepAlive	Indica se utilizzare TCP KEEPALIVE per controllare l'altra estremità della connessione.
z/OS NomeTCP	Nome del sistema TCP/IP che si sta utilizzando.
z/OS TCPStackType	Come l'iniziatore di canali può utilizzare gli indirizzi TCP/IP.
TraceRouteattributo Registrazione	Controlla la registrazione delle informazioni di traccia - instradamento.
TriggerInterval	Intervallo messaggio trigger
Versione	Versione
XrCapability	Specifica se sono supportati i comandi di telemetria.

Note:

1. Questo attributo non può essere interrogato utilizzando la chiamata MQINQ e non è descritto in questa sezione. Per dettagli su questo attributo, consultare Modifica gestore code .

Attività correlate

Specifica che solo i CipherSpecs certificati FIPS vengono utilizzati al runtime sul client MQI

Riferimenti correlati

FIPS (Federal Information Processing Standards) per AIX, Linux, and Windows

Sovrascrittura AccountingConn(MQLONG)

Ciò consente alle applicazioni di sovrascrivere l'impostazione dei valori ACCTMQI e ACCTQDATA nell'attributo Qmgr.

Il valore è uno dei seguenti:

DISABILITAZIONE_MQMON_
: Le applicazioni non possono sovrascrivere l'impostazione degli attributi ACCTMQI e ACCTQ Qmgr utilizzando il campo Opzioni nella struttura MQCNO sulla chiamata MQCONNX. Questo è il valore predefinito.

MMON_ENABLED
: Le applicazioni possono sovrascrivere gli attributi ACCTQ e ACCTMQI Qmgr utilizzando il campo Opzioni nella struttura MQCNO.

Le modifiche a questo valore sono effettive solo per le connessioni al gestore code dopo la modifica all'attributo.

Questo attributo è supportato solo sulle piattaforme seguenti:

- IBM i IBM i

- Linux AIX AIX and Linux
- Windows Windows

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_ACCOUNTING_CONN_OVERRIDE con la chiamata MQINQ.

AccountingInterval (MQLONG)

Specifica il tempo prima che vengano scritti i record di account intermedi (in secondi).

Il valore è un numero intero compreso tra 0 e 604800, con un valore predefinito di 1800 (30 minuti). Specificare 0 per disattivare i record intermedi.

Questo attributo è supportato solo sulle piattaforme seguenti:

- IBM i IBM i
- Linux AIX AIX and Linux
- Linux Linux
- Windows Windows

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_ACCOUNTING_INTERVAL con la chiamata MQINQ.

Sovrascrittura ActivityConn(MQLONG)

Ciò consente alle applicazioni di sovrascrivere l'impostazione del valore ACTVTRC nell'attributo gestore code.

Il valore è uno dei seguenti:

DISABILITAZIONE_MQMON_
Le applicazioni non possono sovrascrivere l'impostazione dell'attributo del gestore code ACTVTRC utilizzando il campo Opzioni nella struttura MQCNO sulla chiamata MQCONNX. Questo è il valore predefinito.

MMON_ENABLED
Le applicazioni possono sovrascrivere l'attributo ACTVTRC del gestore code utilizzando il campo Opzioni nella struttura MQCNO.

Le modifiche a questo valore sono effettive solo per le connessioni al gestore code dopo la modifica all'attributo.

questo attributo è supportato solo su Multipiattaforme.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_ACTIVITY_CONN_OVERRIDE con la chiamata MQINQ .

ActivityTrace (MQLONG)

Controlla la raccolta della traccia dell'attività dell'applicazione IBM MQ MQI.

Il valore è uno dei seguenti:

MMON_UN
Raccogliere la traccia dell'attività dell'applicazione IBM MQ MQI.

MQMON_DISATTIVO
Non raccogliere la traccia dell'attività dell'applicazione MQI IBM MQ . Questo è il valore predefinito.

Se si imposta l'attributo del gestore code ACTVCONO su ENABLED, questo valore potrebbe essere sovrascritto per le singole connessioni utilizzando il campo Opzioni nella struttura MQCNO.

Le modifiche a questo valore sono effettive solo per le connessioni al gestore code dopo la modifica all'attributo.

questo attributo è supportato solo su Multipiattaforme.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_ACTIVITY_TRACE con la chiamata MQINQ .

AdoptNewMCACheck (MQLONG)

Definisce gli elementi da controllare per stabilire se adottare un MCA quando viene rilevato un nuovo canale in entrata che ha lo stesso nome di un MCA già attivo

Il valore è uno dei seguenti:

MQADOPT_CHECK_Q_MGR_NAME
Verificare il nome del gestore code.

MQADOPT_CHECK_NET_ADDR
Controllare l'indirizzo di rete.

CHECK_MQADOPT_ALL
Controllare il nome del gestore code e l'indirizzo di rete. Se possibile, eseguire questo controllo per proteggere i canali dall'arresto, involontario o doloso. Questo è il valore predefinito.

MQADOPT_CHECK_NONE
Non selezionare alcun elemento.

Le modifiche a questo attributo diventano effettive la volta successiva che un canale tenta di adottare un canale.

z/OS questo attributo è supportato solo su z/OS.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_ADOPTNEWMCA_CHECK con la chiamata MQINQ.

z/OS *AdoptNewMCAType (MQLONG)*

Specifica se riavviare automaticamente un'istanza orfana di un MCA di un particolare tipo di canale quando viene rilevata una nuova richiesta di canale in entrata corrispondente all'attributo MCACheck AdoptNew

È una dei seguenti valori:

MQADOPT_TYPE_NO
L'adozione di istanze di canale orfane non è richiesta. Questo è il valore predefinito.

TIPO_MQADOPT_ALL
Adottare tutti i tipi di canale.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_ADOPTNEWMCA_TYPE con la chiamata MQINQ.

AlterationDate (MQCHAR12)

Questa è la data dell'ultima modifica della definizione. Il formato della data è YYYY-MM-DD, riempito con due spazi finali per rendere la lunghezza di 12 byte.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_ALTERATION_DATE con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Questa è l'ora dell'ultima modifica della definizione. Il formato dell'ora è HH.MM.SS.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_ALTERATION_TIME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_TIME_LENGTH.

AuthorityEvent (MQLONG)

Controlla se vengono generati eventi di autorizzazione (non autorizzati). È una dei seguenti valori:

DISABILITAZIONE_MQEVR
Report eventi disabilitato.

MQEVR_ENABLED
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_AUTHORITY_EVENT con la chiamata MQINQ.

z/OS *BridgeEvent (MQLONG)*

Specifica se vengono generati eventi bridge IMS .

Il valore è uno dei seguenti:

MQEVR_ENABLED
Generare gli eventi bridge IMS , come segue:

MQRC_BRIDGE_STARTED
MQRC_BRIDGE_STOPPED

DISABILITAZIONE_MQEVR
Non generare eventi bridge IMS ; questo è il valore predefinito.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_BRIDGE_EVENT con la chiamata MQINQ.

Definizione ChannelAuto(MQLONG)

Questo attributi controlla le definizioni automatiche dei canali di tipo MQCHT_RECEIVER e MQCHT_SVRCONN. La definizione automatica dei canali MQCHT_CLUSSDR è sempre abilitata. Il valore è uno dei seguenti:

DISABLE_MQCHAD
Definizione automatica del canale disabilitata.

ENABLE_MQCHAD
Definizione automatica canale abilitata.

Multi questo attributo è supportato solo su Multipiattaforme.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CHANNEL_AUTO_DEF con la chiamata MQINQ.

ChannelAutoDefEvent (MQLONG)

Controlla se vengono generati gli eventi di definizione automatica del canale. Si applica a canali di tipo MQCHT_RECEIVER, MQCHT_SVRCONN e MQCHT_CLUSSDR. Il valore è uno dei seguenti:

DISABILITAZIONE_MQEVR
Report eventi disabilitato.

MQEVR_ENABLED
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Multi questo attributo è supportato solo su Multipiattaforme.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CHANNEL_AUTO_DEF_EVENT con la chiamata MQINQ.

ChannelAutoDefExit (MQCHARn)

Questo è il nome dell'uscita utente per la definizione di canale automatica. Se questo nome non è vuoto e *ChannelAutoDef* ha il valore MQCHAD_ENABLED, l'uscita viene richiamata ogni volta che il gestore code sta per creare una definizione di canale. Ciò si applica ai canali di tipo MQCHT_RECEIVER, MQCHT_SVRCONN e MQCHT_CLUSSDR. L'uscita può quindi effettuare una delle seguenti operazioni:

- Creare la definizione di canale senza modificare.
- Modificare gli attributi della definizione di canale creata.
- Sopprimere completamente la creazione del canale.

Nota: La lunghezza e il valore di questo attributo sono specifici dell'ambiente. Consultare l'introduzione alla struttura MQCD in "MQCD - Definizione canale" a pagina 1521 per i dettagli del valore di questo attributo in vari ambienti.

z/OS Su z/OS, questo attributo si applica solo ai canali mittente cluster e ricevente cluster.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_CHANNEL_AUTO_DEF_EXIT con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_EXIT_NAME_LENGTH.

ChannelEvent (MQLONG)

Specifica se vengono generati eventi del canale.

È una dei seguenti valori:

MQEVR_ECCEZIONE
Generare solo i seguenti eventi di canale:

- MQRC_CHANNEL_ACTIVATED
- MQRC_CHANNEL_CONV_ERROR
- MQRC_CHANNEL_NOT_ACTIVATED
- MQRC_CHANNEL_STOPPED con i seguenti ReasonQualifiers:

  MQRQ_CHANNEL_STOPPED_ERROR
  MQRQ_CHANNEL_STOPPED_RETRY
  MQRQ_CHANNEL_STOPPED_DISABLED

  MQRC_CHANNEL_STOPPED_BY_USER

MQEVR_ENABLED
Generare tutti gli eventi canale. Ovvero, oltre a quelli generati da EXCEPTION, generano i seguenti eventi di canale:

- MQRC_CHANNEL_STARTED
- MQRC_CHANNEL_STOPPED con il seguente ReasonQualifier:

  MQRQ_CHANNEL_STOPPED_OK

DISABILITAZIONE_MQEVR
Non generare eventi di canale; questo è il valore predefinito.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CHANNEL_EVENT con la chiamata MQINQ.

Controllo ChannelInitiator(MQLONG)

Specifica se l'iniziatore di canali deve essere avviato all'avvio del gestore code.

È una dei seguenti valori:

MQSVC_CONTROL_MANUAL
L'iniziatore del canale non deve essere avviato automaticamente.

MQSVC_CONTROL_Q_MGR
L'iniziatore di canali deve essere avviato automaticamente all'avvio del gestore code.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CHINIT_CONTROL con la chiamata MQINQ.

ChannelMonitoring (MQLONG)
Questo attributo specifica i dati di monitoraggio in linea per i canali.

Il valore è uno dei seguenti:

MQMON_NONE
Disabilitare la raccolta dati per il controllo del canale per tutti i canali indipendentemente dall'impostazione dell'attributo del canale MONCHL. Questo è il valore predefinito.

MQMON_DISATTIVO
Disattivare la raccolta dati di monitoraggio per i canali che specificano QMGR nell'attributo del canale MONCHL.

MMON_LOW
Attivare la raccolta dati di controllo con un rapporto basso di raccolta dati per i canali che specificano QMGR nell'attributo del canale MONCHL.

MQMON_MEDIO
Attivare la raccolta dati di monitoraggio con un rapporto moderato di raccolta dati per i canali che specificano QMGR nell'attributo del canale MONCHL.

MQMON_HIGH
Attivare la raccolta dei dati di monitoraggio con un rapporto elevato di raccolta dati per i canali che specificano QMGR nell'attributo del canale MONCHL.

z/OS Su z/OS sistemi, abilitando questo parametro si attiva semplicemente la raccolta dei dati statistici, indipendentemente dal valore selezionato. La specifica di `LOW`, `MEDIUM` o `HIGH` non comporta alcuna differenza sui risultati.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_MONITORING_CHANNEL con la chiamata MQINQ.

ChannelStatistics (MQLONG)

Controlla la raccolta dei dati statistici per canali.

Il valore è uno dei seguenti:

MQMON_NONE
Disabilitare la raccolta dati per le statistiche del canale per tutti i canali indipendentemente dall'impostazione dell'attributo del canale STATCHL. Questo è il valore predefinito.

MQMON_DISATTIVO
Disattivare la raccolta dei dati statistici per i canali che specificano QMGR nell'attributo del canale STATCHL.

MMON_LOW
Attivare la raccolta dati delle statistiche con un rapporto basso di raccolta dati per i canali che specificano QMGR nell'attributo del canale STATCHL.

MQMON_MEDIO
Attivare la raccolta dati statistici con un rapporto moderato di raccolta dati per i canali che specificano QMGR nell'attributo del canale STATCHL.

MQMON_HIGH
Attivare la raccolta dati delle statistiche con un rapporto elevato di raccolta dati per i canali che specificano QMGR nell'attributo del canale STATCHL.

Per la maggior parte dei sistemi si consiglia di utilizzare MEDIUM. Tuttavia, per un canale che elabora un volume elevato di messaggi ogni secondo, è possibile ridurre il livello di campionamento selezionando

LOW. Inoltre, per un canale che elabora solo pochi messaggi e per cui sono importanti le informazioni più aggiornate, è possibile selezionare ALTO.

z/OS Su z/OS sistemi, abilitando questo parametro si attiva semplicemente la raccolta dei dati statistici, indipendentemente dal valore selezionato. La specifica di LOW, MEDIUM o HIGH non comporta alcuna differenza sui risultati.Questo parametro deve essere abilitato al fine di raccogliere i record di contabilità di canale.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_STATISTICS_CHALLEGATO con la chiamata MQINQ.

z/OS *ChinitAdapters (MQLONG)*

Si tratta del numero di attività secondarie dell'adattatore da utilizzare per elaborare le chiamate IBM MQ . Il valore deve essere compreso tra 0 e 9999, con un valore predefinito di 8.

Il rapporto tra adattatori e dispatcher (l'attributo ChinitDispatchers) deve essere compreso tra 8 e 5. Tuttavia, se si dispone solo di pochi canali, non è necessario diminuire il valore di questo parametro dal valore predefinito. È possibile utilizzare i seguenti valori: per un sistema di test, 8 (predefinito); per un sistema di produzione, 20. Idealmente, è necessario disporre di 20 adattatori, che forniscono un maggiore parallelismo di chiamate IBM MQ . Ciò è di notevole importanza per i messaggi permanenti. Un numero inferiore di adattatori potrebbe essere migliore per i messaggi non persistenti.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il programma di selezione MQIA_CHINIT_ADAPTERS con la chiamata MQINQ.

z/OS *ChinitDispatchers (MQLONG)*

Questo è il numero di dispatcher da utilizzare per l'iniziatore di canali. Il valore deve essere compreso tra 0 e 9999, con un valore predefinito di 5.

Come linea guida, consentire un dispatcher per 50 canali correnti. Tuttavia, se si dispone solo di pochi canali, non è necessario diminuire il valore di questo attributo dal valore predefinito. Se si utilizza TCP/IP, il maggior numero di dispatcher utilizzati per i canali TCP/IP è 100, anche se si specifica un valore maggiore. È possibile utilizzare le seguenti impostazioni: sistemi di test, 5 (impostazione predefinita); sistemi di produzione, 20 (sono necessari 20 dispatcher per gestire fino a 1000 canali attivi).

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CHINIT_DISPATCHER con la chiamata MQINQ.

z/OS *ChinitTraceAutoStart (MQLONG)*

Specifica se avviare automaticamente la traccia dell'iniziatore di canali.

Il valore è uno dei seguenti:

MQTRAXSTR_SÌ
Avviare la traccia dell'iniziatore di canali automaticamente. Questo è il valore predefinito.

MQTRAXSTR_NO
Non avviare automaticamente la traccia dell'iniziatore di canali.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CHINIT_TRACE_AUTO_START con la chiamata MQINQ.

z/OS *ChinitTraceTableSize (MQLONG)*

Questa è la dimensione dello spazio dati di traccia dell'iniziatore del canale (in MB).

Il valore deve essere compreso nell'intervallo tra 0 e 2048, con un valore predefinito di 2.

Nota: Ogni volta che usi largez/OS spazi dati, assicurarsi di disporre di spazio di archiviazione ausiliario sufficiente sul sistema per supportare qualsiasi spazio correlatoz/OS attività di cercapersone. È possibile anche aumentare la dimensione dei data set SYS1.DUMP.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CHINIT_TRACE_TABLE_SIZE con la chiamata MQINQ.

ClusterSenderMonitoringDefault (MQLONG)

Specifica il valore da sostituire per l'attributo ChannelMonitoring dei canali mittenti del cluster definiti automaticamente.

Il valore è uno dei seguenti:

MGR MQMON_Q
La raccolta dei dati di controllo online viene ereditata dall'impostazione dell'attributo **ChannelMonitoring** del gestore code. Questo è il valore predefinito.

MQMON_DISATTIVO
Il monitoraggio per il canale è disabilitato

MMON_LOW
A meno che *ChannelMonitoring* non sia MQMON_NONE, il controllo è abilitato con una bassa frequenza di raccolta dati con un effetto minimo sulle prestazioni del sistema. È probabile che i dati raccolti non siano i più aggiornati.

MQMON_MEDIO
A meno che *ChannelMonitoring* non sia MQMON_NONE, il monitoraggio è abilitato con una velocità moderata di raccolta dati con un effetto limitato sulle prestazioni del sistema.

MQMON_HIGH
A meno che *ChannelMonitoring* non sia MQMON_NONE, il monitoraggio è abilitato con una frequenza elevata di raccolta dati con un probabile effetto sulle prestazioni del sistema. I dati raccolti sono i più attuali disponibili.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_MONITORING_AUTO_CLUSSDR con il richiamo MQINQ.

Statistiche ClusterSender(MQLONG)

Poiché i canali mittente del cluster possono essere definiti automaticamente dalla definizione di CLUSRCVR nel repository, non è possibile modificare l'impostazione dell'attributo STATCHL per questi canali mittente del cluster definiti automaticamente utilizzando il canale ALTER. Per questi canali, la decisione di raccogliere o meno i dati di monitoraggio in linea si basa sull'impostazione di questo attributo del gestore code.

Il valore è uno dei seguenti:

MGR MQMON_Q
La raccolta dei dati statistici per i canali mittenti del cluster definiti automaticamente si basa sul valore dell'attributo del gestore code STATCHL. Questo è il valore predefinito.

MQMON_DISATTIVO
Disattivare la raccolta dei dati statistici per i canali mittenti del cluster definiti automaticamente.

MMON_LOW
Abilitare la raccolta dei dati statistici per i canali mittenti del cluster definiti automaticamente con un rapporto basso di raccolta dati.

MQMON_MEDIO
Abilitare la raccolta dei dati statistici per i canali mittenti del cluster definiti automaticamente con un rapporto moderato di raccolta dati.

MQMON_HIGH
Abilitare la raccolta dei dati statistici per i canali mittenti del cluster definiti automaticamente con un rapporto elevato di raccolta dati.

Per la maggior parte dei sistemi si consiglia MEDIUM. Tuttavia, per un canale mittente del cluster definito automaticamente che elabora un elevato volume di messaggi ogni secondo, è possibile ridurre il livello di campionamento selezionando LOW. Inoltre, per un canale che elabora solo pochi messaggi e per cui sono importanti le informazioni più aggiornate, è possibile selezionare ALTO.

z/OS Su z/OS sistemi, abilitando questo parametro si attiva semplicemente la raccolta dei dati statistici, indipendentemente dal valore selezionato. La specifica di LOW, MEDIUM o HIGH non comporta alcuna differenza sui risultati.Questo parametro deve essere abilitato al fine di raccogliere i record di contabilità di canale.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_STATISTICS_AUTO_CLUSSDR con la chiamata MQINQ.

Dati ClusterWorkload(MQCHAR32)

Si tratta di una stringa di caratteri a 32 byte definita dall'utente che viene passata all'uscita del workload del cluster quando viene richiamata. Se non ci sono dati da passare all'uscita, la stringa è vuota.

Per individuare il valore di questo attributo, utilizzare il selettore MQCA_CLUSTER_WORKLOAD_DATA con la chiamata MQINQ.

Uscita ClusterWorkload(MQCHARn)

Questo è il nome dell'uscita utente per la gestione del workload del cluster. Se questo nome non è vuoto, l'uscita viene richiamata ogni volta che un messaggio viene inserito in una coda cluster o spostato da una coda mittente cluster a un'altra. L'uscita può quindi accettare l'istanza della coda selezionata dal gestore code come destinazione del messaggio oppure selezionare un'altra istanza della coda.

Nota: La lunghezza e il valore di questo attributo sono specifici dell'ambiente.

Per stabilire il valore di questo attributo, utilizzare il selettore MQCA_CLUSTER_WORKLOAD_EXIT con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_EXIT_NAME_LENGTH.

Lunghezza ClusterWorkload(MQLONG)

Questa è la lunghezza massima dei dati del messaggio passati all'uscita del workload del cluster. La lunghezza effettiva dei dati passati all'uscita è il minimo dei seguenti:

- La lunghezza del messaggio.
- L'attributo **MaxMsgLength** del gestore code.
- L'attributo **ClusterWorkloadLength** .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CLUSTER_WORKLOAD_LENGTH con la chiamata MQINQ.

CLWLMRUChannel (MQLONG)

Specifica il numero massimo di canali cluster utilizzati più di recente, da considerare per l'utilizzo da parte dell'algoritmo di scelta del carico di lavoro del cluster.

Questo è un valore compreso tra 1 e 999999999.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CLWL_MRU_CHANNELS con la chiamata MQINQ.

CLWLUseQ (MQLONG)

Specifica se utilizzare le code remote per il carico di lavoro del cluster.

Il valore è uno dei seguenti:

MQCLWL_USEQ_ANY
Utilizzare sia code locali che remote.

MQCLWL_USEQ_LOCALE
Non utilizzare code remote. Questo è il valore predefinito.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CLWL_USEQ con la chiamata MQINQ.

CodedCharSetId (MQLONG)

Definisce la serie di caratteri utilizzata dal gestore code per tutti i campi stringa di caratteri definiti nell'MQI, ad esempio i nomi degli oggetti e la data e ora di creazione della coda. La serie di caratteri deve essere una serie di caratteri a byte singolo per i caratteri validi nei nomi oggetto. Non si applica ai dati dell'applicazione trasmessi nel messaggio. Il valore dipende dall'ambiente:

- Su z/OS, il valore viene impostato dai parametri di sistema all'avvio del gestore code; il valore predefinito è 500.
- In Windows, il valore è il CODEPAGE primario dell'utente che crea il gestore code.
- Su IBM i, il valore è quello impostato nell'ambiente quando il gestore code viene creato per la prima volta.
- Su AIX and Linux, il valore è il CODESET predefinito per la locale dell'utente che crea il gestore code.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CODED_CHAR_SET_ID con la chiamata MQINQ.

CommandEvent (MQLONG)

Specifica se vengono generati eventi di comando, come riportato di seguito:

DISABILITAZIONE_MQEVR
Non generare eventi comando. Questa è l'opzione predefinita.

MQEVR_ENABLED
Generare eventi di comando.

MQEVR_NO_DISPLAY
Gli eventi di comando vengono generati per tutti i comandi riusciti diversi da MQINQ.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_COMMAND_EVENT con la chiamata MQINQ.

CommandInputQName (MQCHAR48)

È il nome della coda di input del comando definita sul gestore code locale. Si tratta di una coda a cui gli utenti possono inviare comandi, se autorizzati. Il nome della coda dipende dall'ambiente:

- Su z/OS, il nome della coda è SYSTEM.COMMAND.INPUT; i comandi MQSC e PCF possono essere inviati. Consultare Comandi MQSC per i dettagli dei comandi MQSC e Definizioni dei formati dei comandi programmabili per dettagli dei comandi PCF.
- In tutti gli ambienti, il nome della coda è SYSTEM.ADMIN.COMMAND.QUEUEe solo i comandi PCF possono essere inviati. Tuttavia, un comando MQSC può essere inviato a questa coda se il comando MQSC è racchiuso all'interno di un comando PCF di tipo MQCMD_ESCAPE. Consultare Escape per informazioni sul comando Escape.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_COMMAND_INPUT_Q_NAME con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da MQ_Q_NAME_LENGTH.

CommandLevel (MQLONG)

Nota: Il supporto per il sistema operativo HP-UX per tutti i componenti IBM MQ , inclusi server e client, è stato rimosso in IBM MQ 9.1.

Indica il livello dei comandi di controllo del sistema supportati dal gestore code. Può essere uno dei seguenti valori:

MQCMDL_LEVEL_800
Livello 800 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 8.0
- IBM MQ for IBM i 8.0
- IBM MQ for Linux 8.0
- IBM MQ for Windows 8.0
- IBM MQ for z/OS 8.0

MQCMDL_LEVEL_801
Livello 801 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 8.0.0 Fix Pack 2
- IBM MQ for HP-UX 8.0.0 Fix Pack 2
- IBM MQ for IBM i 8.0.0 Fix Pack 2
- IBM MQ for Linux 8.0.0 Fix Pack 2

MQCMDL_LEVEL_802
Livello 802 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 8.0.0 Fix Pack 3
- IBM MQ for IBM i 8.0.0 Fix Pack 3
- IBM MQ for Linux 8.0.0 Fix Pack 3
- IBM MQ for Windows 8.0.0 Fix Pack 3

MQCMDL_LEVEL_900
Livello 900 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.0
- IBM MQ for IBM i 9.0
- IBM MQ for Linux 9.0
- IBM MQ for Windows 9.0
- IBM MQ for z/OS 9.0

MQCMDL_LEVEL_901
Livello 901 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for Linux 9.0.1
- IBM MQ for Windows 9.0.1
- IBM MQ for z/OS 9.0.1

MQCMDL_LEVEL_902
Livello 902 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for Linux 9.0.2
- IBM MQ for Windows 9.0.2

- IBM MQ for z/OS 9.0.2

MQCMDL_LEVEL_903
Livello 903 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for Linux 9.0.3
- IBM MQ for Windows 9.0.3
- IBM MQ for z/OS 9.0.3

MQCMDL_LEVEL_904
Livello 904 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.0.4
- IBM MQ for Linux 9.0.4
- IBM MQ for Windows 9.0.4
- IBM MQ for z/OS 9.0.4

MQCMDL_LEVEL_905
Livello 905 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.0.5
- IBM MQ for Linux 9.0.5
- IBM MQ for Windows 9.0.5
- IBM MQ for z/OS 9.0.5

MQCMDL_LEVEL_910
Livello 910 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.1
- IBM MQ for IBM i 9.1
- IBM MQ for Linux 9.1
- IBM MQ for Windows 9.1
- IBM MQ for z/OS 9.1

MQCMDL_LEVEL_911
Livello 911 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.1.1
- IBM MQ for Linux 9.1.1
- IBM MQ for Windows 9.1.1
- IBM MQ for z/OS 9.1.1

MQCMDL_LEVEL_912
Livello 912 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.1.2
- IBM MQ for Linux 9.1.2
- IBM MQ for Windows 9.1.2
- IBM MQ for z/OS 9.1.2

MQCMDL_LEVEL_913
Livello 913 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.1.3
- IBM MQ for Linux 9.1.3
- IBM MQ for Windows 9.1.3
- IBM MQ for z/OS 9.1.3

MQCMDL_LEVEL_914
Livello 914 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.1.4
- IBM MQ for Linux 9.1.4
- IBM MQ for Windows 9.1.4
- IBM MQ for z/OS 9.1.4

MQCMDL_LEVEL_915
Livello 915 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.1.5
- IBM MQ for Linux 9.1.5
- IBM MQ for Windows 9.1.5
- IBM MQ for z/OS 9.1.5

MQCMDL_LEVEL_910
Livello 910 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.1
- IBM MQ for IBM i 9.1
- IBM MQ for Linux 9.1
- IBM MQ for Windows 9.1
- IBM MQ for z/OS 9.1

MQCMDL_LEVEL_920
Livello 920 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.2
- IBM MQ for IBM i 9.2
- IBM MQ for Linux 9.2
- IBM MQ for Windows 9.2
- IBM MQ for z/OS 9.2

MQCMDL_LEVEL_921
Livello 921 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.2.1
- IBM MQ for Linux 9.2.1
- IBM MQ for Windows 9.2.1

- IBM MQ for z/OS 9.2.1

MQCMDL_LEVEL_922
Livello 922 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.2.2
- IBM MQ for Linux 9.2.2
- IBM MQ for Windows 9.2.2
- IBM MQ for z/OS 9.2.2

MQCMDL_LEVEL_923
Livello 923 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.2.3
- IBM MQ for Linux 9.2.3
- IBM MQ for Windows 9.2.3
- IBM MQ for z/OS 9.2.3

MQCMDL_LEVEL_924
Livello 924 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.2.4
- IBM MQ for Linux 9.2.4
- IBM MQ for Windows 9.2.4
- IBM MQ for z/OS 9.2.4

MQCMDL_LEVEL_925
Livello 925 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.2.5
- IBM MQ for Linux 9.2.5
- IBM MQ for Windows 9.2.5
- IBM MQ for z/OS 9.2.5

MQCMDL_LEVEL_930
Livello 930 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.3
- IBM MQ for IBM i 9.3
- IBM MQ for Linux 9.3
- IBM MQ for Windows 9.3
- IBM MQ for z/OS 9.3

MQCMDL_LEVEL_931
Livello 931 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.3.1
- IBM MQ for Linux 9.3.1
- IBM MQ for Windows 9.3.1

- IBM MQ for z/OS 9.3.1

MQCMDL_LEVEL_932
Livello 932 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.3.2
- IBM MQ for Linux 9.3.2
- IBM MQ for Windows 9.3.2
- IBM MQ for z/OS 9.3.2

MQCMDL_LEVEL_933
Livello 933 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.3.3
- IBM MQ for Linux 9.3.3
- IBM MQ for Windows 9.3.3
- IBM MQ for z/OS 9.3.3

MQCMDL_LEVEL_934
Livello 934 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.3.4
- IBM MQ for Linux 9.3.4
- IBM MQ for Windows 9.3.4
- IBM MQ for z/OS 9.3.4

MQCMDL_LEVEL_935
Livello 935 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.3.5
- IBM MQ for Linux 9.3.5
- IBM MQ for Windows 9.3.5
- IBM MQ for z/OS 9.3.5

MQCMDL_LEVEL_940
Livello 940 dei comandi di controllo del sistema.

Questo valore viene restituito dalle versioni seguenti:

- IBM MQ for AIX 9.4.0
- IBM MQ for Linux 9.4.0
- IBM MQ for Windows 9.4.0
- IBM MQ for z/OS 9.4.0

La serie di comandi di controllo del sistema che corrisponde ad un determinato valore dell'attributo **CommandLevel** varia in base al valore dell'attributo **Platform** ; entrambi devono essere utilizzati per decidere quali comandi di controllo del sistema sono supportati.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_COMMAND_LEVEL con la chiamata MQINQ .

Controllo CommandServer(MQLONG)
Specifica se il server dei comandi deve essere avviato all'avvio del gestore code.

Il valore può essere uno dei seguenti:

MQSVC_CONTROL_MANUAL
Il server dei comandi non deve essere avviato automaticamente.

MQSVC_CONTROL_Q_MGR
Il server dei comandi deve essere avviato automaticamente all'avvio del gestore code.

Questo attributo non è supportato su z/OS.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CMD_SERVER_CONTROL con la chiamata MQINQ.

ConfigurationEvent (MQLONG)

Controlla se vengono generati eventi di configurazione.

Per determinare il valore di questo attributo, utilizzare il selettore MKIA_CONFIGURATION_EVENT con la chiamata MQINQ.

Il valore può essere uno dei seguenti:

DISABILITAZIONE_MQEVR
Report eventi disabilitato.

MQEVR_ENABLED
Segnalazione eventi abilitata.

Multi *Dimensione CurrentQFile(MQLONG)*

La dimensione corrente del file di coda in megabyte, arrotondata al megabyte più vicino.

Tabella 558. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Il valore per questo attributo di stato della coda è la dimensione della coda, arrotondata al megabyte più vicino. Per una nuova coda con attributi predefiniti, il valore di **CurrentQFileSize** è 1.

Il valore massimo di questo attributo è 99,999,9999 MB e per questo attributo non esiste alcun valore predefinito.

Multi *CurrentMaxQFileSize (MQLONG)*

La dimensione massima corrente che il file della coda può raggiungere, arrotondata al megabyte più vicino, data la dimensione del blocco corrente in uso su una coda.

Tabella 559. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

L'uso di questo campo è duplice:

- Se si imposta **MaxQFileSize** sul valore predefinito per la dimensione del blocco corrente, **CurrentMaxQFileSize** mostra il valore effettivo a cui equivale il valore predefinito.
- Se **CurrentMaxQFileSize** non corrisponde a **MaxQFileSize**, è necessario svuotare la coda per adottare una granularità maggiore.

Nota: Consultare Modifica dei file della coda IBM MQ per ulteriori informazioni sulla modifica della dimensione dei file della coda e della granularità e della dimensione del blocco.

Il valore massimo di questo attributo è 99,999,9999 MB e non esiste alcun valore predefinito. Il valore è qualsiasi sia il valore massimo attualmente impostato; per una nuova coda con gli attributi predefiniti, il valore di **CurrentMaxQFileSize** è 2.088.960 MB.

QName DeadLetter(MQCHAR48)

Questo è il nome di una coda definita sul gestore code locale come coda di messaggi non recapitabili. I messaggi vengono inviati a questa coda se non possono essere instradati alla destinazione corretta.

Ad esempio, i messaggi vengono inseriti in questa coda quando:

- Un messaggio arriva a un gestore code, destinato a una coda non ancora definita su tale gestore code
- Un messaggio arriva a un gestore code, ma la coda a cui è destinato non può riceverlo perché:
  - La coda è piena
  - Le richieste di inserimento sono inibite
  - Il nodo di invio non dispone dell'autorità per inserire i messaggi nella coda

Le applicazioni possono anche inserire messaggi nella coda di messaggi non recapitabili.

I messaggi di report vengono trattati allo stesso modo dei messaggi ordinari; se il messaggio di report non può essere consegnato alla relativa coda di destinazione (di solito la coda specificata dal campo *ReplyToQ* nel descrittore del messaggio originale), il messaggio di report viene inserito nella coda dei messaggi non recapitabili (messaggio non recapitato).

Nota: Messaggi che hanno superato la scadenza (consultare MQMD - Campo Scadenza) non vengono trasferiti a questa coda quando vengono eliminati. Tuttavia, un messaggio di report di scadenza (MQRO_EXPIRATION) viene ancora generato e inviato alla coda *ReplyToQ* , se richiesto dall'applicazione mittente.

I messaggi non vengono inseriti nella coda dei messaggi non recapitabili (messaggio non recapitato) quando l'applicazione che ha emesso la richiesta di inserimento ha ricevuto una notifica sincrona del problema mediante il codice di errore restituito dalla chiamata MQPUT o MQPUT1 (ad esempio, un messaggio inserito in una coda locale per cui le richieste di inserimento sono inibite).

I messaggi sulla coda dei messaggi non recapitabili (messaggi non recapitati) a volte hanno come prefisso i dati del messaggio dell'applicazione con una struttura MQDLH. Questa struttura contiene informazioni aggiuntive che indicano il motivo per cui il messaggio è stato inserito nella coda dei messaggi non recapitabili (non recapitati). Per ulteriori dettagli su questa struttura, consultare “MQDLH - Intestazione lettera non instradabile” a pagina 359 .

Questa coda deve essere una coda locale, con un attributo **Usage** di MQUS_NORMAL.

Se un gestore code non supporta una coda di messaggi non recapitabili (messaggi non recapitabili) o non ne è stata definita una, il nome è vuoto. Tutti i gestori code IBM MQ supportano una coda di messaggi non recapitabili (messaggi non recapitabili), ma per impostazione predefinita non è definita.

Se la coda dei messaggi non recapitabili (messaggi non recapitati) non è definita, piena o inutilizzabile per qualche altro motivo, un messaggio che sarebbe stato trasferito ad essa da un agente del canale dei messaggi viene conservato invece nella coda di trasmissione.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_DEAD_LETTER_Q_NAME con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da MQ_Q_NAME_LENGTH.

DefClusterXmitQueue(MQLONG)

L'attributo `DefClusterXmitQueue` controlla la coda di trasmissione selezionata per impostazione predefinita dai canali mittenti del cluster da cui richiamare i messaggi, per inviare i messaggi ai canali riceventi del cluster.

I valori di **DefClusterXmitQueueType** sono `MQCLXQ_SCTQ` o `MQCLXQ_CHANNEL`.

MQCLXQ_SCTQ

Tutti i canali mittenti del cluster inviano messaggi da SYSTEM.CLUSTER.TRANSMIT.QUEUE. Il correlID dei messaggi inseriti nella coda di trasmissione identifica a quale canale mittente del cluster è destinato il messaggio.

SCTQ viene impostato quando viene definito un gestore code.

MQCLXQ_CHANNEL

Ogni canale mittente del cluster invia messaggi da una coda di trasmissione differente. Ciascuna coda di trasmissione viene creata come una coda dinamica permanente dalla coda modello SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE.

Se l'attributo del gestore code, DefClusterXmitQueueTipo, è impostato su CHANNEL, la configurazione predefinita viene modificata nei canali mittenti del cluster che vengono associati alle singole code di trasmissione cluster. Le code di trasmissione sono code dinamiche permanenti create a partire dalla coda modello SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE. Ciascuna coda di trasmissione è associata a un canale mittente del cluster. Poiché un canale mittente del cluster serve una coda di trasmissione, la coda di trasmissione contiene messaggi per un solo gestore code in un cluster. È possibile configurare i cluster in modo che ogni gestore code in un cluster contenga una sola coda cluster. In questo caso, il traffico di messaggi da un gestore code a ogni coda del cluster viene trasferito separatamente dai messaggi alle altre code.

Per interrogare il valore, richiamare MQINQo inviare un comando PCF MQCMD_INQUIRE_Q_MGR(Inquire Queue Manager), impostando il selettore MQIA_DEF_CLUSTER_XMIT_Q_TYPE . Per modificare il valore, inviare un comando PCF del gestore code di modifica (MQCMD_CHANGE_Q_MGR), impostando il selettore MQIA_DEF_CLUSTER_XMIT_Q_TYPE .

Riferimenti correlati

Modifica gestore code

Interrogazione gestore code

“MQINQ - Richiedi attributi oggetto” a pagina 723

La chiamata MQINQ restituisce un array di numeri interi e una serie di stringhe di carattere contenenti attributi di un oggetto.

DefXmitQName (MQCHAR48)

Questo è il nome della coda di trasmissione utilizzata per la trasmissione dei messaggi ai gestori code remoti, se non vi è alcuna altra indicazione di quale coda di trasmissione utilizzare.

Se non esiste una coda di trasmissione predefinita, il nome è completamente vuoto. Il valore iniziale di questo attributo è vuoto.

Per stabilire il valore di questo attributo, utilizzare il selettore MQCA_DEF_XMIT_Q_NAME con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da MQ_Q_NAME_LENGTH.

DistLists (MQLONG)

Ciò indica se il gestore code locale supporta gli elenchi di distribuzione sulle chiamate MQPUT e MQPUT1 . È una dei seguenti valori:

MQDL_SUPPORTED

Elenchi di distribuzione supportati.

MQDL_NOT_SUPPORTED

Elenchi di distribuzione non supportati.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_DIST_LISTS con la chiamata MQINQ.

z/OS *Gruppo DNS(MQCHAR18)*

Questo parametro non è più utilizzato. Consultare Cosa è stato modificato in IBM MQ 8.0.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_DNS_GROUP con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_DNS_GROUP_NAME_LENGTH.

z/OS *DNSWLM (MQLONG)*

Questo parametro non è più utilizzato. Consultare Cosa è stato modificato in IBM MQ 8.0.

Il valore è uno dei seguenti:

SÌ MQDNSWLM
Questo valore può essere visualizzato su un gestore code migrato da una release precedente. Il valore viene ignorato.

MQDNSWLM_NO
Questo è l'unico valore supportato dal gestore code.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_DNS_WLM con la chiamata MQINQ.

ExpiryInterval (MQLONG)

Indica la frequenza con cui il gestore code esegue la scansione delle code alla ricerca di messaggi scaduti. Si tratta di un intervallo di tempo in secondi compreso tra 1 e 99 999 999 o del seguente valore speciale:

MQEXPI_OFF
Il gestore code non esegue la scansione delle code alla ricerca di messaggi scaduti.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_EXPIRY_INTERVAL con la chiamata MQINQ.

z/OS questo attributo è supportato solo su z/OS.

IGQPutAuthority (MQLONG)

Questo attributo si applica solo se il gestore code locale è un membro di un gruppo di condivisione code. Indica il tipo di controllo dell'autorizzazione eseguito quando l'agent di accodamento interno al gruppo locale (agent IGQ) rimuove un messaggio dalla coda di trasmissione condivisa e inserisce il messaggio in una coda locale. Il valore è uno dei seguenti:

MQIGQPA_PREDEFINITO
L'identificativo utente controllato per l'autorizzazione è il valore del campo *UserIdentifier* in MQMD *separato* associato con il messaggio quando il messaggio è nella coda di trasmissione condivisa. Si tratta dell'identificativo utente del programma che ha inserito il messaggio nella coda di trasmissione condivisa ed è generalmente lo stesso dell'identificativo utente con cui è in esecuzione il gestore code remoto.

Se il profilo RESLEVEL indica che è necessario controllare più di un identificativo utente, viene controllato anche l'identificativo utente dell'agent IGQ locale (*IGQUserId*).

CONTEXT MQIGQPA
L'identificativo utente controllato per l'autorizzazione è il valore del campo *UserIdentifier* in MQMD *separato* associato con il messaggio quando il messaggio è nella coda di trasmissione condivisa. Si tratta dell'identificativo utente del programma che ha inserito il messaggio nella coda di trasmissione condivisa ed è generalmente lo stesso dell'identificativo utente con cui è in esecuzione il gestore code remoto.

Se il profilo RESLEVEL indica che è necessario controllare più di un identificativo utente, vengono controllati anche l'identificativo utente dell'agent IGQ locale (*IGQUserId*) e il valore del campo *UserIdentifier* in MQMD *integrato* . L'ultimo identificativo utente è di solito l'identificativo utente dell'applicazione che ha creato il messaggio.

MQIGQPA_ONLY_IGQ
L'identificativo utente controllato per l'autorizzazione è quello dell'agent IGQ locale (*IGQUserId*).

Se il profilo RESLEVEL indica che è necessario controllare più di un identificativo utente, questo identificativo utente viene utilizzato per tutti i controlli.

MQIGQPA_ALTERNATE_OR_IGQ
L'identificativo utente controllato per l'autorizzazione è quello dell'agent IGQ locale (*IGQUserId*).

Se il profilo RESLEVEL indica che è necessario controllare più di un identificativo utente, viene controllato anche il valore del campo *UserIdentifier* in MQMD *integrato* . Questo identificativo utente è di solito l'identificativo utente dell'applicazione che ha originato il messaggio.

Per determinare il valore di questo attributo, utilizzare l'utilità di selezione MQIA_IGQ_PUT_AUTHORITY con la chiamata MQINQ.

z/OS questo attributo è supportato solo su z/OS.

IGQUserId (MQLONG)

Questo attributo è applicabile solo se il gestore code locale è membro di un gruppo di condivisione code. Specifica l'identificativo utente associato all'agent di accodamento interno al gruppo locale (agent IGQ). Questo identificativo è uno degli identificativi utente che possono essere controllati per l'autorizzazione quando l'agent IGQ inserisce i messaggi nelle code locali. Gli identificativi utente effettivi selezionati dipendono dall'impostazione dell'attributo **IGQPutAuthority** e dalle opzioni di sicurezza esterne.

Se *IGQUserId* è vuoto, nessun identificativo utente è associato all'agent IGQ e il corrispondente controllo di autorizzazione non viene eseguito (anche se altri identificativi utente potrebbero ancora essere controllati per l'autorizzazione).

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_IGQ_USER_ID con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_USER_ID_LENGTH.

z/OS questo attributo è supportato solo su z/OS.

InhibitEvent (MQLONG)

Controlla se vengono generati eventi di inibizione (Inibisci acquisizione e inibisci immissione). Il valore è uno dei seguenti:

DISABILITAZIONE_MQEVR
Report eventi disabilitato.

MQEVR_ENABLED
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare l'MQIA_INIB_EVENT selector con la chiamata MQINQ.

Su z/OS, non è possibile utilizzare la chiamata MQINQ per determinare il valore di questo attributo.

z/OS *Accodamento IntraGroup(MQLONG)*

Questo attributo si applica solo se il gestore code locale è un membro di un gruppo di condivisione code. Indica se l'accodamento all'interno del gruppo è abilitato per il gruppo di condivisione code.

Il valore è uno dei seguenti:

MQIGQ_DISABLED
Tutti i messaggi destinati ad altri gestori code nel gruppo di condivisione code vengono trasmessi utilizzando canali convenzionali.

MQIGQ_ENABLED
I messaggi destinati ad altri gestori code nel gruppo di condivisione code vengono trasmessi utilizzando la coda di trasmissione condivisa se viene soddisfatta la condizione seguente:

- La lunghezza dei dati del messaggio più l'intestazione di trasmissione non supera 63 KB (64 512 byte).

  Si consiglia di assegnare un po' più di spazio rispetto alla dimensione di MQXQH per l'intestazione di trasmissione; la costante MQ_MSG_HEADER_LENGTH viene fornita per questo scopo.

Se questa condizione non viene soddisfatta, il messaggio viene trasmesso utilizzando i canali convenzionali.

Nota: Quando l'accodamento all'interno del gruppo è abilitato, l'ordine dei messaggi trasmessi utilizzando la coda di trasmissione condivisa non viene conservato rispetto a quelli trasmessi utilizzando canali convenzionali.

Per determinare il valore di questo attributo, utilizzare l'utilità di selezione MQIA_INTRA_GROUP_QUEUING con la chiamata MQINQ.

IPAddressVersion (MQLONG)

Specifica quale versione dell'indirizzo IP, IPv4 o IPv6, viene utilizzata.

Questo attributo è rilevante solo per i sistemi che eseguono IPv4 e IPv6 e interessa solo i canali definiti come *TransportType* di MQXPY_TCP quando si verifica una delle seguenti condizioni:

- *ConnectionName* del canale è un nome host che si risolve in un indirizzo IPv4 e IPv6 e il suo **LocalAddress** parametro non è specificato.
- *ConnectionName* e *LocalAddress* del canale sono entrambi nomi host che si risolvono in entrambi gli indirizzi IPv4 e IPv6 .

Il valore può essere uno dei seguenti:

MQIPADDR_IPV4
IPv4 .

MQIPADDR_IPV6
IPv6 .

Per stabilire il valore di questo attributo, utilizzare il selettore MQIA_IP_ADDRESS_VERSION con la chiamata MQINQ.

z/OS *ListenerTimer (MQLONG)*

Questo è l'intervallo di tempo (in secondi) tra i tentativi di IBM MQ di riavviare il listener se si è verificato un errore APPC o TCP/IP. Il valore deve essere compreso tra 5 e 9999, con un valore predefinito di 60.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_LISTENER_TIMER con la chiamata MQINQ.

LocalEvent (MQLONG)

Controlla se vengono generati eventi di errore locali. Il valore è uno dei seguenti:

DISABILITAZIONE_MQEVR
Report eventi disabilitato.

MQEVR_ENABLED
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_LOCAL_EVENT con la chiamata MQINQ.

Su z/OS, non è possibile utilizzare la chiamata MQINQ per determinare il valore di questo attributo.

LoggerEvent (MQLONG)

Controlla se vengono generati eventi di log di recupero. Il valore è uno dei seguenti:

DISABILITAZIONE_MQEVR
Report eventi disabilitato.

MQEVR_ENABLED
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_LOGGER_EVENT con la chiamata MQINQ.

Multi questo attributo è supportato solo su Multipiattaforme.

z/OS *LUGroupName (MQCHAR8)*

Questo è il nome LU generico per il listener LU 6.2 che gestisce le trasmissioni in entrata per il gruppo di condivisione code. Se si lascia questo nome vuoto, non è possibile utilizzare questo listener.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_LU_GROUP_NAME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_LU_NAME_LENGTH.

z/OS *Nome LUN (MQCHAR8)*

È il nome della LU da utilizzare per le trasmissioni della LU in uscita 6.2 . Impostare questo valore sulla stessa LU utilizzata dal listener per le trasmissioni in entrata. Se si lascia questo nome vuoto, viene utilizzata la LU predefinita APPC/MVS; questa è una variabile, quindi impostare sempre LUName se si utilizza LU6.2.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_LU_NAME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_LU_NAME_LENGTH.

LU62ARMSuffix (MQCHAR2)

Questo è il suffisso di SYS1.PARMLIB membro APPCPMxx, che designa LUADD per questo iniziatore di canali. Il comando z/OS SET APPC=xx viene emesso quando ARM riavvia l'iniziatore di canali. Se si lascia questo nome vuoto, non viene emesso alcun comando SET APPC=xx.

Questo attributo è supportato solo su z/OS .

Per stabilire il valore di questo attributo, utilizzare il selettore MQCA_LU62_ARM_SUFFIX con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_ARM_SUFFIX_LENGTH.

z/OS *LU62Channels (MQLONG)*

Questo è il numero massimo di canali che possono essere correnti o di client che possono essere connessi, che utilizzano il protocollo di trasmissione LU 6.2 .

Il valore deve essere compreso tra 0 e 9999, con un valore predefinito pari a 200. Se si imposta questo valore su zero, il protocollo di trasmissione LU 6.2 non viene utilizzato.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizza il selettore MQIA_LU62_CHANNELS con la chiamata MQINQ.

Canali MaxActive(MQLONG)

Questo attributo è il numero massimo di canali che possono essere *attivi* in qualsiasi momento.

Il valore predefinito è quello specificato per l'attributo `MaxChannels`.

Per z/OS, il valore deve essere compreso tra 1 e 9 999.

Per tutte le altre piattaforme, il valore predefinito è 999 999 999, che significa che il numero di canali attivi è illimitato o può essere impostato su un numero effettivo per imporre un limite.

MQ Appliance Non modificare il valore **MaxActiveChannels** in IBM MQ Appliance. Se si desidera limitare il numero massimo di canali client, utilizzare gli attributi MAXINST e MAXINSTC per canale nelle definizioni di canale SVRCONN per definire i limiti per ogni canale SVRCONN, fare riferimento a Configurazione del gestore code su IBM MQ Appliance nella documentazione di IBM MQ Appliance .

Il parametro **MaxActiveChannels** è un attributo del gestore code solo su z/OS . Sulle altre piattaforme, **MaxActiveChannels** è un attributo nel file `qm.ini` . Consultare Stanza del file di configurazione per l'accodamento distribuito per informazioni su come impostare l'attributo **MaxActiveChannels** su altre piattaforme.

Per determinare il valore di questo attributo, utilizza il selettore MQIA_ACTIVE_CHANNELS con la chiamata **MQINQ** .

Concetti correlati

Stati del canale

MaxChannels (MQLONG)

Questo attributo è il numero massimo di canali che possono essere *correnti* (inclusi canali di connessione server con client connessi).

Per z/OS, il valore deve essere compreso tra 1 e 9 999, con un valore predefinito di 200.

MQ Appliance Per IBM MQ Appliance, il valore predefinito è 999 999 999 e non deve essere modificato. Se si desidera limitare il numero massimo di canali client, utilizzare gli attributi MAXINST e MAXINSTC per canale nelle definizioni di canale SVRCONN per definire i limiti per ogni canale SVRCONN, fare riferimento a Configurazione del gestore code su IBM MQ Appliance nella documentazione di IBM MQ Appliance .

Un sistema che è occupato a servire le connessioni dalla rete potrebbe richiedere un numero superiore rispetto all'impostazione predefinita. Determinare il valore corretto per il proprio ambiente, osservando idealmente il comportamento del sistema durante la verifica.

Per tutte le altre piattaforme, il valore predefinito è 100. È possibile impostare **MaxChannels** su un valore diverso per limitare il numero massimo di canali correnti, se necessario.

Il parametro **MaxChannels** è un attributo del gestore code solo su z/OS . Sulle altre piattaforme, **MaxChannels** è un attributo nel file `qm.ini` . Consultare Stanza del file di configurazione per l'accodamento distribuito per informazioni su come impostare l'attributo **MaxChannels** su altre piattaforme.

Per determinare il valore di questo attributo, utilizza il selettore MQIA_MAX_CHANNELS con la chiamata **MQINQ** .

Concetti correlati

Stati del canale

MaxHandles (MQLONG)

Questo è il numero massimo di handle aperti che qualsiasi attività può utilizzare contemporaneamente. Ogni chiamata MQOPEN riuscita per una singola coda (o per un oggetto che non è una coda) utilizza un handle. Tale handle diventa disponibile per il riutilizzo quando l'oggetto viene chiuso. Tuttavia, quando viene aperto un elenco di distribuzione, a ogni coda nell'elenco di distribuzione viene assegnato un handle separato e in modo che la chiamata MQOPEN utilizzi un numero di handle pari al numero di code presenti nell'elenco di distribuzione. Questo deve essere preso in considerazione quando si decide un valore adatto per *MaxHandles*.

La chiamata MQPUT1 esegue una chiamata MQOPEN come parte della sua elaborazione; di conseguenza, MQPUT1 utilizza un numero di handle pari a quello di MQOPEN, ma gli handle vengono utilizzati solo per la durata della chiamata MQPUT1 .

Su z/OS, *attività* indica un'attività CICS , un'attività MVS o una regione dipendente da IMS .

Il valore è compreso tra 1 e 999 999 999. Il valore predefinito è determinato dall'ambiente:

- Su z/OS, il valore predefinito è 100.
- In tutti gli altri ambienti, il valore predefinito è 256.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_MAX_HANDLES con la chiamata MQINQ.

Lunghezza MaxMsg(MQLONG)

Si tratta della lunghezza del messaggio *fisico* più lungo che il gestore code può gestire. Tuttavia, poiché l'attributo gestore code **MaxMsgLength** può essere impostato indipendentemente dall'attributo coda **MaxMsgLength** , il messaggio fisico più lungo che può essere inserito in una coda è il minore di questi due valori.

Se il gestore code supporta la segmentazione, un'applicazione può inserire un messaggio logico più lungo del minore dei due **MaxMsgLength** attributi, ma solo se l'applicazione specifica l'indicatore MQMF_SEGMENTATION_ALLOWED in MQMD. Se viene specificato tale indicatore, il limite superiore per la lunghezza di un messaggio logico è 999 999 999 999 byte, ma di solito i vincoli di risorsa imposti dal sistema operativo o dall'ambiente in cui l'applicazione è in esecuzione, risultano in un limite inferiore.

Il limite inferiore per l'attributo **MaxMsgLength** è 32 KB (32 768 byte). Il limite superiore è 100 MB (104 857 600 byte).

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_MAX_MSG_LENGTH con la chiamata MQINQ.

MaxPriority (MQLONG)

Questa è la priorità massima dei messaggi supportata dal gestore code. Le priorità vanno da zero (più basso) a *MaxPriority* (più alto).

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_MAX_PRIORITY con la chiamata MQINQ.

Lunghezza MaxProperties(MQLONG)

Viene utilizzato per controllare la dimensione delle proprietà che possono fluire con un messaggio. Ciò include il nome della proprietà in byte e la dimensione del valore della proprietà anche in byte.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_MAX_PROPERTIES_LENGTH con la chiamata MQINQ.

Multi *Dimensione MaxQFile(MQLONG)*

La dimensione massima, in megabyte, che un file della coda può raggiungere.

Tabella 560. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

È possibile che un file della coda superi la dimensione massima, se è configurato su un valore inferiore alla dimensione del file della coda corrente. Se ciò si verifica, il file della coda non accetta più nuovi messaggi, ma consente l'utilizzo dei messaggi esistenti. Quando la dimensione del file della coda è scesa al di sotto del valore configurato, è consentito inserire nuovi messaggi nella coda.

Nota: Questa figura può essere diversa dal valore dell'attributo configurato sulla coda, poiché internamente il gestore code potrebbe dover utilizzare una dimensione di blocco maggiore per raggiungere la dimensione scelta. Consultare Modifica dei file della coda IBM MQ per ulteriori informazioni sulla modifica della dimensione dei file della coda e della granularità e della dimensione del blocco.

Quando la granularità deve essere modificata perché questo attributo è stato aumentato, viene scritto il messaggio di avvertenza AMQ7493W `Granularità modificata` nei log AMQERR. Ciò indica che è necessario pianificare lo svuotamento della coda, affinché IBM MQ adotti la nuova granularità.

Il valore massimo di questo attributo è 267.386.880 MB e il valore predefinito e il valore migrato è 2.088.960 MB, che è il valore massimo corrente per una coda con una granularità pari a 512.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_MAX_Q_FILE_SIZE con la chiamata MQINQ.

MaxUncommitted(MQLONG)

Questo è il numero massimo di messaggi di cui non è stato eseguito il commit che possono esistere all'interno di un'unità di lavoro. Il numero di messaggi di cui non è stato eseguito il commit è la somma dei seguenti dall'inizio dell'unità di lavoro corrente:

- Messaggi immessi dall'applicazione con l'opzione MQPMO_SYNCPOINT
- Messaggi richiamati dall'applicazione con l'opzione MQGMO_SYNCPOINT
- Messaggi di trigger e messaggi di report COA generati dal gestore code per i messaggi inseriti con l'opzione MQPMO_SYNCPOINT
- Messaggi di report COD generati dal gestore code per i messaggi richiamati con l'opzione MQGMO_SYNCPOINT

I seguenti messaggi non vengono conteggiati come non sottoposti a commit:

- Messaggi immessi o richiamati dall'applicazione all'esterno di un'unità di lavoro
- Attivare i messaggi o i messaggi di report COA/COD generati dal gestore code come risultato di messaggi immessi o richiamati all'esterno di un'unità di lavoro
- Messaggi di report di scadenza generati dal gestore code (anche se la chiamata che causa il messaggio di report di scadenza ha specificato MQGMO_SYNCPOINT)
- I messaggi di evento generati dal gestore code (anche se la chiamata che causa il messaggio di evento ha specificato MQPMO_SYNCPOINT o MQGMO_SYNCPOINT)

Nota:

1. I messaggi di report di eccezione vengono generati da MCA (Message Channel Agent) o dall'applicazione e vengono trattati allo stesso modo dei messaggi ordinari immessi o richiamati dall'applicazione.
2. Quando un messaggio o un segmento viene inserito con l'opzione MQPMO_SYNCPOINT, il numero di messaggi di cui non è stato eseguito il commit viene incrementato di uno indipendentemente dal numero di messaggi fisici effettivamente risultanti dall'inserimento. (È possibile che si verifichi più di un messaggio fisico se il gestore code deve suddividere il messaggio o il segmento.)
3. Quando un elenco di distribuzione viene inserito con l'opzione MQPMO_SYNCPOINT, il numero di messaggi di cui non è stato eseguito il commit viene incrementato di un *per ciascun messaggio fisico generato*. Può essere piccolo come uno o grande come il numero di destinazioni nell'elenco di distribuzione.

Il limite inferiore per questo attributo è 1; il limite superiore è 999 999 999. Il valore predefinito è 10000.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_MAX_UNCOMMITTED_MSGS con la chiamata MQINQ.

Account MQI (MQLONG)

Controlla la raccolta delle informazioni di account per i dati MQI.

Il valore è uno dei seguenti:

MMON_UN
Raccogliere i dati di account API.

MQMON_DISATTIVO
Non raccogliere i dati di account API. Questo è il valore predefinito.

Se si imposta l'attributo del gestore code ACCTCONO su ENABLED, questo valore potrebbe essere sovrascritto per le singole connessioni utilizzando il campo Opzioni nella struttura MQCNO. Le modifiche a questo valore sono effettive solo per le connessioni al gestore code che si verificano dopo la modifica all'attributo.

questo attributo è supportato solo su Multipiattaforme.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_ACCOUNTING_MQI con la chiamata MQINQ.

Statistiche MQI (MQLONG)

Controlla la raccolta delle informazioni di controllo statistiche per il gestore code.

Il valore è uno dei seguenti:

MMON_UN
Raccogliere statistiche MQI.

MQMON_DISATTIVO
Non raccogliere statistiche MQI. Questo è il valore predefinito.

questo attributo è supportato solo su Multipiattaforme.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_STATISTICS_MQI con la chiamata MQINQ.

MsgMarkBrowseInterval (MQLONG)

L'intervallo di tempo in millisecondi dopo il quale il gestore code può rimuovere automaticamente il contrassegno dai messaggi di ricerca.

Questo è un intervallo di tempo (in millisecondi) dopo il quale il gestore code può rimuovere automaticamente il contrassegno dai messaggi di ricerca.

Questo attributo descrive l'intervallo di tempo per cui è previsto che i messaggi contrassegnati come sfogliati da una chiamata a MQGET, utilizzando l'opzione di richiamo messaggio MQGMO_MARK_BROWSE_CO_OP, rimangano contrassegnati come sfogliati.

Il gestore code potrebbe deselezionare automaticamente i messaggi sfogliati che sono stati contrassegnati come sfogliati per la serie di handle cooperante quando sono stati contrassegnati per un intervallo superiore a quello approssimativo.

Ciò non influisce sullo stato di qualsiasi messaggio contrassegnato come browse, ottenuto da una chiamata a MQGET, utilizzando l'opzione di richiamo messaggio MQGMO_MARK_BROWSE_HANDLE.

Il valore massimo è 999 999 999 e il valore predefinito è 5000. Un valore speciale di -1 per *MsgMarkBrowseInterval* rappresenta un intervallo di tempo illimitato.

Attenzione: Questo valore non deve essere inferiore al valore predefinito di 5000.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_MSG_MARK_BROWSE_INTERVAL con la chiamata MQINQ.

z/OS *OutboundPortmassimo (MQLONG)*

Si tratta del numero di porta più alto nell'intervallo, definito da OutboundPortMin e da OutboundPortMax, dei numeri di porta da utilizzare per collegare i canali in uscita.

Il valore è un numero intero compreso tra 0 e 65535 e deve essere uguale o maggiore del valore minimo di OutboundPort. Il valore predefinito è 0.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_OUTBOUND_PORT_MAX con la chiamata MQINQ.

z/OS *OutboundPortmin (MQLONG)*

È il numero di porta più basso nell'intervallo, definito da OutboundPortMin e OutboundPortMax, dei numeri di porta da utilizzare per collegare i canali in uscita.

Il valore è un numero intero compreso tra 0 e 65535 e deve essere uguale o inferiore al valore massimo di OutboundPort. Il valore predefinito è 0.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_OUTBOUND_PORT_MIN con la chiamata MQINQ.

PerformanceEvent (MQLONG)

Controlla se vengono generati eventi correlati alle prestazioni. È una dei seguenti valori:

DISABILITAZIONE_MQEVR
Report eventi disabilitato.

MQEVR_ENABLED
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_PERFORMANCE_EVENT con la chiamata MQINQ.

Piattaforma (MQLONG)

Indica il sistema operativo su cui è in esecuzione il gestore code:

AIX MQPL
AIX (stesso valore di MQPL_UNIX).

APPLICAZIONE_MQPL
IBM MQ Appliance

MVS MQPL
z/OS (stesso valore di MQPL_ZOS).

MQPL_OS390
z/OS (stesso valore di MQPL_ZOS).

MQPL_OS400
IBM i.

MQPL_UNIX
UNIX.

MQPL_WINDOWS_NT
Windows in altri sistemi.

ZOS MQPL
z/OS.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_PLATFORM con la chiamata MQINQ.

PubSubNPInputMsg (MQLONG)

Indica se eliminare o conservare un messaggio di input non consegnato.

Il valore è uno dei seguenti:

MQUNDELIVER_DISCARD
I messaggi di input non persistenti possono essere eliminati se non è possibile elaborarli.

Questo è il valore predefinito.

KEEP MQUNDELIVER_
I messaggi di input non persistenti non verranno eliminati se non è possibile elaborarli. In questa situazione l'interfaccia di pubblicazione / sottoscrizione in coda continuerà a ritentare il processo ad intervalli appropriati e non continuerà l'elaborazione dei successivi messaggi.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_PUBSUB_NP_MSG con la chiamata MQINQ.

Risposta NP PubSub(MQLONG)
Controlla il comportamento dei messaggi di risposta non recapitati.

Il valore è uno dei seguenti:

MQUNDELIVER_NORMAL
Le risposte non persistenti che non possono essere inserite nella coda di risposta vengono inserite nella coda di messaggi non recapitabili, se non possono essere inserite nella DLQ, vengono eliminate.

MQUNDELIVER_SAFE
Le risposte non persistenti che non è possibile collocare sulla coda di risposta vengono collocate sulla coda messaggi non recapitabili. Se la risposta non può essere impostata e non può essere posizionata nella DLQ, l'interfaccia di pubblicazione / sottoscrizione in coda eseguirà il rollback dell'operazione corrente, quindi riproverà a intervalli appropriati e non continuerà l'elaborazione dei successivi messaggi.

MQUNDELIVER_DISCARD
Le risposte non persistenti non vengono inserite nella coda di risposta e vengono eliminate.

Questo è il valore predefinito per i nuovi gestori code.

KEEP MQUNDELIVER_
Le risposte non persistenti non vengono inserite nella coda di messaggi non recapitabili o eliminate. Invece, l'interfaccia di pubblicazione / sottoscrizione accodata eseguirà il backout dell'operazione corrente e ritenterà l'operazione ad intervalli appropriati.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_PUBSUB_NP_RESP con la chiamata MQINQ.

Valore predefinito per i gestori code migrati.
Se il gestore code è stato migrato da IBM MQ V6.0, il valore iniziale di questo attributo dipende dai valori di *DiscardNonPersistentResponse* e *DLQNonPersistentResponse* prima della migrazione, come mostrato nella seguente tabella.

		Risposta DLQNonPersistent		
		Sì	**No**	**Non impostato**
DiscardNonPersistentResponse	**Sì**	MQUNDELIVER_NORMAL	MQUNDELIVER_DISCARD	MQUNDELIVER_NORMAL
	No	MQUNDELIVER_SAFE	KEEP MQUNDELIVER_	MQUNDELIVER_SAFE
	Non impostato	Se SyncPointPersistente = No, MQUNDELIVERED_SAFE altrimenti MQUNDELIVERED_NORMAL	Se SyncPointPersistente = No, MQUNDELIVERED_KEEP altrimenti MQUNDELIVERED_DISCARD	Se SyncPointPersistente = No, MQUNDELIVERED_SAFE altrimenti MQUNDELIVERED_NORMAL

PubSubMaxMsgRetryCount (MQLONG)
Il numero di tentativi durante l'elaborazione di un messaggio di comando non riuscito nel punto di sincronizzazione.

Il valore è uno dei seguenti:

0 - 999 999 999
Il valore predefinito è 5.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_PUBSUB_MAXMSG_RETRY_COUNT con la chiamata MQINQ.

PubSubSyncPoint (MQLONG)

Indica se solo i messaggi persistenti o tutti i messaggi vengono elaborati nel punto di sincronizzazione.

Il valore è uno dei seguenti:

MQSYNCPOINT_IFPER
Ciò fa sì che l'interfaccia di pubblicazione / sottoscrizione accodata riceva messaggi non persistenti all'esterno del punto di sincronizzazione. Se il daemon riceve una pubblicazione al di fuori del syncpoint, inoltra la pubblicazione ai sottoscrittori ad esso noti al di fuori del syncpoint.

Questo è il valore predefinito.

MQSYNCPOINT_SÌ
Ciò fa sì che l'interfaccia di pubblicazione / sottoscrizione accodata riceva tutti i messaggi nel punto di sincronizzazione.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_PUBSUB_SYNC_PT con la chiamata MQINQ.

Modalità PubSub(MQLONG)

Se il motore di pubblicazione / sottoscrizione e l'interfaccia di pubblicazione / sottoscrizione accodata sono in esecuzione, consentendo quindi alle applicazioni di pubblicare / sottoscrivere utilizzando l'interfaccia di programmazione dell'applicazione e le code monitorate dall'interfaccia di pubblicazione / sottoscrizione accodata.

Il valore è uno dei seguenti:

COMPAT_MQPSM
Il motore di pubblicazione/sottoscrizione è in esecuzione. È quindi possibile pubblicare / sottoscrivere utilizzando l'API (application programming interface). L'interfaccia di pubblicazione / sottoscrizione accodata non è in esecuzione, pertanto qualsiasi messaggio inserito nelle code monitorate dall'interfaccia di pubblicazione / sottoscrizione accodata non viene utilizzato. Questa impostazione viene utilizzata per compatibilità con WebSphere Message Broker V6 o versioni precedenti utilizzando questo gestore code, poiché deve leggere le stesse code da cui l'interfaccia di pubblicazione / sottoscrizione accodata normalmente legge.

DISABILITAZIONE_MQPSM
Il motore di pubblicazione/sottoscrizione e l'interfaccia di pubblicazione/sottoscrizione in coda non sono in esecuzione. Non è quindi possibile pubblicare / sottoscrivere utilizzando l'API (application programming interface). I messaggi di pubblicazione / sottoscrizione inseriti nelle code monitorate dall'interfaccia di pubblicazione / sottoscrizione accodata non vengono utilizzati.

MQPSM_ENABLED
Il motore di pubblicazione/sottoscrizione e l'interfaccia di pubblicazione/sottoscrizione in coda sono in esecuzione. È quindi possibile pubblicare / sottoscrivere utilizzando l'API (application programming interface) e le code monitorate dall'interfaccia di pubblicazione / sottoscrizione accodata. Questo è il valore predefinito iniziale del gestore code.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_PUBSUB_MODE con la chiamata MQINQ.

QMgrDesc (MQCHAR64)

Utilizzare questo campo per un commento che descrive il gestore code. Il contenuto del campo non ha alcun significato per il gestore code, ma il gestore code potrebbe richiedere che il campo contenga solo caratteri che possono essere visualizzati. Non può contenere caratteri null; se necessario, viene riempito a destra con spazi. In un'installazione DBCS, questo campo può contenere caratteri DBCS (con una lunghezza massima di 64 byte).

Nota: Se questo campo contiene caratteri non presenti nella serie di caratteri del gestore code (come definito dall'attributo del gestore code **CodedCharSetId**), tali caratteri potrebbero essere tradotti in modo non corretto se questo campo viene inviato a un altro gestore code.

- Su z/OS, il valore predefinito è il nome del prodotto e il numero di versione.

- In tutti gli altri ambienti, il valore predefinito è vuoto.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_Q_MGR_DESC con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_Q_MGR_DESC_LENGTH.

QMgrIdentifier (MQCHAR48)

Questo è un nome univoco generato internamente per il gestore code.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_K_MGR_IDENTIFIER con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_Q_MGR_IDENTIFIER_LENGTH.

Questo attributo è supportato nei seguenti ambienti:

- 

AIX
- 

IBM i
- 

Linux
- 

Windows
- 

z/OS

e i client IBM MQ collegati a questi sistemi.

QMgrName (MQCHAR48)

Questo è il nome del gestore code locale, vale a dire il nome del gestore code a cui l'applicazione è collegata.

I primi 12 caratteri del nome vengono utilizzati per creare un identificativo di messaggio univoco (vedere MQMD - campo MsgId). I gestori code che possono comunicare tra loro devono quindi avere nomi diversi nei primi 12 caratteri, in modo che gli ID messaggio siano univoci nella rete del gestore code.

Su z/OS, il nome è uguale al nome del sottosistema, che è limitato a 4 caratteri non vuoti.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_Q_MGR_NAME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_Q_MGR_NAME_LENGTH.

QSGName (MQCHAR4)

È il nome del gruppo di condivisione code a cui appartiene il gestore code locale. Se il gestore code locale non appartiene a un gruppo di condivisione code, il nome è vuoto.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_QSG_NAME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_QSG_NAME_LENGTH.

questo attributo è supportato solo su z/OS.

QueueAccounting (MQLONG)

Controlla la raccolta delle informazioni di account per le code.

Il valore è uno dei seguenti:

MQMON_NONE
Non raccogliere i dati di account per le code, indipendentemente dall'impostazione dell'attributo di account della coda ACCTQ. Questo è il valore predefinito.

MQMON_DISATTIVO
Non raccogliere i dati di account per le code che specificano QMGR nell'attributo della coda ACCTQ.

MMON_UN
Raccogliere i dati di account per code che specificano QMGR nell'attributo della coda ACCTQ.

Le modifiche a questo valore sono effettive solo per le connessioni al gestore code che si verificano dopo la modifica all'attributo.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_ACCOUNTING_Q con la chiamata MQINQ.

QueueMonitoring (MQLONG)

Specifica l'impostazione predefinita per il monitoraggio online delle code.

Se l'attributo della coda **QueueMonitoring** è impostato su MQMON_Q_MGR, questo attributo specifica il valore assunto dal canale. Il valore può essere:

MQMON_DISATTIVO
La raccolta dei dati di monitoraggio online è disattivata. Questo è il valore predefinito iniziale del gestore code.

MQMON_NONE
La raccolta dei dati di monitoraggio in linea è disattivata per le code indipendentemente dall'impostazione del loro attributo **QueueMonitoring** .

MMON_LOW
La raccolta dei dati di controllo online è attivata, con un rapporto basso di raccolta dati.

MQMON_MEDIO
La raccolta dati di monitoraggio online è attivata, con un rapporto moderato di raccolta dati.

MQMON_HIGH
La raccolta dati di monitoraggio online è attivata, con un rapporto elevato di raccolta dati.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_MONITORING_Q con la chiamata MQINQ.

QueueStatistics (MQLONG)

Controlla la raccolta dei dati statistici per le code.

È una dei seguenti valori:

MQMON_NONE
Non raccogliere le statistiche della coda per le code, indipendentemente dall'impostazione dell'attributo della coda **QueueStatistics** . Questo è il valore predefinito.

MQMON_DISATTIVO
Non raccogliere i dati statistici per le code che specificano Gestore code nell'attributo della coda **QueueStatistics** .

MMON_UN
Raccogliere i dati delle statistiche per le code che specificano il gestore code nell'attributo della coda **QueueStatistics** .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_STATISTICS_Q con la chiamata MQINQ.

z/OS *ReceiveTimeout (MQLONG)*

Specifica il tempo di attesa di un canale TCP/IP per ricevere i dati, inclusi gli heartbeat, dal partner prima di tornare allo stato inattivo. Si riferisce solo ai canali di messaggi e non ai canali MQI.

Il significato esatto di ReceiveTimeout viene modificato dal valore specificato nel tipo ReceiveTimeout. Il tipo ReceiveTimeoutpuò essere impostato su uno dei seguenti:

- MQRCVTIME_EQUAL - Questo valore è il numero in secondi di attesa del canale. Specificare un valore compreso nell'intervallo 0 - 999999.
- MQRCVTIME_ADD - questo valore è il numero in secondi da aggiungere all'HBINT negoziato e determina per quanto tempo un canale attende. Specificare un valore compreso tra 1 e 999999.

- MQRCVTIME_MULTIPLY - questo valore è un moltiplicatore da applicare all'HBINT negoziato. Specificare un valore 0 o un valore compreso nell'intervallo 2 - 99.

Il valore predefinito è 0.

Impostare il tipo ReceiveTimeoutsu MQRCVTIME_MULTIPLY o MQRCVTIME_EQUAL e ReceiveTimeout su 0, per impedire a un canale di andare in timeout in attesa di ricevere i dati dal partner.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_RECEIVE_TIMEOUT con la chiamata MQINQ.

z/OS *ReceiveTimeoutmin (MQLONG)*

Questo è il tempo minimo, in secondi, che un canale TCP/IP attende per ricevere i dati, inclusi gli heartbeat, dal proprio partner, prima di tornare allo stato inattivo.

Si applica solo ai canali di messaggi, non ai canali MQI. Il valore deve essere compreso nell'intervallo tra 0 e 999999, con un valore predefinito di 0.

Se si utilizza il tipo ReceiveTimeoutper specificare che il tempo di attesa del canale TCP/IP deve essere calcolato in relazione al valore negoziato di HBINT e il valore risultante è inferiore al valore di questo parametro, viene invece utilizzato questo valore.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare l'utilità di selezione MQIA_RECEIVE_TIMEOUT_MIN con la chiamata MQINQ.

z/OS *Tipo ReceiveTimeout(MQLONG)*

Questo è il qualificatore, applicato a ReceiveTimeout per definire il tempo di attesa di un canale TCP/IP per ricevere i dati, inclusi gli heartbeat, dal partner, prima di tornare allo stato inattivo. Si applica solo ai canali di messaggi, non ai canali MQI.

Il valore è uno dei seguenti:

MQRCVTIME_MULTIPLY
ReceiveTimeout è un moltiplicatore da applicare al valore HBINT negoziato per determinare il tempo di attesa di un canale. Questo è il valore predefinito.

MQRCVTIME_ADD
ReceiveTimeout è un valore, espresso in secondi, da aggiungere al valore HBINT negoziato per determinare il tempo di attesa di un canale.

MQRCVTIME_EQUAL
ReceiveTimeout è un valore, in secondi, che il canale attende.

Per arrestare il timeout di un canale per ricevere i dati dal partner, impostare il tipo ReceiveTimeoutsu MQRCVTIME_MULTIPLY o MQRCVTIME_EQUAL e ReceiveTimeout su 0.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_RECEIVE_TIMEOUT_TYPE con la chiamata MQINQ.

RemoteEvent (MQLONG)

Controlla se vengono generati eventi di errori remoti. È una dei seguenti valori:

DISABILITAZIONE_MQEVR
Report eventi disabilitato.

MQEVR_ENABLED
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_REMOTE_EVENT con la chiamata MQINQ.

RepositoryName (MQCHAR48)

Questo è il nome di un cluster per il quale questo gestore code fornisce un servizio gestore repository. Se il gestore code fornisce questo servizio per più di un cluster, *RepositoryNamelist* specifica il nome di un oggetto elenco nomi che identifica i cluster e *RepositoryName* è vuoto. Almeno uno tra *RepositoryName* e *RepositoryNamelist* deve essere vuoto.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_REPOSITORY_NAME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_Q_MGR_NAME_LENGTH.

RepositoryNamelist (MQCHAR48)

Questo è il nome di un oggetto elenco nomi che contiene i nomi dei cluster per cui questo gestore code fornisce un servizio gestore repository. Se il gestore code fornisce questo servizio solo per un cluster, l'oggetto elenco nomi contiene un unico nome. In alternativa, *RepositoryName* può essere utilizzato per indicare il nome del cluster, nel qual caso *RepositoryNamelist* è vuoto. Almeno uno tra *RepositoryName* e *RepositoryNamelist* deve essere vuoto.

Per stabilire il valore di questo attributo, utilizzare il selettore MQCA_REPOSITORY_NAMELIST con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_NAMELIST_NAME_LENGTH.

ScyCase(MQCHAR8)

Specifica se il gestore code supporta i nomi dei profili di sicurezza in caratteri misti o solo in caratteri maiuscoli.

Il valore è uno dei seguenti:

MQSCYC_UPPER
I nomi dei profili di protezione devono essere in maiuscolo.

MIXED MQSCY
I nomi dei profili di sicurezza possono essere in caratteri maiuscoli o maiuscoli e minuscoli.

Le modifiche a questo attributo diventano effettive quando un comando Aggiorna sicurezza viene eseguito con *SecurityType(MQSECTYPE_CLASSES)* specificato.

z/OS questo attributo è supportato solo su z/OS.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_SECURITY_CASE con la chiamata MQINQ.

Nome SharedQMgr(MQLONG)

Specifica se *ObjectQmgrName* deve essere utilizzato o trattato come gestore code locale su una chiamata MQOPEN, per una coda condivisa, quando *ObjectQmgrName* è quello di un altro gestore code nel gruppo di condivisione code.

Il valore può essere uno dei seguenti:

MQSQQM_USO
ObjectQmgrName viene utilizzato e viene aperta la coda di trasmissione appropriata.

MQSQQM_IGNORE
Se la coda di destinazione è condivisa e il *ObjectQmgrName* è quello di un gestore code nello stesso gruppo di condivisione code, l'apertura viene eseguita localmente.

Questo attributo è valido solo su z/OS.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_SHARED_Q_Q_MGR_NAME con la chiamata MQINQ.

SPLCAP
Indica se le funzionalità di sicurezza di Advanced Message Security sono disponibili per un gestore code.

MQCAP_SUPPORTATO
Questo è il valore predefinito se il componente AMS è installato per l'installazione in cui è in esecuzione il gestore code.

MQCAP_NON_SUPPORTATO

SSLEvent (MQLONG)
Specifica se vengono generati eventi TLS.

È una dei seguenti valori:

MQEVR_ENABLED
Generare eventi TLS, come segue:

MQRC_CHANNEL_SSL_ERROR

DISABILITAZIONE_MQEVR
Non generare eventi TLS; questo è il valore predefinito.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_SSL_EVENT con la chiamata MQINQ.

SSLFIPSRequired (MQLONG)

Nota: Su AIX, Linux, and Windows, IBM MQ fornisce la conformità FIPS 140-2 tramite il modulo crittografico IBM Crypto for C (ICC) . Il certificato per questo modulo è stato spostato nello stato cronologico. I clienti devono visualizzare il certificatoIBM Crypto for C (ICC) ed essere a conoscenza di eventuali consigli forniti da NIST. Un modulo FIPS 140-3 di sostituzione è attualmente in corso e il relativo stato può essere visualizzato ricercandolo in NIST CMVP modules in process list.

IBM MQ Operator 3.2.0 e l'immagine del contenitore del gestore code 9.4.0.0 sono basati su UBI 9. La conformità FIPS 140-3 è attualmente in sospeso e il suo stato può essere visualizzato ricercando "Red Hat Enterprise Linux 9 - OpenSSL FIPS Provider" in NIST CMVP modules in process list.

Ciò consente di specificare che solo gli algoritmi certificati FIPS devono essere utilizzati se la crittografia viene eseguita in IBM MQ, piuttosto che nell'hardware di crittografia. Se l'hardware di crittografia è configurato, i moduli di crittografia utilizzati sono quei moduli forniti dal prodotto hardware; questi moduli potrebbero o meno essere certificati FIPS ad un livello particolare a seconda del prodotto hardware in uso.

Il valore è uno dei seguenti:

MQSSL_FIPS_NO
Utilizzare qualsiasi CipherSpec supportato sulla piattaforma in uso. Questo è il valore predefinito.

SÌ MQSSL_FIPS
Utilizzare solo algoritmi di codifica certificati FIPS in CipherSpecs consentiti su tutte le connessioni TLS da e verso questo gestore code.

Questo parametro è valido solo su piattaforme z/OS, AIX, Linux, and Windows .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_SSL_FIPS_REQUIRED con la chiamata MQINQ.

Attività correlate
Specifica che solo i CipherSpecs certificati FIPS vengono utilizzati al runtime sul client MQI
Riferimenti correlati
FIPS (Federal Information Processing Standards) per AIX, Linux, and Windows

Conteggio SSLKeyReset(MQLONG)
Specifica quando gli MCA (Message Channel Agent) del canale TLS che avviano la comunicazione reimpostano la chiave segreta utilizzata per la codifica sul canale.

Il valore rappresenta il numero totale di byte non crittografati inviati e ricevuti sul canale prima che la chiave segreta venga rinegoziata. Il numero di byte include le informazioni di controllo inviate da MCA.

Il valore è un numero compreso tra 0 e 999 999 999, con un valore predefinito pari a 0. Se si specifica un conteggio di reimpostazione della chiave segreta TLS compreso tra 1 byte e 32 KB, i canali TLS utilizzeranno un conteggio di reimpostazione della chiave segreta di 32 KB. Ciò consente di evitare il costo di elaborazione di un numero eccessivo di reimpostazioni della chiave che si verificherebbe per valori di reimpostazione della chiave segreta TLS di piccole dimensioni.

La chiave segreta viene rinegoziata quando il numero totale di byte non codificati inviati e ricevuti dall'MCA del canale di avvio supera il valore specificato. Se gli heartbeat del canale sono abilitati, la chiave segreta viene rinegoziata prima che i dati vengano inviati o ricevuti in seguito a un heartbeat del canale o quando il numero totale di byte non codificati supera il valore specificato, a seconda di quale di essi si verifica per primo.

Il conteggio dei byte inviati e ricevuti per la rinegoziazione include le informazioni di controllo inviate e ricevute dal canale MCA e viene reimpostato ogni volta che si verifica una rinegoziazione.

Utilizzare il valore 0 per indicare che le chiavi segrete non vengono mai rinegoziate.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_SSL_RESET_COUNT con la chiamata MQINQ.

Evento StartStop(MQLONG)

Controlla se vengono generati eventi di avvio e arresto. Il valore è uno dei seguenti:

DISABILITAZIONE_MQEVR
Report eventi disabilitato.

MQEVR_ENABLED
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_START_STOP_EVENT con la chiamata MQINQ.

StatisticsInterval (MQLONG)

Specifica la frequenza (in secondi) con cui scrivere i dati di monitoraggio delle statistiche nella coda di controllo.

Il valore è un numero intero compreso tra 0 e 604800, con un valore predefinito di 1800 (30 minuti).

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_STATISTICS_INTERVALcon la chiamata MQINQ.

SyncPoint (MQLONG)

Ciò indica se il gestore code locale supporta le unità di lavoro e di sincronizzazione con le chiamate MQGET, MQPUT e MQPUT1 .

MQSP_DISPONIBILE
Unità di lavoro e punto di sincronizzazione disponibile.

MQSP_NON_DISPONIBILE
Unità di lavoro e punto di sincronizzazione non disponibile.

- Su z/OS questo valore non viene mai restituito.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_SYNCPOINT con la chiamata MQINQ.

z/OS *TCPChannel (MQLONG)*

Questo è il numero massimo di canali che possono essere correnti o di client che possono essere connessi, che utilizzano il protocollo di trasmissione TCP/IP.

Il valore deve essere compreso tra 0 e 9999, con un valore predefinito pari a 200. Se si specifica 0, TCP/IP non viene utilizzato.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_TCP_CHANNELS con la chiamata MQINQ.

z/OS *TCPKeepAlive (MQLONG)*

Specifica se utilizzare TCP KEEPALIVE per verificare che l'altra estremità della connessione sia ancora disponibile. Se non è disponibile, il canale viene chiuso.

Il valore è uno dei seguenti:

SÌ MQTCPKEEP
Utilizzare TCP KEEPALIVE come specificato nel dataset di configurazione del profilo TCP. Se si specifica l'attributo del canale KeepAliveInterval (KAINT), viene utilizzato il valore su cui è impostato.

MQTCPKEEP_NO
Non utilizzare TCP KEEPALIVE. Questo è il valore predefinito.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_TCP_KEEP_ALIVE con la chiamata MQINQ.

z/OS *TCPName (MQCHAR8)*

Questo è il nome dell'unico stack TCP/IP o dello stack TCP/IP preferito che verrà utilizzato, a seconda del valore di TCPStackType. Questo parametro è applicabile solo in ambienti CINET a più stack. Il valore predefinito è TCPIP.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_TCP_NAME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_TCP_NAME_LENGTH.

z/OS *TCPStackType (MQLONG)*

Specifica se l'iniziatore di canali può utilizzare solo lo stack TCP/IP specificato in TCPName o se può eseguire il bind a qualsiasi stack TCP/IP selezionato. Questo parametro è applicabile solo in ambienti CINET a più stack.

Il valore è uno dei seguenti:

MQTCPSTACK_SINGLE
L'iniziatore di canali può utilizzare solo gli spazi di indirizzo TCP/IP denominati in TCPName. Questo è il valore predefinito.

MQTCPSTACK_MULTIPLE
L'iniziatore di canali può utilizzare qualsiasi spazio di indirizzo TCP/IP disponibile. Il valore predefinito è quello specificato in TCPName se non ne è specificato un altro per un canale o un listener.

Questo attributo è supportato solo su z/OS .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_TCP_STACK_TYPE con la chiamata MQINQ.

Registrazione TraceRoute(MQLONG)

Controlla la registrazione delle informazioni di traccia - instradamento.

Il valore è uno dei seguenti:

MQRECORDING_DISABILITATO
Non è consentita alcuna aggiunta ai messaggi di instradamento traccia.

MQRECORDING_Q
Inserire i messaggi di indirizzamento traccia nella coda denominata fissa.

MQRECORDING_MSG
Inserire i messaggi di indirizzamento traccia in una coda determinata utilizzando il messaggio stesso. Questo è il valore predefinito

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_TRACE_ROUTE_RECORDING con la chiamata MQINQ.

TriggerInterval (MQLONG)

Questo è un intervallo di tempo (in millisecondi) utilizzato per limitare il numero di messaggi trigger. Ciò è rilevante solo quando *TriggerType* è MQTT_FIRST. In questo caso, i messaggi trigger vengono generalmente generati solo quando un messaggio adatto arriva sulla coda e la coda era precedentemente vuota. In determinate circostanze, tuttavia, è possibile generare un ulteriore messaggio trigger con l'attivazione MQTT_FIRST anche se la coda non era vuota. Questi messaggi di trigger aggiuntivi non vengono generati più spesso di ogni *TriggerInterval* millisecondi.

Per ulteriori informazioni sull'attivazione, consultare Trigger dei canali.

Il valore non è inferiore a 0 e non è superiore a 999 999 999. Il valore predefinito è 999 999 999.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_TRIGGER_INTERVAL con la chiamata MQINQ.

TriggerInterval (MQLONG)

Questo è un intervallo di tempo (in millisecondi) utilizzato per limitare il numero di messaggi trigger. Ciò è rilevante solo quando *TriggerType* è MQTT_FIRST. In questo caso, i messaggi trigger vengono generalmente generati solo quando un messaggio adatto arriva sulla coda e la coda era precedentemente vuota. In determinate circostanze, tuttavia, è possibile generare un ulteriore messaggio trigger con l'attivazione MQTT_FIRST anche se la coda non era vuota. Questi messaggi di trigger aggiuntivi non vengono generati più spesso di ogni *TriggerInterval* millisecondi.

Per ulteriori informazioni sull'attivazione, consultare Trigger dei canali.

Il valore non è inferiore a 0 e non è superiore a 999 999 999. Il valore predefinito è 999 999 999.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_TRIGGER_INTERVAL con la chiamata MQINQ.

Versione (MQCFST)

Questa è la versione del codice IBM MQ come VVRRMMFF, dove:

VV - Versione

RR - Rilascio

MM - Livello di manutenzione

FF - Livello di correzione

XrCapability(MQLONG)

Controlla se i comandi MQ Telemetry sono supportati dal gestore code.

Il valore è uno dei seguenti:

MQCAP_SUPPORTATO
Il componente MQ Telemetry installato e i comandi di telemetria sono supportati.

MQCAP_NON_SUPPORTATO
Componente MQ Telemetry non installato.

questo attributo è supportato solo su Multipiattaforme.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_XR_CAPABILITY con la chiamata MQINQ .

Attributi per le code

Esistono cinque tipi di definizione della coda. Alcuni attributi della coda si applicano a tutti i tipi di coda; altri attributi della coda si applicano solo a determinati tipi di coda.

Tipi di coda

Il gestore code supporta i seguenti tipi di definizione della coda:

Coda locale
È possibile memorizzare i messaggi su una coda locale.


Su z/OS è possibile renderla una coda condivisa o privata.

Una coda è nota a un programma come *locale* se è di proprietà del gestore code al quale è connesso il programma. È possibile ricevere messaggi dalle code locali e inserire messaggi nelle code locali.

L'oggetto di definizione della coda contiene le informazioni di definizione della coda, nonché messaggi fisici inseriti nella coda.

Coda gestore code locale
La coda esiste sul gestore code locale.


La coda è nota come coda privata su z/OS.


Coda condivisa (soloz/OS)
La coda esiste in un repository condiviso accessibile a tutti i gestori code appartenenti al gruppo di condivisione code proprietario del repository condiviso.

Le applicazioni connesse a qualsiasi gestore code nel gruppo di condivisione code possono inserire e rimuovere messaggi da code di questo tipo. Tali code sono effettivamente le stesse delle code locali. Il valore dell'attributo coda **QType** è MQQT_LOCAL.

Le applicazioni connesse al gestore code locale possono inserire e rimuovere messaggi da code di questo tipo. Il valore dell'attributo coda **QType** è MQQT_LOCAL.

Coda cluster

È possibile memorizzare i messaggi su una coda cluster nel gestore code in cui è definita. Una coda cluster è una coda ospitata da un gestore code cluster e resa disponibile ad altri gestori code del cluster. Il valore dell'attributo coda **QType** è MQQT_CLUSTER.

Una definizione di coda cluster viene pubblicizzata in altri gestori code nel cluster. Gli altri gestori code nel cluster possono inserire i messaggi in una coda cluster senza la necessità di una definizione di coda remota corrispondente. Una coda cluster può essere pubblicizzata in più di un cluster utilizzando un elenco dei nomi di cluster.

Quando una coda viene pubblicizzata, qualsiasi gestore code del cluster può inserire dei messaggi al suo interno. Per inserire un messaggio, il gestore code deve scoprire, dai repository completi, la posizione in cui è ospitata la coda. Aggiunge quindi alcune informazioni di instradamento al messaggio e inserisce tale messaggio su una coda di trasmissione del cluster.

Un gestore code può memorizzare i messaggi per altri gestori code di un cluster su più code di trasmissione. È possibile configurare un gestore code per memorizzare messaggi su più code di trasmissione cluster in due diversi modi. Se si imposta l'attributo del gestore code **DEFCLXQ** su CHANNEL, viene creata automaticamente una coda di trasmissione cluster differente da SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE per ogni canale mittente del cluster. Se si imposta l'opzione della coda di trasmissione CLCHNAME per trovare la corrispondenza con uno o più canali mittenti del cluster, il gestore code può memorizzare i messaggi per i canali corrispondenti su tale coda di trasmissione.

Attenzione: Se si stanno utilizzando delle SYSTEM.CLUSTER.TRANSMIT.QUEUES dedicate con un gestore code che era stato aggiornato da una versione del prodotto antecedente a IBM WebSphere MQ 7.5, assicurarsi che laSYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE abbia l'opzione SHARE/NOSHARE impostata su **SHARE**.

Una coda del cluster può essere una coda condivisa dai membri di un gruppo di condivisione di code in IBM MQ for z/OS.

Coda remota

Una coda remota non è una coda fisica; è la definizione locale di una coda che esiste su un gestore code remoto. La definizione locale della coda remota contiene informazioni che indicano al gestore code locale come instradare i messaggi al gestore code remoto.

Le applicazioni connesse con il gestore code locale possono inserire i messaggi su code di questo tipo; i messaggi vengono inseriti nella coda di trasmissione locale utilizzata per instradare i messaggi al gestore code remoto. Le applicazioni non possono rimuovere i messaggi dalle code remote. Il valore dell'attributo coda **QType** è MQQT_REMOTE.

È anche possibile utilizzare una definizione di coda remota per:

- Aliasing coda di risposta

  In questo caso, il nome della definizione è il nome di una coda di risposta. Per ulteriori informazioni, consultare Cluster e alias della coda di risposta.

- Alias del gestore code

  In tal caso, il nome della definizione è un alias per un gestore code e non il nome di una coda. Per ulteriori informazioni, fare riferimento alla sezione Cluster e alias dei gestori code.

Coda alias

Questa non è una coda fisica; è un nome alternativo per una coda locale, una coda condivisa, una coda cluster o una coda remota. Il nome della coda in cui l'alias si risolve fa parte della definizione della coda alias.

Le applicazioni connesse al gestore code locale possono inserire messaggi in code di questo tipo; i messaggi vengono inseriti nella coda in cui si risolve l'alias. Le applicazioni possono rimuovere i messaggi dalle code di questo tipo se l'alias si risolve in una coda locale, in una coda condivisa o in una coda cluster che dispone di un'istanza locale. Il valore dell'attributo coda **QType** è MQQT_ALIAS.

Coda modello

Questa non è una coda fisica; è una serie di attributi della coda da cui è possibile creare una coda locale.

I messaggi non possono essere memorizzati su code di questo tipo.

Limiti coda

È possibile configurare e monitorare code che supportano sostanzialmente più del limite predefinito di due terabyte utilizzato nelle release precedenti di IBM MQ. È inoltre possibile ridurre la dimensione di un file di coda.

Per consentire la configurazione delle code, è possibile utilizzare l'attributo **MAXFSIZE** su code locali e modello e, per monitorare code, è possibile utilizzare gli attributi di stato della coda **CURFSIZE** e **CURMAXFS**.

Per ulteriori informazioni, consultare Modifica dei file della coda IBM MQ.

Attributi Coda

Alcuni attributi della coda si applicano a tutti i tipi di coda; altri attributi della coda si applicano solo a determinati tipi di coda. I tipi di coda a cui si applica un attributo sono riportati in Tabella 561 a pagina 860 e nelle tabelle successive.

Tabella 561 a pagina 860 riepiloga gli attributi specifici delle code. Gli attributi sono descritti in ordine alfabetico.

Nota: I nomi degli attributi mostrati in questa sezione sono nomi descrittivi utilizzati con le chiamate MQINQ e MQSET ; i nomi sono gli stessi dei comandi PCF. Quando i comandi MQSC vengono utilizzati per definire, modificare o visualizzare gli attributi, vengono utilizzati nomi brevi alternativi; per i dettagli, consultare Comandi MQSC .

Nella seguente tabella, le colonne si applicano come segue:

- La colonna per le code locali si applica anche alle code condivise.
- La colonna delle code modello indica quali attributi vengono ereditati dalla coda locale creata dalla coda modello.
- La colonna per le code cluster indica gli attributi che possono essere interrogati quando la coda cluster è aperta per l'interrogazione da sola o per l'interrogazione e l'output. Se vengono interrogati altri attributi, la chiamata restituisce il codice di completamento MQCC_WARNING e il codice motivo MQRC_SELECTOR_NOT_FOR_TYPE (2068).

  Se la coda del cluster è aperta per l'interrogazione più uno o più di input, ricerca o impostazione, viene invece applicata la colonna per le code locali.

  Se la coda del cluster è aperta per l'interrogazione da sola, o per l'interrogazione e l'emissione, oltre a specificare il nome del gestore code di base, si applica invece la colonna per le code locali.

Tabella 561. Attributi per le code

Attributo	Descrizione	Locale	Modello	Alias	Remoto	Cluster
AlterationDate	Data dell'ultima modifica della definizione	X		X	X	
AlterationTime	Ora dell'ultima modifica della definizione	X		X	X	
BackoutRequeueQName	Nome coda di riaccodamento di backout eccessivo	X	X			
BackoutThreshold	Soglia di ripristino	X	X			
BaseQName	Nome coda in cui si risolve l'alias			X		
CFStrucName	Nome struttura CF (Coupling Facility)	X	X			
CLCHNAME	Nomi canale mittente del cluster	✓	✓			
ClusterName	Nome del cluster a cui appartiene la coda	X		X	X	X
ClusterNamelist	Nome dell'oggetto elenco nomi contenente i nomi dei cluster a cui appartiene la coda	X		X	X	
CLWLQueuePriority	Priorità coda carico di lavoro cluster	X		X	X	X
CLWLQueueRank	Classificazione coda carico di lavoro cluster	X		X	X	X
CLWLUseQ	Utilizza coda remota	X				
CreationDate	Data di creazione della coda	X				
CreationTime	L'ora in cui è stata creata la coda	X				
CurrentQDepth	Profondità corrente coda	X				
DefaultPutResponse	Risposta inserimento predefinita	✓	✓	✓	✓	
DefBind	Binding predefinito	X		X	X	X
DefinitionType attribute	Il tipo di definizione della coda	X	X			

Tabella 561. Attributi per le code (Continua)

Attributo	Descrizione	Locale	Modello	Alias	Remoto	Cluster
DefInputOpenOption	Opzione predefinita di apertura in immissione	X	X			
DefPersistence	Persistenza predefinita messaggio	X	X	X	X	X
DefPriority	Priorità predefinita messaggio	✓	✓	✓	✓	✓
DefReadAhead	Lettura anticipata predefinita	X	X	X		
DistLists	Supporto elenco di distribuzione	X	X			
HardenGetBackout	Indica se mantenere un conteggio di backout accurato	X	X			
IndexType	Il tipo di indice	X	X			
InhibitGet	Se le operazioni di acquisizione per la coda sono consentite	X	X	X		
InhibitPut	Se le operazioni di inserimento per la coda sono consentite	X	X	X	X	X
InitiationQName	Nome della coda di iniziazione	X	X			
MaxMsgLength	La lunghezza massima del messaggio in byte	X	X			
MaxQDepth	Massima profondità coda	X	X			
MsgDeliverySequence attribute	Sequenza di consegna messaggi	X	X			
NonPersistentMessage Class	Obiettivo di affidabilità per i messaggi non persistenti	X	X			
OpenInputCount	Numero di aperture per input	X				
OpenOutputCount	Numero di aperture per l'emissione	X				
PropertyControl	Controllo delle proprietà	✓	✓	✓		
ProcessName	Nome processo	X	X			
QDepthHighEvent attribute	Indica se vengono generati eventi Grandezza coda elevata	X	X			
QDepthHighLimit	Limite massimo per la profondità della coda	X	X			
QDepthLowEvent attribute	Indica se vengono generati eventi Profondità coda bassa	X	X			
QDepthLowLimit attribute	Limite inferiore per la profondità della coda	X	X			
QDepthMaxEvent	Se vengono generati eventi Coda piena	X	X			
QDesc	Descrizione coda	X	X	X	X	X
QName	Nome coda	X		X	X	X
QServiceInterval	Destinazione per intervallo di servizio coda	X	X			
QServiceIntervalEvent attribute	Se vengono generati eventi Intervallo di servizio elevato o Intervallo di servizio OK	X	X			
QSGDisp attribute	Disposizione del gruppo di condivisione code	X		X	X	
QueueAccounting	Raccolta dati account coda	X	X	X	X	X

Tabella 561. Attributi per le code (Continua)						
Attributo	**Descrizione**	**Locale**	**Modello**	**Alias**	**Remoto**	**Cluster**
QueueMonitoring	Dati di monitoraggio in linea per le code	X	✓			
QueueStatistics	Raccolta dati statistiche coda	X	X	X	X	X
QType	Tipo coda	X		X	X	X
RemoteQMgrName	Nome del gestore code remoto				X	
RemoteQName	Nome della coda remota				X	
RetentionInterval	Intervallo di conservazione	X	X			
Scope	Se una voce per la coda esiste anche in una directory di celle	X		X	X	
Shareability	Condividibilità coda	X	X			
StorageClass	Classe di memoria per la coda	X	X			
TriggerControl	Controllo trigger	X	X			
TriggerData	Dati trigger	X	X			
TriggerDepth	Capacità di Trigger	X	X			
TriggerMsgPriority	Priorità messaggi di soglia per i trigger	X	X			
TriggerType	Tipo trigger	X	X			
Usage attribute	Utilizzo coda	X	X			
XmitQName	Nome coda di trasmissione				X	

Concetti correlati

Code cluster

Code locali

Come scegliere quale tipo di coda di trasmissione del cluster utilizzare

AlterationDate (MQCHAR12)

Data dell'ultima modifica della definizione.

Tabella 562. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X		X	X	

Questa è la data dell'ultima modifica della definizione. Il formato della data è YYYY-MM-DD, riempito con due spazi vuoti finali per rendere la lunghezza di 12 byte (ad esempio, 1992-09-23¬¬, dove ¬¬ rappresenta due caratteri vuoti).

I valori di alcuni attributi (ad esempio, *CurrentQDepth*) cambiano man mano che il gestore code opera. Le modifiche a questi attributi non hanno effetto su *AlterationDate*.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_ALTERATION_DATE con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

L'ora dell'ultima modifica della definizione.

Tabella 563. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X		X	X	

Questa è l'ora dell'ultima modifica della definizione. Il formato dell'ora è HH.MM.SS utilizzando l'orologio di 24 ore, con uno zero iniziale se l'ora è inferiore a 10 (ad esempio 09.10.20).

- Su z/OS, l'ora è GMT (Greenwich Mean Time), soggetto all'orologio di sistema impostato in modo accurato su GMT.
- In altri ambienti, l'ora è l'ora locale.

I valori di alcuni attributi (ad esempio, *CurrentQDepth*) cambiano man mano che il gestore code opera. Le modifiche a questi attributi non hanno effetto su *AlterationTime*.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_ALTERATION_TIME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_TIME_LENGTH.

QName BackoutRequeue(MQCHAR48)

Questo è il nome della coda di backout eccessivo. Oltre a consentire la query del relativo valore, il gestore code non esegue alcuna azione in base al valore di questo attributo.

Tabella 564. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Le applicazioni in esecuzione all'interno di WebSphere Application Server e quelle che utilizzano IBM MQ Application Server Facilities utilizzano questo attributo per determinare dove devono andare i messaggi di cui è stato eseguito il backout. Per tutte le altre applicazioni, il gestore code non esegue alcuna azione in base al valore dell'attributo.

IBM MQ classes for JMS utilizza questo attributo per determinare dove trasferire un messaggio di cui è già stato eseguito il backout il numero massimo di volte specificato dall'attributo *BackoutThreshold* .

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_BACKOUT_REQ_Q_NAME con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da MQ_Q_NAME_LENGTH.

BackoutThreshold (MQLONG)

Questa è la soglia di backout. Oltre a consentire la query del relativo valore, il gestore code non esegue alcuna azione in base al valore di questo attributo.

Tabella 565. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Le applicazioni in esecuzione all'interno di WebSphere Application Server e quelle che utilizzano IBM MQ Application Server Facilities utilizzeranno questo attributo per determinare se è necessario eseguire il backout di un messaggio. Per tutte le altre applicazioni, il gestore code non esegue alcuna azione in base al valore dell'attributo.

IBM MQ classes for JMS utilizza questo attributo per determinare quante volte consentire il backout di un messaggio prima di trasferire il messaggio alla coda specificata dall'attributo *BackoutRequeueQName* .

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_BACKOUT_THRESHOLD con la chiamata MQINQ.

BaseQName (MQCHAR48)

Questo è il nome di una coda definita sul gestore code locale.

Tabella 566. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
		X		

(Per ulteriori informazioni sui nomi delle code, consultare il campo MQOD - ObjectName.) La coda è uno dei seguenti tipi:

LOCALE MQQT
Coda locale.

REMOTE MQQT
Definizione locale di una coda remota.

CLUSTER MQQT_
Coda cluster.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_BASE_Q_NAME con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da MQ_Q_NAME_LENGTH.

BaseType
Il tipo di oggetto in cui l'alias viene risolto.

Tabella 567. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
		X		

È una dei seguenti valori:

MQOT_Q
Il tipo di oggetto di base è una coda

TOPIC MQOT_T
Il tipo di oggetto di base è un argomento

CFStrucName (MQCHAR12)
Questo è il nome della struttura CFS (coupling facility structure) in cui sono memorizzati i messaggi sulla coda. Il primo carattere del nome è compreso tra A e Z, mentre i restanti caratteri sono compresi tra A e Z, tra 0 e 9 o sono vuoti.

Tabella 568. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Per ottenere il nome completo della struttura nella CF (Coupling Facility), aggiungere un suffisso al valore dell'attributo del gestore code **QSGName** con il valore dell'attributo della coda **CFStrucName** .

Questo attributo si applica solo alle code condivise; viene ignorato se *QSGDisp* non ha il valore MQQSGD_SHARED.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_CF_STRUC_NAME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_CF_STRUC_NAME_LENGTH.

z/OS questo attributo è supportato solo su z/OS.

Nome ClusterChannel(MQCHAR20)

ClusterChannelNome è il nome generico dei canali mittenti del cluster che utilizzano questa coda come coda di trasmissione. L'attributo specifica quali canali mittenti del cluster inviano messaggi a un canale ricevente del cluster da questa coda di trasmissione cluster.

Tabella 569. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

La configurazione predefinita del gestore code per tutti i canali mittenti del cluster prevede di inviare i messaggi da una singola coda di trasmissione, SYSTEM.CLUSTER.TRANSMIT.QUEUE. La configurazione predefinita può essere modificata modificando l'attributo del gestore code, **DefClusterXmitQueueType**. Il valore predefinito dell'attributo è SCTQ. È possibile modificare il valore in CHANNEL. Se si imposta l'attributo **DefClusterXmitQueueType** su CHANNEL, ogni canale mittente del cluster utilizza per impostazione predefinita una specifica coda di trasmissione del cluster. SYSTEM.CLUSTER.TRANSMIT.*ChannelName*.

È anche possibile impostare l'attributo della coda di trasmissione ClusterChannelName su un canale mittente del cluster manualmente. I messaggi destinati al gestore code connesso dal canale mittente del cluster vengono memorizzati nella coda di trasmissione che identifica il canale mittente del cluster. Non vengono memorizzati nella coda di trasmissione del cluster predefinita. Se si imposta l'attributo ClusterChannelName su un valore vuoto, il canale passa alla coda di trasmissione del cluster predefinita quando il canale viene riavviato. La coda predefinita è SYSTEM.CLUSTER.TRANSMIT.*ChannelName* o SYSTEM.CLUSTER.TRANSMIT.QUEUE, a seconda del valore dell'attributo del gestore code DefClusterXmitQueueType.

Specificando gli asterischi, "*", in **ClusterChannelName**, è possibile associare una coda di trasmissione a una serie di canali mittenti del cluster. Gli asterischi possono essere all'inizio, alla fine o in qualsiasi numero di posizioni intermedie della stringa di nome canale. **ClusterChannelName** è limitato a una lunghezza di 20 caratteri: MQ_CHANNEL_NAME_LENGTH.

ClusterName (MQCHAR48)

Questo è il nome del cluster a cui appartiene la coda.

Tabella 570. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X		X	X	X

Se la coda appartiene a più di un cluster, *ClusterNamelist* specifica il nome di un oggetto elenco nomi che identifica i cluster e *ClusterName* è vuoto. Almeno uno tra *ClusterName* e *ClusterNamelist* deve essere vuoto.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_CLUSTER_NAME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_CLUSTER_NAME_LENGTH.

ClusterNamelist (MQCHAR48)

Questo è il nome di un oggetto elenco nomi che contiene i nomi dei cluster a cui appartiene questa coda.

Tabella 571. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X		X	X	

Se la coda appartiene ad un solo cluster, l'oggetto elenco nomi contiene un solo nome. In alternativa, *ClusterName* può essere utilizzato per indicare il nome del cluster, nel qual caso *ClusterNamelist* è vuoto. Almeno uno tra *ClusterName* e *ClusterNamelist* deve essere vuoto.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_CLUSTER_NAMELIST con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_NAMELIST_NAME_LENGTH.

CLWLQueuePriority (MQLONG)

Questa è la priorità della coda del carico di lavoro del cluster, un valore compreso tra 0 e 9 che rappresenta la priorità della coda.

Tabella 572. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X		X	X	X

Per ulteriori informazioni, consultare Code cluster.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CLWL_Q_PRIORITY con la chiamata MQINQ.

CLWLQueueRank (MQLONG)

Si tratta della classificazione della coda del carico di lavoro del cluster, un valore compreso tra 0 e 9 che rappresenta la classificazione della coda.

Tabella 573. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X		X	X	X

Per ulteriori informazioni, consultare Code cluster.

Per determinare il valori di questo attributo, utilizzare il selettore MQIA_CLWL_Q_RANK con la chiamata MQINQ.

CLWLUseQ (MQLONG)

Definisce il comportamento di un MQPUT quando la coda di destinazione ha sia un'istanza locale che almeno un'istanza cluster remota. Se l'immissione ha origine da un canale cluster, questo attributo non viene applicato.

Tabella 574. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X				

Il valore è uno dei seguenti:

MQCLWL_USEQ_ANY
Utilizzare code remote e locali.

MQCLWL_USEQ_LOCALE
Non utilizzare code remote.

MQCLWL_USEQ_AS_Q_MGR
Eredita definizione da MQIA_CLWL_USEQ del gestore code.

Per ulteriori informazioni, consultare Code cluster.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CLWL_USEQ con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_CLWL_USEQ_LENGTH.

CreationDate (MQCHAR12)

Questa è la data in cui è stata creata la coda.

Tabella 575. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X				

Il formato della data è YYYY-MM-DD, riempito con due spazi vuoti finali per rendere la lunghezza di 12 byte (ad esempio, 2013-09-23¬¬, dove ¬¬ rappresenta 2 caratteri vuoti).

- Su IBM i, la data di creazione di una coda può essere diversa da quella dell'entità del sistema operativo sottostante (file o spazio utente) che rappresenta la coda.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_CREATION_DATE con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_CREATION_DATE_LENGTH.

CreationTime (MQCHAR8)

Questa è l'ora in cui è stata creata la coda.

Tabella 576. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X				

Il formato dell'ora è HH.MM.SS utilizzando l'orologio di 24 ore, con uno zero iniziale se l'ora è inferiore a 10 (ad esempio 09.10.20).

- Su z/OS, l'ora è GMT (Greenwich Mean Time), soggetto all'orologio di sistema impostato in modo accurato su GMT.
- In altri ambienti, l'ora è l'ora locale.
- Su IBM i, l'orario di creazione di una coda può essere diverso da quello dell'entità del sistema operativo sottostante (file o spazio utente) che rappresenta la coda.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_CREATION_TIME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_CREATION_TIME_LENGTH.

CurrentQDepth (MQLONG)

Questo è il numero di messaggi presenti nella coda.

Tabella 577. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X				

Viene incrementato durante una chiamata MQPUT e durante il backout di una chiamata MQGET. Viene ridotto durante una chiamata MQGET non browse e durante il backout di una chiamata MQPUT. L'effetto è che il conteggio include i messaggi che sono stati inseriti nella coda all'interno di un'unità di lavoro, ma di cui non è stato ancora eseguito il commit, anche se non sono idonei per essere richiamati dalla chiamata MQGET. Allo stesso modo, esclude i messaggi che sono stati richiamati all'interno di un'unità di lavoro utilizzando la chiamata MQGET, ma che devono ancora essere sottoposti a commit.

Il conteggio include anche i messaggi che hanno superato la data di scadenza ma non sono stati ancora eliminati, sebbene questi messaggi non siano idonei per essere richiamati. Per ulteriori informazioni, consultare MQMD - Campo di scadenza .

L'elaborazione dell'unità di lavoro e la segmentazione dei messaggi possono causare il superamento di *MaxQDepth*da parte di *CurrentQDepth* . Tuttavia, ciò non influisce sul richiamo dei messaggi; *tutti* i messaggi sulla coda possono essere richiamati utilizzando la chiamata MQGET nel modo normale.

Il valore di questo attributo varia quando il gestore code opera.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_CURRENT_Q_DEPTH con la chiamata MQINQ.

Risposta DefaultPut(MQLONG)

Specifica il tipo di risposta da utilizzare per le operazioni di inserimento nella coda quando un'applicazione specifica MQPMO_RESPONSE_AS_Q_DEF.

Tabella 578. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X	X	X	

È una dei seguenti valori:

RISPOSTA MQPRT_SYNC_RESPONSE
L'operazione di inserimento viene emessa in modo sincrono, restituendo una risposta.

MQPRT_ASYNC_RESPONSE
L'operazione di inserimento viene eseguita in modo asincrono, restituendo una sottoserie di campi MQMD.

DefBind (MQLONG)

Questo è il binding predefinito che viene utilizzato quando MQOO_BIND_AS_Q_DEF è specificato nella chiamata MQOPEN e la coda è una coda cluster.

Tabella 579. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X		X	X	X

Il valore è uno dei seguenti:

MQBND_BIND_ON_OPEN
Collegamento corretto dalla chiamata MQOPEN.

MQBND_BIND_NO_FIXED
Collegamento non corretto.

MQBND_BIND_ON_XX_ENCODE_CASE_ONE gruppo
Consente a una applicazione di richiedere che un gruppo di messaggi sia assegnato alla stessa istanza di destinazione.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_DEF_BIND con la chiamata MQINQ.

DefinitionType (MQLONG)

Indica come è stata definita la coda.

Tabella 580. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Il valore è uno dei seguenti:

MQQDT_PREDEFINED
La coda è una coda permanente creata dall'amministratore di sistema; solo l'amministratore di sistema può eliminarla.

Le code predefinite vengono create utilizzando il comando MQSC DEFINE e possono essere eliminate solo utilizzando il comando MQSC DELETE . Le code predefinite non possono essere create dalle code modello.

I comandi possono essere emessi da un operatore o da un utente autorizzato che invia un messaggio di comando alla coda di input del comando (per ulteriori informazioni, consultare CommandInputQName attribute).

MQQDT_PERMANENT_DYNAMIC
La coda è una coda permanente creata da un'applicazione che emette una chiamata MQOPEN con il nome di una coda modello specificata nel descrittore oggetto MQOD. La definizione della coda modello aveva il valore MQQDT_PERMANENT_DYNAMIC per l'attributo **DefinitionType** .

Questo tipo di coda può essere eliminato utilizzando la chiamata MQCLOSE. Per ulteriori dettagli, vedere “MQCLOSE - Chiudi oggetto” a pagina 667.

Il valore dell'attributo **QSGDisp** per una coda dinamica continua è MQQSGD_Q_MGR.

MQQDT_TEMPORARY_DYNAMIC
La coda è una coda temporanea creata da un'applicazione che emette una chiamata MQOPEN con il nome di una coda modello specificata nel descrittore oggetto MQOD. La definizione della coda modello aveva il valore MQQDT_TEMPORARY_DYNAMIC per l'attributo **DefinitionType** .

Questo tipo di coda viene eliminato automaticamente dalla chiamata MQCLOSE quando viene chiusa dall'applicazione che l'ha creato.

Il valore dell'attributo **QSGDisp** per una coda dinamica temporanea è MQQSGD_Q_MGR.

MQQDT_SHARED_DYNAMIC
La coda è una coda permanente condivisa creata da un'applicazione che emette una chiamata MQOPEN con il nome di una coda modello specificata nel descrittore oggetto MQOD. La definizione della coda modello aveva il valore MQQDT_SHARED_DYNAMIC per l'attributo **DefinitionType** .

Questo tipo di coda può essere eliminato utilizzando la chiamata MQCLOSE. Per ulteriori dettagli, vedere “MQCLOSE - Chiudi oggetto” a pagina 667.

Il valore dell'attributo **QSGDisp** per una coda dinamica condivisa è MQQSGD_SHARED.

Questo attributo in una definizione di coda modello non indica come è stata definita la coda modello, poiché le code modello sono sempre predefinite. Invece, il valore di questo attributo nella coda modello viene utilizzato per determinare il *DefinitionType* di ciascuna delle code dinamiche create dalla definizione della coda modello utilizzando la chiamata MQOPEN.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_DEFINITION_TYPE con la chiamata MQINQ.

DefInputOpenOption (MQLONG)
Questo è il modo predefinito in cui aprire la coda per l'input.

Tabella 581. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Si applica se l'opzione MQOO_INPUT_AS_Q_DEF viene specificata nella chiamata MQOPEN quando la coda viene aperta. Il valore è uno dei seguenti:

MQOO_INPUT_EXCLUSIVE
Aprire la coda per ottenere i messaggi con accesso esclusivo.

La coda viene aperta per essere utilizzata con successive chiamate MQGET. La chiamata ha esito negativo con codice motivo MQRC_OBJECT_IN_USE se la coda è attualmente aperta da questa o da un'altra applicazione per l'input di qualsiasi tipo (MQOO_INPUT_SHARED o MQOO_INPUT_EXCLUSIVE).

MQOO_INPUT_SHARED
Aprire la coda per richiamare i messaggi con accesso condiviso.

La coda viene aperta per essere utilizzata con successive chiamate MQGET. La chiamata può avere esito positivo se la coda è attualmente aperta da questa o da un'altra applicazione con MQOO_INPUT_SHARED, ma ha esito negativo con codice motivo MQRC_OBJECT_IN_USE se la coda è attualmente aperta con MQOO_INPUT_EXCLUSIVE.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_DEF_INPUT_OPEN_OPTION con la chiamata MQINQ.

DefPersistence (MQLONG)

Questa è la persistenza predefinita dei messaggi sulla coda. Si applica se MQPER_PERSISTENCE_AS_Q_DEF è specificato nel descrittore del messaggio quando il messaggio viene inserito.

Tabella 582. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X	X	X	X

Se è presente più di una definizione nel percorso di risoluzione del nome coda, la persistenza predefinita viene presa dal valore di questo attributo nella *prima* definizione nel percorso al momento della chiamata MQPUT o MQPUT1 . È possibile che si tratti di:

- Una coda alias
- Una coda locale
- Una definizione locale di una coda remota
- Un alias del gestore code
- Una coda di trasmissione (ad esempio, la coda *DefXmitQName*)

Il valore è uno dei seguenti:

PERSISTORA_MQPER_
Il messaggio sopravvive agli errori di sistema e al riavvio del gestore code. I messaggi persistenti non possono essere posizionati su:

- Code dinamiche temporanee
- Code condivise che si associano a un oggetto CFSTRUCT a CFLEVEL (2) o inferiore o dove l'oggetto CFSTRUCT è definito come RECOVER (NO).

I messaggi persistenti possono essere posizionati su code dinamiche permanenti e code predefinite.

MQPER_NOT_PERSISTENT
Il messaggio normalmente non sopravvive agli errori di sistema o al riavvio del gestore code. Ciò si applica anche se una copia intatta del messaggio viene trovata sulla memoria ausiliaria durante un riavvio del gestore code.

Nel caso di code condivise, i messaggi non persistenti *sopravvivono* ai riavvii dei gestori code nel gruppo di condivisione code, ma non sopravvivono agli errori della CF utilizzata per memorizzare i messaggi nelle code condivise.

Sia i messaggi persistenti che quelli non persistenti possono esistere nella stessa coda.

Per stabilire il valore di questo attributo, utilizzare il programma di selezione MQIA_DEF_PERSISTENCE con la chiamata MQINQ.

DefPriority (MQLONG)

Questa è la priorità predefinita per i messaggi sulla coda. Ciò si applica se MQPRI_PRIORITY_AS_Q_DEF viene specificata nel descrittore del messaggio quando il messaggio viene inserito nella coda.

Tabella 583. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X	X	X	X

Se esiste più di una definizione nel percorso di risoluzione del nome coda, la priorità predefinita per il messaggio viene presa dal valore di questo attributo nella *prima* definizione nel percorso al momento dell'operazione di inserimento. È possibile che si tratti di:

- Una coda alias
- Una coda locale
- Una definizione locale di una coda remota
- Un alias del gestore code
- Una coda di trasmissione (ad esempio, la coda *DefXmitQName*)

Il modo in cui un messaggio viene inserito in una coda dipende dal valore dell'attributo **MsgDeliverySequence** della coda:

- Se l'attributo **MsgDeliverySequence** è MQMDS_PRIORITY, la posizione logica in cui un messaggio viene posizionato nella coda dipende dal valore del campo *Priority* nel descrittore del messaggio.
- Se l'attributo **MsgDeliverySequence** è MQMDS_FIFO, i messaggi vengono collocati nella coda come se avessero una priorità uguale al *DefPriority* della coda risolta, indipendentemente dal valore del campo *Priority* nel descrittore del messaggio. Tuttavia, il campo *Priority* conserva il valore specificato dall'applicazione che ha inserito il messaggio. Per ulteriori informazioni, consultare MsgDeliveryMsgDelivery .

Le priorità sono comprese nell'intervallo tra zero (minimo) e *MaxPriority* (massimo); consultare attributoMaxPriority.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_DEF_PRIORITY con la chiamata MQINQ.

DefReadin testa (MQLONG)

Specifica il comportamento di lettura anticipata predefinito per i messaggi non permanenti consegnati al client.

Tabella 584. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X	X		

DefReadPuò essere impostato su uno dei seguenti valori:

MQREADA_NO
I messaggi non persistenti non vengono inviati in anticipo al client prima che un'applicazione li richieda. Un massimo di un messaggio non persistente può andare perduto se il client termina in maniera irregolare.

SÌ MQREADA_
I messaggi non persistenti vengono inviati in anticipo al client prima che un'applicazione li richieda. I messaggi non persistenti possono essere persi se il client termina in modo anomalo o se il client non utilizza tutti i messaggi inviati.

DISABILITA_MQREAD_LED
La lettura anticipata dei messaggi non persistenti non è abilitata per questa coda. I messaggi non vengono inviati in anticipo al client indipendentemente dal fatto che la lettura anticipata sia richiesta dall'applicazione client.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_DEF_READ_AHEAD con la chiamata MQINQ.

DefPResp (MQLONG)

L'attributo tipo di risposta put predefinito (DEFPRESP) definisce il valore utilizzato dalle applicazioni quando il tipo PutResponsein MQPMO è stato impostato su MQPMO_RESPONSE_AS_Q_DEF. Questo attributo è valido per tutti i tipi di coda.

Tabella 585. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X	X	X	X

Il valore è uno dei seguenti:

SINC
L'operazione di inserimento viene emessa in modo sincrono restituendo una risposta.

ASINC
L'operazione di inserimento viene eseguita in modo asincrono, restituendo una sottoserie di campi MQMD.

Per stabilire il valore di questo attributo, utilizzare il selettore MQIA_DEF_PUT_RESPONSE_TYPE con la chiamata MQINQ.

DistLists (MQLONG)

Indica se i messaggi dell'elenco di distribuzione possono essere inseriti nella coda.

Tabella 586. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Un agente MCA (message channel agent) imposta l'attributo per informare il gestore code locale se il gestore code all'altra estremità del canale supporta gli elenchi di distribuzione. Quest' ultimo gestore code (denominato gestore code *partnering*) è quello che riceve successivamente il messaggio, dopo che è stato rimosso dalla coda di trasmissione locale da un MCA mittente.

L'MCA mittente imposta l'attributo ogni volta che stabilisce una connessione all'MCA ricevente sul gestore code partner. In questo modo, l'MCA mittente può far sì che il gestore code locale inserisca nella coda di trasmissione solo i messaggi che il gestore code partner può elaborare correttamente.

Questo attributo è principalmente da utilizzare con le code di trasmissione, ma l'elaborazione descritta viene eseguita indipendentemente dall'utilizzo definito per la coda (consultare Attributo di utilizzo).

Il valore è uno dei seguenti:

MQDL_SUPPORTED
I messaggi dell'elenco di distribuzione possono essere memorizzati nella coda e trasmessi al gestore code partner in tale modulo. Ciò riduce la quantità di elaborazione richiesta per inviare il messaggio a più destinazioni.

MQDL_NOT_SUPPORTED
I messaggi dell'elenco di distribuzione non possono essere memorizzati nella coda, perché il gestore code partner non supporta gli elenchi di distribuzione. Se un'applicazione inserisce un messaggio dell'elenco di distribuzione e tale messaggio deve essere inserito in questa coda, il gestore code suddivide il messaggio dell'elenco di distribuzione e inserisce i singoli messaggi nella coda. Ciò aumenta la quantità di elaborazione richiesta per inviare il messaggio a più destinazioni, ma garantisce che i messaggi vengano elaborati correttamente dal gestore code partner.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_DIST_LISTS con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET.

Questo attributo non è supportato su z/OS.

Backout HardenGet(MQLONG)

Per ogni messaggio, viene mantenuto un conteggio del numero di volte in cui il messaggio viene richiamato da una chiamata MQGET all'interno di un'unità di lavoro e tale unità di lavoro viene successivamente ripristinata.

Tabella 587. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Questo conteggio è disponibile nel campo *BackoutCount* nel descrittore del messaggio una volta completata la chiamata MQGET.

Il conteggio di backout dei messaggi sopravvive ai riavvii del gestore code. Tuttavia, per garantire che il conteggio sia accurato, le informazioni devono essere *rinforzate* (registrate su disco o su un altro dispositivo di archiviazione permanente) ogni volta che una chiamata MQGET richiama un messaggio all'interno di un'unità di lavoro per questa coda. Se questa operazione non viene eseguita, il gestore code ha esito negativo e la chiamata MQGET esegue il backout, il conteggio potrebbe essere incrementato o meno.

Tuttavia, l'irrigidimento delle informazioni per ogni chiamata MQGET all'interno di un'unità di lavoro impone costi di elaborazione aggiuntivi, quindi impostare l'attributo **HardenGetBackout** su MQQA_BACKOUT_HARDENED solo se è essenziale che il conteggio sia accurato.

Su Multipiattaforme, il conteggio di backout del messaggio è sempre forzato, indipendentemente dall'impostazione di questo attributo.

Sono possibili i seguenti valori:

MQQA_BACKOUT_HARDENED
Il potenziamento viene utilizzato per garantire che il numero di backout per i messaggi su questa coda sia accurato.

MQQA_BACKOUT_NOT_HARDENED
Il potenziamento non viene utilizzato per garantire che il numero di backout per i messaggi su questa coda sia accurato. Il conteggio potrebbe quindi essere inferiore a quello che dovrebbe essere.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_HARDEN_GET_BACKOUT con la chiamata MQINQ.

IndexType (MQLONG)

Specifica il tipo di indice che il gestore code conserva per i messaggi sulla coda.

Tabella 588. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Il tipo di indice richiesto dipende dal modo in cui l'applicazione richiama i messaggi e se la coda è una coda condivisa o non condivisa (vedere attributo QSGDisp). I seguenti valori sono possibili per *IndexType*:

MQIT_NONE
Nessun indice viene gestito dal gestore code per questa coda. Utilizzare questo valore per le code generalmente elaborate in modo sequenziale, ovvero senza utilizzare alcun criterio di selezione nella chiamata MQGET.

ID_MSG_MQIT
Il gestore code conserva un indice che utilizza gli ID dei messaggi sulla coda. Utilizzare questo valore per le code in cui l'applicazione in genere richiama i messaggi utilizzando l'identificativo del messaggio come criterio di selezione sulla chiamata MQGET.

ID CORREL_MQIT
Il gestore code conserva un indice che utilizza gli ID di correlazione dei messaggi sulla coda. Utilizzare questo valore per le code in cui l'applicazione di solito richiama i messaggi utilizzando l'identificativo di correlazione come criterio di selezione sulla chiamata MQGET.

MQIT_MSG_TOKEN

Importante: Questo tipo di indice deve essere utilizzato solo per le code utilizzate con il prodotto IBM MQ Workflow for z/OS .

Il gestore code conserva un indice che utilizza i token dei messaggi sulla coda per l'utilizzo con le funzioni WLM (workload manager) di z/OS.

È *necessario* specificare questa opzione per le code gestite da WLM; non specificarla per nessun altro tipo di coda. Inoltre, non utilizzare questo valore per una coda in cui un'applicazione non sta utilizzando le funzioni del gestore del carico di lavoro z/OS , ma sta richiamando i messaggi utilizzando il token del messaggio come criterio di selezione sulla chiamata MQGET.

ID_GROUP_MQIT
Il gestore code conserva un indice che usa gli identificativi di gruppo dei messaggi sulla coda. Questo valore deve essere utilizzato per le code in cui l'applicazione richiama i messaggi utilizzando l'opzione MQGMO_LOGICAL_ORDER sulla chiamata MQGET.

Una coda con questo tipo di indice non può essere una coda di trasmissione. È necessario definire una coda condivisa con questo tipo di indice per associare un oggetto CFSTRUCT a CFLEVEL (3) o superiore.

Nota:

1. L'ordine fisico dei messaggi su una coda con tipo di indice MQIT_GROUP_ID non è definito, poiché la coda è ottimizzata per il richiamo efficiente dei messaggi utilizzando l'opzione MQGMO_LOGICAL_ORDER sulla chiamata MQGET. Ciò significa che l'ordine fisico dei messaggi non è in genere l'ordine in cui i messaggi sono arrivati sulla coda.
2. Se una coda MQIT_GROUP_ID ha un *MsgDeliverySequence* di MQMDS_PRIORITY, il gestore code utilizza le priorità dei messaggi 0 e 1 per ottimizzare il richiamo dei messaggi in ordine logico. Di conseguenza, il primo messaggio in un gruppo non deve avere una priorità pari a zero o uno; in tal caso, il messaggio viene elaborato come se avesse una priorità pari a due. Il campo *Priority* nella struttura MQMD non viene modificato.

Per ulteriori informazioni sui gruppi di messaggi, consultare la descrizione delle opzioni del segmento e del gruppo nel campo MQGMO - Opzioni.

Il tipo di indice che deve essere utilizzato in vari casi viene mostrato in Tabella 589 a pagina 874 e Tabella 590 a pagina 875.

Tabella 589. Valori suggeriti o richiesti del tipo di indice della coda quando MQGMO_LOGICAL_ORDER non è specificato

Criteri di selezione sulla chiamata MQGET	**Tipo di indice per la coda non condivisa**	**Tipo di indice per la coda condivisa**
Nessuna	Qualsiasi	Qualsiasi
Selezione utilizzando un identificativo:		
ID messaggio	MQIT_MSG_ID suggerito	MQIT_NONE o MQIT_MSG_ID richiesto; MQIT_MSG_ID suggerito
Identificativo di correlazione	MQIT_CORREL_ID suggerito	MQIT_CORREL_ID richiesto
ID gruppo	MQIT_GROUP_ID suggerito	MQIT_GROUP_ID richiesto
Selezione mediante due identificatori:		

Tabella 589. Valori suggeriti o richiesti del tipo di indice della coda quando MQGMO_LOGICAL_ORDER non è specificato (Continua)

Criteri di selezione sulla chiamata MQGET	**Tipo di indice per la coda non condivisa**	**Tipo di indice per la coda condivisa**
Identificativo del messaggio più identificativo di correlazione	MQIT_MSG_ID o MQIT_CORREL_ID suggeriti	MQIT_NONE o MQIT_MSG_ID o MQIT_CORREL_ID richiesti (Per una maggiore efficienza, si consiglia di scegliere il tipo di indice in modo che corrisponda al campo MQMD che avrà le chiavi più distinte)
Identificativo messaggio più identificativo gruppo	MQIT_MSG_ID o MQIT_GROUP_ID suggeriti	Non supportato
Identificativo di correlazione più identificativo di gruppo	MQIT_CORREL_ID o MQIT_GROUP_ID suggeriti	Non supportato
Selezione utilizzando tre identificatori:		
Identificativo del messaggio più identificativo di correlazione più identificativo del gruppo	MQIT_MSG_ID o MQIT_CORREL_ID o MQIT_GROUP_ID suggeriti	Non supportato
Selezione mediante criteri relativi al gruppo:		
Identificativo del gruppo più numero di sequenza del messaggio	MQIT_GROUP_ID richiesto	MQIT_GROUP_ID richiesto
Numero di sequenza del messaggio (deve essere 1)	MQIT_GROUP_ID richiesto	MQIT_GROUP_ID richiesto
Selezione mediante token di messaggio:		
Token del messaggio per l'utilizzo dell'applicazione	Non utilizzare MQIT_MSG_TOKEN	
Token del messaggio per l'utilizzo di WLM	MQIT_MSG_TOKEN richiesto	Non supportato

Tabella 590. Valori suggeriti o richiesti del tipo di indice della coda quando è specificato MQGMO_LOGICAL_ORDER

Criteri di selezione sulla chiamata MQGET	**Tipo di indice per la coda non condivisa**	**Tipo di indice per la coda condivisa**
Nessuna	MQIT_GROUP_ID richiesto	MQIT_GROUP_ID richiesto
Selezione utilizzando un identificativo:		
ID messaggio	MQIT_GROUP_ID richiesto	Non supportato
Identificativo di correlazione	MQIT_GROUP_ID richiesto	Non supportato
ID gruppo	MQIT_GROUP_ID richiesto	MQIT_GROUP_ID richiesto
Selezione mediante due identificatori:		

Tabella 590. Valori suggeriti o richiesti del tipo di indice della coda quando è specificato MQGMO_LOGICAL_ORDER (Continua)		
Criteri di selezione sulla chiamata MQGET	**Tipo di indice per la coda non condivisa**	**Tipo di indice per la coda condivisa**
Identificativo del messaggio più identificativo di correlazione	MQIT_GROUP_ID richiesto	Non supportato
Identificativo messaggio più identificativo gruppo	MQIT_GROUP_ID richiesto	Non supportato
Identificativo di correlazione più identificativo di gruppo	MQIT_GROUP_ID richiesto	Non supportato
Selezione utilizzando tre identificatori:		
Identificativo del messaggio più identificativo di correlazione più identificativo del gruppo	MQIT_GROUP_ID richiesto	Non supportato

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_INDEX_TYPE con la chiamata MQINQ.

z/OS questo attributo è supportato solo su z/OS.

InhibitGet (MQLONG)

Controlla se le operazioni di richiamo per questa coda sono consentite.

Tabella 591. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X	X		

Se la coda è una coda alias, le operazioni get devono essere consentite sia per l'alias che per la coda di base al momento dell'operazione get, affinché la chiamata MQGET abbia esito positivo. Il valore è uno dei seguenti:

MQQA_GET_INIBITO
Le operazioni get sono inibite.

Le chiamate MQGET hanno esito negativo con codice motivo MQRC_GET_INHIBITED. Sono incluse chiamate MQGET che specificano MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT.

Nota: Se una chiamata MQGET che opera all'interno di un'unità di lavoro viene completata correttamente, la modifica del valore dell'attributo **`InhibitGet`** successivamente in MQQA_GET_INIBITED non impedisce il commit dell'unità di lavoro.

MQQA_GET_ALLOWED
Le operazioni get sono consentite.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_INIB_GET con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET.

InhibitPut (MQLONG)

Controlla se le operazioni di inserimento per questa coda sono consentite.

Tabella 592. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X	X	X	X

Se è presente più di una definizione nel percorso di risoluzione del nome della coda, le operazioni di inserimento devono essere consentite per *ogni* definizione nel percorso (incluse tutte le definizioni di alias del gestore code) al momento dell'operazione di inserimento, affinché la chiamata MQPUT o MQPUT1 abbia esito positivo. Il valore è uno dei seguenti:

MQQA_PUT_INIBITO
Le operazioni di inserimento sono inibite.

Le chiamate MQPUT e MQPUT1 hanno esito negativo con codice motivo MQRC_PUT_INIBITED.

Nota: Se una chiamata MQPUT che opera all'interno di un'unità di lavoro viene completata correttamente, la modifica del valore dell'attributo **InhibitPut** successivamente in MQQA_PUT_INIBITED non impedisce il commit dell'unità di lavoro.

MQQA_PUT_CONSENTITO
Le operazioni di inserimento sono consentite.

Per determinare il valore di questo attributo, utilizzare l'utilità di selezione MQIA_INIB_PUT con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET.

InitiationQName (MQCHAR48)

Questo è il nome di una coda definita sul gestore code locale; la coda deve essere di tipo MQQT_LOCAL.

Tabella 593. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X				

Il gestore code invia un messaggio trigger alla coda di iniziazione quando è richiesto l'avvio dell'applicazione come risultato dell'arrivo di un messaggio sulla coda a cui appartiene questo attributo. La coda di iniziazione deve essere monitorata da un'applicazione di controllo trigger che avvia l'applicazione appropriata dopo aver ricevuto il messaggio di trigger.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_INITIATION_Q_NAME con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da MQ_Q_NAME_LENGTH.

Lunghezza MaxMsg(MQLONG)

Questo è un limite superiore per la lunghezza del messaggio *fisico* più lungo che può essere inserito nella coda.

Tabella 594. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Tuttavia, poiché l'attributo della coda **MaxMsgLength** può essere impostato indipendentemente dall'attributo del gestore code **MaxMsgLength** , il limite superiore effettivo per la lunghezza del messaggio fisico più lungo che può essere inserito nella coda è il minore di questi due valori.

Se il gestore code supporta la segmentazione, è possibile per un'applicazione inserire un messaggio *logico* più lungo del minore dei due attributi **MaxMsgLength** , ma solo se l'applicazione specifica l'indicatore MQMF_SEGMENTATION_ALLOWED in MQMD. Se viene specificato tale indicatore, il limite superiore per la lunghezza di un messaggio logico è 999 999 999 999 byte, ma di solito i vincoli di risorsa imposti dal sistema operativo o dall'ambiente in cui l'applicazione è in esecuzione, risultano in un limite inferiore.

Un tentativo di inserire nella coda un messaggio troppo lungo ha esito negativo con uno dei seguenti codici di errore:

- MQRC_MSG_TOO_BIG_FOR_Q se il messaggio è troppo grande per la coda
- MQRC_MSG_TOO_BIG_FOR_Q_MGR se il messaggio è troppo grande per il gestore code, ma non troppo grande per la coda

Il limite inferiore per l'attributo **MaxMsgLength** è zero; il limite superiore è 100 MB (104 857 600 byte).

Per ulteriori informazioni, consultare il parametro MQPUT - BufferLength.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_MAX_MSG_LENGTH con la chiamata MQINQ.

MaxQDepth (MQLONG)

Si tratta del limite superiore definito per il numero di messaggi fisici che possono esistere sulla coda contemporaneamente.

Tabella 595. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Un tentativo di inserire un messaggio su una coda che contiene già messaggi **MaxQDepth** ha esito negativo con codice motivo MQRC_Q_FULL.

L'elaborazione dell'unità di lavoro e la segmentazione dei messaggi possono causare il superamento di **MaxQDepth**da parte del numero effettivo di messaggi fisici sulla coda. Tuttavia, ciò non influisce sul richiamo del messaggio poiché tutti i messaggi sulla coda possono essere richiamati utilizzando la chiamata MQGET.

Il valore di questo attributo è zero o maggiore. Il limite superiore è determinato dall'ambiente:

- Sulle seguenti piattaforme, il valore non può superare 999 999 999:
  - AIX
  - Linux
  - Windows
  - z/OS
- IBM i Su IBM i, il valore non può superare 640 000.

Nota: Lo spazio di memoria disponibile per la coda potrebbe essere esaurito anche se nella coda sono presenti meno di **MaxQDepth** messaggi.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_MAX_Q_DEPTH con la chiamata MQINQ.

Sequenza MsgDelivery(MQLONG)

Tabella 596. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Ciò determina l'ordine in cui la chiamata MQGET restituisce i messaggi all'applicazione:

FIFO MQMDS
I messaggi vengono restituiti in ordine FIFO (first in, first out).

Una chiamata MQGET restituisce il *primo* messaggio che soddisfa i criteri di selezione specificati nella chiamata, indipendentemente dalla priorità del messaggio.

PRIORITà_MQMDS
I messaggi vengono restituiti in ordine di priorità.

Una chiamata MQGET restituisce il messaggio *priorità più alta* che soddisfa i criteri di selezione specificati nella chiamata. All'interno di ciascun livello di priorità, i messaggi vengono restituiti in ordine FIFO (first in, first out).

- Su z/OS, se la coda ha un *IndexType* di MQIT_GROUP_ID, l'attributo **MsgDeliverySequence** specifica l'ordine in cui i gruppi di messaggi vengono restituiti all'applicazione. La particolare sequenza in cui vengono restituiti i gruppi è determinata dalla posizione o dalla priorità del primo messaggio in ciascun gruppo. L'ordine fisico dei messaggi nella coda non viene definito, poiché la coda è ottimizzata per il richiamo efficiente dei messaggi utilizzando l'opzione MQGMO_LOGICAL_ORDER nella chiamata MQGET.
- Su z/OS, se *IndexType* è MQIT_GROUP_ID e *MsgDeliverySequence* è MQMDS_PRIORITY, il gestore code utilizza le priorità dei messaggi zero e una per ottimizzare il richiamo dei messaggi in ordine logico. Di conseguenza, il primo messaggio in un gruppo non deve avere una priorità pari a zero o uno; in tal caso, il messaggio viene elaborato come se avesse una priorità pari a due. Il campo *Priority* nella struttura MQMD non viene modificato.

Se gli attributi rilevanti vengono modificati mentre sono presenti messaggi nella coda, la sequenza di consegna è la seguente:

- L'ordine in cui i messaggi vengono restituiti dalla chiamata MQGET è determinato dai valori degli attributi di **MsgDeliverySequence** e **DefPriority** in vigore per la coda nel momento in cui il messaggio arriva nella coda:
  - Se *MsgDeliverySequence* è MQMDS_FIFO quando arriva il messaggio, il messaggio viene inserito nella coda come se la sua priorità fosse *DefPriority*. Ciò non influisce sul valore del campo *Priority* nel descrittore del messaggio; quel campo conserva il valore che aveva quando il messaggio è stato inserito per la prima volta.
  - Se *MsgDeliverySequence* è MQMDS_PRIORITY quando arriva il messaggio, il messaggio viene posizionato nella coda nel punto appropriato alla priorità data dal campo *Priority* nel descrittore del messaggio.

  Se il valore dell'attributo **MsgDeliverySequence** viene modificato mentre sono presenti messaggi nella coda, l'ordine dei messaggi nella coda non viene modificato.

  Se il valore dell'attributo **DefPriority** viene modificato mentre ci sono messaggi nella coda, i messaggi non vengono necessariamente consegnati in ordine FIFO, anche se l'attributo **MsgDeliverySequence** è impostato su MQMDS_FIFO; quelli che sono stati inseriti nella coda con la priorità più alta vengono consegnati per primi.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_MSG_DELIVERY_SEQUENCE con la chiamata MQINQ.

NonPersistentMessageClass (MQLONG)

L'obiettivo di affidabilità per i messaggi non persistenti.

Tabella 597. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Specifica le circostanze in cui i messaggi non persistenti inseriti in questa coda vengono eliminati:

MQNPM_CLASS_NORMAL
I messaggi non persistenti sono limitati alla durata della sessione del gestore code; i messaggi vengono eliminati in caso di riavvio del gestore code. Questo è valido solo per le code non condivise ed è il valore predefinito.

MQNPM_CLASS_HIGH
Il gestore code tenta di conservare i messaggi non persistenti per la durata della coda. I messaggi non persistenti potrebbero ancora essere persi in caso di errore. Questo valore viene applicato per le code condivise.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_NPM_CLASS con la chiamata MQINQ.

Conteggio OpenInput(MQLONG)
Questo è il numero di handle attualmente validi per la rimozione di messaggi dalla coda mediante la chiamata MQGET.

Tabella 598. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X				

È il numero totale di tali handle noti al gestore code *locale* . Se la coda è una coda condivisa, il conteggio non include le aperture per l'input eseguite per la coda su altri gestori code nel gruppo di condivisione code a cui appartiene il gestore code locale.

Il conteggio include gli handle in cui una coda alias che si risolve in questa coda è stata aperta per l'input. Il conteggio non include gli handle in cui la coda è stata aperta per le azioni che non includevano l'input (ad esempio, una coda aperta solo per la ricerca).

Il valore di questo attributo varia quando il gestore code opera.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_OPEN_INPUT_COUNT con la chiamata MQINQ.

Conteggio OpenOutput(MQLONG)
Questo è il numero di handle attualmente validi per aggiungere messaggi alla coda mediante la chiamata MQPUT.

Tabella 599. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X				

È il numero totale di tali handle noti al gestore code *locale* ; non include le aperture per l'output eseguite per questa coda sui gestori code remoti. Se la coda è una coda condivisa, il conteggio non include le aperture per l'output eseguite per la coda su altri gestori code nel gruppo di condivisione code a cui appartiene il gestore code locale.

Il conteggio include gli handle in cui una coda alias che si risolve in questa coda è stata aperta per l'emissione. Il conteggio non include gli handle in cui la coda è stata aperta per le azioni che non includevano l'output (ad esempio, una coda aperta solo per l'interrogazione).

Il valore di questo attributo varia quando il gestore code opera.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_OPEN_OUTPUT_COUNT con la chiamata MQINQ.

ProcessName (MQCHAR48)
Questo è il nome di un oggetto processo definito nel gestore code locale. L'oggetto processo identifica un programma che può servire la coda.

Tabella 600. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_PROCESS_NAME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_PROCESS_NAME_LENGTH.

PropertyControl (MQLONG)

Specifica in che modo vengono gestite le proprietà del messaggio per i messaggi richiamati dalle code utilizzando la chiamata MQGET con l'opzione MQGMO_PROPERTIES_AS_Q_DEF.

Tabella 601. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X	X		

Il valore è uno dei seguenti:

TUTTE le MQPROP
Tutte le proprietà del messaggio sono incluse con il messaggio quando viene consegnato all'applicazione. Le proprietà, eccetto quelle nel descrittore di messaggi (o estensione) vengono collocate in una o più intestazioni MQRFH2 nei dati del messaggio. Se viene fornito un handle del messaggio, il funzionamento è quello di restituire le proprietà nel handle del messaggio.

COMPATIBILITÀ_MQPROP_
Se il messaggio contiene una proprietà con un prefisso mcd., jms., usr. o mqext., tutte le propriet ... del messaggio vengono consegnate all'applicazione in una intestazione MQRFH2 . Altrimenti tutte le proprietà del messaggio, eccetto quelle contenute nel descrittore messaggi (o nell'estensione), vengono eliminate e non sono più accessibili sull'applicazione. Questo è il valore predefinito; consente alle applicazioni che prevedono che le proprietà correlate a JMS si trovano in un'intestazione MQRFH2 nei dati del messaggio di continuare a funzionare senza modifiche. Se viene fornito un handle del messaggio, il funzionamento consiste nel restituire le proprietà nell'handle del messaggio.

MQPROP_FORCE_MQRFH2
Le proprietà vengono sempre restituite nei dati del messaggio in un'intestazione MQRFH2, a prescindere dal fatto che l'applicazione specifichi o meno un gestore messaggi. Un handle del messaggio valido fornito nel campo MsgHandle della struttura MQGMO sulla chiamata MQGET viene ignorato. Le proprietà del messaggio non sono accessibili attraverso la gestione del messaggio.

MQPROP_NONE
Tutte le proprietà del messaggio, tranne quelle nel descrittore del messaggio (o estensione), vengono rimosse dal messaggio prima che il messaggio venga consegnato all'applicazione. Se viene fornito un handle del messaggio, il funzionamento è quello di restituire le proprietà nel handle del messaggio.

Questo parametro è applicabile alle code Locale, Alias e Modello. Per determinare il valore, utilizzare il selettore MQIA_PROPERTY_CONTROL con la chiamata MQINQ.

Evento QDepthHigh(MQLONG)

Controlla se vengono generati eventi Grandezza coda elevata.

Tabella 602. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Un evento Profondità massima coda indica che un'applicazione ha inserito un messaggio in una coda e ciò ha fatto sì che il numero di messaggi nella coda diventasse maggiore o uguale alla soglia massima di profondità della coda (consultare l'attributo **QDepthHighLimit**).

Nota: Il valore di questo attributo può essere modificato dinamicamente.

Il valore è uno dei seguenti:

DISABILITAZIONE_MQEVR
Report eventi disabilitato.

MQEVR_ENABLED
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_Q_DEPTH_HIGH_EVENT con la chiamata MQINQ.

Questo attributo è supportato su z/OS, ma non è possibile utilizzare la chiamata MQINQ per determinarne il valore.

Limite QDepthHigh(MQLONG)

Si tratta della soglia rispetto alla quale la profondità della coda viene confrontata per generare un evento Grandezza coda elevata.

Tabella 603. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Questo evento indica che un'applicazione ha inserito un messaggio in una coda e che ciò ha fatto sì che il numero di messaggi nella coda diventasse maggiore o uguale alla soglia superiore della profondità della coda. Vedere QDepthHigh.

Il valore è espresso come percentuale della profondità massima della coda (attributo**MaxQDepth**) ed è maggiore o uguale a 0 e minore o uguale a 100. Il valore predefinito è 80.

Per determinare il valore di questo attributo, utilizzare il selettore QIA_Q_DEPTH_HIGH_LIMIT con la chiamata MQINQ.

Questo attributo è supportato su z/OS, ma non è possibile utilizzare la chiamata MQINQ per determinarne il valore.

Evento QDepthLow(MQLONG)

Controlla se vengono generati eventi Profondità coda bassa.

Tabella 604. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Un evento Profondità minima coda indica che un'applicazione ha richiamato un messaggio da una coda e che ciò ha fatto sì che il numero di messaggi nella coda diventasse inferiore o uguale alla soglia minima di profondità coda (consultare QDepthLow).

Nota: Il valore di questo attributo può essere modificato dinamicamente.

Il valore è uno dei seguenti:

DISABILITAZIONE_MQEVR
Report eventi disabilitato.

MQEVR_ENABLED
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_Q_DEPTH_LOW_EVENT con la chiamata MQINQ.

Questo attributo è supportato su z/OS, ma non è possibile utilizzare la chiamata MQINQ per determinarne il valore.

Limite QDepthLow(MQLONG)

Questa è la soglia rispetto alla quale la profondità della coda viene confrontata per generare un evento Grandezza coda bassa.

Tabella 605. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Questo evento indica che un'applicazione ha richiamato un messaggio da una coda e che ciò ha fatto sì che il numero di messaggi nella coda diventasse inferiore o uguale alla soglia inferiore della profondità della coda. Vedere QDepthLow.

Il valore è espresso come percentuale della profondità massima della coda (attributo**MaxQDepth**) ed è maggiore o uguale a 0 e minore o uguale a 100. Il valore predefinito è 20.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_Q_DEPTH_LOW_LIMIT con la chiamata MQINQ.

Questo attributo è supportato su z/OS, ma non è possibile utilizzare la chiamata MQINQ per determinarne il valore.

Evento QDepthMax(MQLONG)

Controlla se vengono generati eventi Coda piena. Un evento Coda piena indica che un inserimento in una coda è stato rifiutato perché la coda è piena, ovvero la profondità della coda ha già raggiunto il valore massimo.

Tabella 606. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Nota: Il valore di questo attributo può essere modificato dinamicamente.

Il valore è uno dei seguenti:

DISABILITAZIONE_MQEVR
Report eventi disabilitato.

MQEVR_ENABLED
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_Q_DEPTH_MAX_EVENT con la chiamata MQINQ.

Questo attributo è supportato su z/OS, ma non è possibile utilizzare la chiamata MQINQ per determinarne il valore.

Descrizione coda (MQCHAR64)

Utilizzare questo campo per commenti descrittivi.

Tabella 607. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X	X	X	X

Il contenuto del campo non ha alcun significato per il gestore code, ma il gestore code potrebbe richiedere che il campo contenga solo caratteri che possono essere visualizzati. Non può contenere

caratteri null; se necessario, viene riempito a destra con spazi. In un'installazione DBCS, il campo può contenere caratteri DBCS (con una lunghezza massima di 64 byte).

Nota: Se questo campo contiene caratteri non presenti nella serie di caratteri del gestore code (come definito dall'attributo del gestore code **CodedCharSetId**), tali caratteri potrebbero essere tradotti in modo non corretto se questo campo viene inviato a un altro gestore code.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_Q_DESC con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_Q_DESC_LENGTH.

QName (MQCHAR48)

Questo è il nome di una coda definita sul gestore code locale.

Tabella 608. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X		X	X	X

Tutte le code definite su un gestore code condividono lo stesso spazio dei nomi della coda. Pertanto, una coda MQQT_LOCAL e una coda MQQT_ALIAS non possono avere lo stesso nome.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_Q_NAME con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da MQ_Q_NAME_LENGTH.

QServiceInterval (MQLONG)

Questo è l'intervallo di servizio utilizzato per il confronto per generare eventi Intervallo di servizio elevato e Intervallo di servizio OK.

Tabella 609. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Consultare QServiceInterval.

Il valore è in unità di millisecondi ed è maggiore o uguale a zero e minore o uguale a 999 999 999.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_Q_SERVICE_INTERVAL con la chiamata MQINQ.

Questo attributo è supportato su z/OS, ma non è possibile utilizzare la chiamata MQINQ per determinarne il valore.

Evento QServiceInterval(MQLONG)

Controlla se vengono generati eventi Intervallo di servizio elevato o Intervallo di servizio OK.

Tabella 610. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

- Un evento Intervallo servizio elevato viene generato quando un controllo indica che non sono stati richiamati messaggi dalla coda per almeno il tempo indicato dall'attributo **QServiceInterval** .
- Un evento Intervallo di servizio OK viene generato quando un controllo indica che i messaggi sono stati richiamati dalla coda entro il tempo indicato dall'attributo **QServiceInterval** .

Nota: Il valore di questo attributo può essere modificato dinamicamente.

Il valore è uno dei seguenti:

MQQSIE_HIGH
Eventi di intervallo del servizio coda alto abilitati.

- Gli eventi Intervallo servizio coda elevato sono **abilitati** e
- Gli eventi OK dell'intervallo di servizio della coda sono **disabilitati**.

MQQSIE_OK
Eventi di intervallo del servizio coda OK abilitati.

- Gli eventi Intervallo servizio coda elevato sono **disabilitati** e
- Gli eventi OK dell'intervallo di servizio della coda sono **abilitati**.

MQQSIE_NONE
Nessun evento di intervallo del servizio coda abilitato.

- Gli eventi Intervallo servizio coda elevato sono **disabilitati** e
- Gli eventi OK dell'intervallo di servizio della coda sono **disabilitati**.

Per le code condivise, il valore di questo attributo viene ignorato; viene assunto il valore MQQSIE_NONE.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_Q_SERVICE_INTERVAL_EVENT con la chiamata MQINQ.

Su z/OS, non è possibile utilizzare la chiamata MQINQ per determinare il valore di questo attributo.

QSGDisp (MQLONG)

Specifica la disposizione della coda.

Tabella 611. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X		X	X	

Il valore è uno dei seguenti:

MQQSGD_Q_MGR
L'oggetto ha la disposizione del gestore code. Ciò significa che la definizione dell'oggetto è nota solo al gestore code locale; la definizione non è nota ad altri gestori code nel gruppo di condivisione code.

Ogni gestore code nel gruppo di condivisione code può avere un oggetto con lo stesso nome e tipo dell'oggetto corrente, ma si tratta di oggetti separati e non vi è alcuna correlazione tra loro. I loro attributi non sono vincolati ad essere gli stessi.

MQQSGD_XX_ENCODE_CASE_ONE copia
L'oggetto è una copia locale di una definizione di oggetto principale che esiste nel repository condiviso. Ogni gestore code nel gruppo di condivisione code può avere la propria copia dell'oggetto. Inizialmente, tutte le copie hanno gli stessi attributi, ma utilizzando i comandi MQSC, è possibile modificare ciascuna copia in modo che i suoi attributi differiscano da quelli delle altre copie. Gli attributi delle copie vengono risincronizzati quando la definizione principale nel repository condiviso viene modificata.

MQQSGD_SHARED
L'oggetto presenta una disposizione condivisa. Ciò significa che esiste nel repository condiviso una singola istanza dell'oggetto nota a tutti i gestori code nel gruppo di condivisione code. Quando un gestore code nel gruppo accede all'oggetto, accede alla sola istanza condivisa dell'oggetto.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_QSG_DISP con la chiamata MQINQ.

z/OS questo attributo è supportato solo su z/OS.

QueueAccounting (MQLONG)

Tabella 612. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X	X	X	

Controlla la raccolta dei dati di account per la coda. Per i dati di account da raccogliere per questa coda, è necessario abilitare anche i dati di account per questa connessione, utilizzando l'attributo QMGR ACCTQ o il campo Opzioni nella struttura MQCNO nella chiamata MQCONNX.

Questo attributo può presentare uno dei seguenti valori:

MGR MQMON_Q
I dati di account per questa coda vengono raccolti in base all'impostazione dell'attributo QMGR ACCTQ. Questa è l'impostazione predefinita.

MQMON_DISATTIVO
Non raccogliere i dati di account per questa coda.

MMON_UN
Raccogliere i dati di account per questa coda.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_ACCOUNTING_Q con la chiamata MQINQ.

QueueMonitoring (MQLONG)

Controlla la raccolta dei dati di controllo online per le code.

Tabella 613. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Il valore è uno dei seguenti:

MGR MQMON_Q
Raccogliere i dati di monitoraggio in base alle impostazioni dell'attributo del gestore code **QueueMonitoring** . Questo è il valore predefinito.

MQMON_DISATTIVO
La raccolta dati di monitoraggio in linea è disattivata per questa coda.

MMON_LOW
Se il valore dell'attributo del gestore code **QueueMonitoring** non è MQMON_NONE, la raccolta dei dati di monitoraggio in linea è attivata, con una frequenza bassa di raccolta dati per questa coda.

MQMON_MEDIO
Se il valore dell'attributo del gestore code **QueueMonitoring** non è MQMON_NONE, la raccolta dati di monitoraggio in linea è attivata, con una frequenza moderata di raccolta dati per questa coda.

MQMON_HIGH
Se il valore dell'attributo del gestore code **QueueMonitoring** non è MQMON_NONE, la raccolta dati di monitoraggio in linea è attivata, con una frequenza elevata di raccolta dati per questa coda.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_MONITORING_Q con la chiamata MQINQ.

QueueStatistics (MQCHAR12)

Tabella 614. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X	X	X	

Controlla la raccolta dei dati statistici per la coda.

Questo attributo può presentare uno dei seguenti valori:

MGR MQMON_Q
I dati di account per questa coda vengono raccolti in base all'impostazione dell'attributo QMGR STATQ. Questa è l'impostazione predefinita.

MQMON_DISATTIVO
Disattivare la raccolta dati delle statistiche per questa coda.

MMON_UN
Abilitare la raccolta dati delle statistiche per questa coda.

Tipo di coda (MQLONG)

Tabella 615. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X		X	X	X

Questo è il tipo di coda; ha uno dei seguenti valori:

ALIAS MQQT
Definizione coda alias.

CLUSTER MQQT_
Coda cluster.

LOCALE MQQT
Coda locale.

REMOTE MQQT
Definizione locale di una coda remota.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_Q_TYPE con la chiamata MQINQ.

RemoteQMgrNome (MQCHAR48)

Tabella 616. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
			X	

Questo è il nome del gestore code remoto su cui è definita la coda **RemoteQName** . Se la coda **RemoteQName** ha un valore **QSGDisp** di MQQSGD_COPY o MQQSGD_SHARED, **RemoteQMgrName** può essere il nome del gruppo di condivisione code che possiede **RemoteQName**.

Se un'applicazione apre la definizione locale di una coda remota, **RemoteQMgrName** non deve essere vuoto e non deve essere il nome del gestore code locale. Se **XmitQName** è vuoto, la coda locale con lo stesso nome di **RemoteQMgrName** viene utilizzata come coda di trasmissione. Se non è presente alcuna coda con il nome **RemoteQMgrName**, viene utilizzata la coda identificata dall'attributo del gestore code **DefXmitQName** .

Se questa definizione viene utilizzata per un alias del gestore code, **RemoteQMgrName** è il nome del gestore code di cui si sta eseguendo l'alias. Può essere il nome del gestore code locale. Altrimenti, se **XmitQName** è vuoto quando si verifica l'apertura, deve essere presente una coda locale con un nome uguale a **RemoteQMgrName**; questa coda viene utilizzata come coda di trasmissione.

Se questa definizione viene utilizzata per un alias reply - to, questo nome è il nome del gestore code che deve essere **ReplyToQMgr**.

Nota: Non viene eseguita alcuna convalida sul valore specificato per questo attributo quando la definizione della coda viene creata o modificata.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_REMOTE_Q_MGR_NAME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_Q_MGR_NAME_LENGTH.

RemoteQName (MQCHAR48)

Tabella 617. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
			X	

Questo è il nome della coda come è noto sul gestore code remoto *RemoteQMgrName*.

Se un'applicazione apre la definizione locale di una coda remota, quando si verifica l'apertura *RemoteQName* non deve essere vuoto.

Se questa definizione viene utilizzata per una definizione di alias del gestore code, quando si verifica l'apertura *RemoteQName* deve essere vuoto.

Se la definizione viene utilizzata per un alias di risposta, questo nome è il nome della coda che deve essere *ReplyToQ*.

Nota: Non viene eseguita alcuna convalida sul valore specificato per questo attributo quando la definizione della coda viene creata o modificata.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_REMOTE_Q_NAME con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da MQ_Q_NAME_LENGTH.

RetentionInterval (MQLONG)

Questo è il periodo di tempo per cui conservare la coda. Una volta trascorso questo tempo, la coda è idonea per l'eliminazione.

Tabella 618. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

L'ora viene misurata in ore, contando la data e ora in cui è stata creata la coda. La data e l'ora di creazione della coda vengono registrate negli attributi **CreationDate** e **CreationTime** .

Queste informazioni vengono fornite per consentire a un'applicazione di manutenzione o all'operatore di identificare ed eliminare le code che non sono più richieste.

Nota: Il gestore code non esegue mai alcuna azione per eliminare le code in base a questo attributo o per impedire l'eliminazione delle code con un intervallo di conservazione non scaduto; è responsabilità dell'utente intraprendere qualsiasi azione richiesta.

Utilizzare un intervallo di conservazione realistico per impedire l'accumulo di code dinamiche permanenti (vedere attributoDefinitionType). Tuttavia, questo attributo può essere utilizzato anche con code predefinite.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_RETENTION_INTERVAL con la chiamata MQINQ.

Ambito (MQLONG)

Controlla se una voce per questa coda esiste anche in una directory della cella.

Tabella 619. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X		X	X	

Una directory di celle viene fornita da un Servizio nomi installabile. Il valore è uno dei seguenti:

MGR coda MQSCO
La definizione della coda ha un ambito gestore code: la definizione della coda non si estende oltre il gestore code che la possiede. Per aprire la coda per l'output da un altro gestore code, è necessario specificare il nome del gestore code proprietario oppure l'altro gestore code deve avere una definizione locale della coda.

CCELL MQSCO
La definizione della coda ha un ambito cella: la definizione della coda viene inserita anche in una directory cella disponibile per tutti i gestori code nella cella. La coda può essere aperta per l'output da qualsiasi gestore code nella cella specificando il nome della coda; non è necessario specificare il nome del gestore code proprietario della coda. Tuttavia, la definizione della coda non è disponibile per alcun gestore code nella cella che dispone anche di una definizione locale di una coda con tale nome, poiché la definizione locale ha la precedenza.

Una directory di celle viene fornita da un Servizio nomi installabile.

Le code modello e dinamiche non possono avere l'ambito della cella.

Questo valore è valido solo se è stato configurato un servizio nomi che supporta una directory della cella.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_SCOPE con la chiamata MQINQ.

Il supporto per questo attributo è soggetto alle seguenti limitazioni:

- Su IBM i, l'attributo è supportato, ma è valido solo MQSCO_Q_MGR.
- Su z/OS, l'attributo non è supportato.

Condivisibile (MQLONG)
Indica se la coda può essere aperta per l'input più volte contemporaneamente.

Tabella 620. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Il valore è uno dei seguenti:

MQQA_XX_ENCODE_CASE_ONE abilitazione
La coda è condivisibile.

Sono consentite più aperture con l'opzione MQOO_INPUT_SHARED.

MQQA_NOT_SHAREABLE
La coda non è condivisibile.

Una chiamata MQOPENcon l'opzione MQOO_INPUT_SHARED viene considerata come MQOO_INPUT_EXCLUSIVE.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_SHAREABILITY con la chiamata MQINQ.

StorageClass (MQCHAR8)
Si tratta di un nome definito dall'utente che definisce la memoria fisica utilizzata per conservare la coda. In pratica, un messaggio viene scritto su disco solo se è necessario paginarlo fuori dal relativo buffer di memoria.

Tabella 621. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_STORAGE_CLASS con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da MQ_STORAGE_CLASS_LENGTH.

z/OS questo attributo è supportato solo su z/OS.

TriggerControl (MQLONG)

Controlla se i messaggi trigger vengono scritti in una coda di iniziazione per avviare un'applicazione per servire la coda.

Tabella 622. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Il valore è uno dei seguenti:

MQT_DISATTIVO
Nessun messaggio trigger deve essere scritto per questa coda. Il valore di *TriggerType* è irrilevante in questo caso.

MQT_ATTIVO
I messaggi trigger devono essere scritti per questa coda quando si verificano gli eventi trigger appropriati.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_TRIGGER_CONTROL con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET.

TriggerData (MQCHAR64)

Si tratta di dati in formato libero che il gestore code inserisce nel messaggio trigger quando un messaggio in arrivo su questa coda causa la scrittura di un messaggio trigger nella coda di avvio.

Tabella 623. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Il contenuto di questi dati non è significativo per il gestore code. È significativo per l'applicazione di controllo dei trigger che elabora la coda di iniziazione o per l'applicazione che avvia il monitoraggio dei trigger.

La stringa di caratteri non deve contenere valori null. Se necessario, viene riempito a destra con spazi vuoti.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_TRIGGER_DATA con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET. La lunghezza di questo attributo è fornita da MQ_TRIGGER_DATA_LENGTH.

TriggerDepth (MQLONG)

Tabella 624. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Questo è il numero di messaggi con priorità *TriggerMsgPriority* o superiore che devono essere sulla coda prima che venga scritto un messaggio trigger. Ciò si applica quando *TriggerType* è impostato su MQTT_DEPTH. Il valore di *TriggerDepth* è uno o superiore. Questo attributo non viene utilizzato altrimenti.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_TRIGGER_DEPTH con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET.

Priorità TriggerMsg(MQLONG)

Questa è la priorità del messaggio al di sotto della quale i messaggi non contribuiscono alla generazione di messaggi trigger (ossia, il gestore code ignora questi messaggi quando decide se generare un messaggio trigger).

Tabella 625. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

TriggerMsgPriority può essere compreso tra zero (il più basso) e *MaxPriority* (il più alto; consultare l'attributo MaxPriority); un valore zero fa sì che tutti i messaggi contribuiscano alla generazione di messaggi trigger.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_TRIGGER_MSG_PRIORITY con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET.

TriggerType (MQLONG)

Ciò controlla le condizioni in cui i messaggi trigger vengono scritti come risultato dei messaggi in arrivo su questa coda.

Tabella 626. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Ha uno dei seguenti valori:

MQTT_NONE
Non viene scritto alcun messaggio trigger come risultato dei messaggi su questa coda. Ciò ha lo stesso effetto dell'impostazione di *TriggerControl* su MQTC_OFF.

MQTT_FIRST
Un messaggio trigger viene scritto ogni volta che il numero di messaggi con priorità *TriggerMsgPriority* o maggiore sulla coda cambia da 0 a 1.

MQTT_EVERY
Un messaggio trigger viene scritto ogni volta che un messaggio di priorità *TriggerMsgPriority* o superiore arriva sulla coda.

DEPTH MQT
Un messaggio di trigger viene scritto ogni volta che il numero di messaggi con priorità *TriggerMsgPriority* o superiore sulla coda è uguale o superiore a *TriggerDepth*. Una volta scritto il messaggio di trigger, *TriggerControl* viene impostato su MQTC_OFF per impedire ulteriori trigger fino a quando non viene nuovamente attivato esplicitamente.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_TRIGGER_TYPE con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET.

Utilizzo (MQLONG)

Indica per cosa viene utilizzata la coda.

Tabella 627. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Il valore è uno dei seguenti:

MQUS_NORMALE
Si tratta di una coda utilizzata dalle applicazioni durante l'inserimento e il richiamo dei messaggi; la coda non è una coda di trasmissione.

MQUS_TRASMISSIONE
Questa è una coda utilizzata per contenere i messaggi destinati ai gestori code remoti. Quando un'applicazione invia un messaggio a una coda remota, il gestore code locale memorizza temporaneamente il messaggio sulla coda di trasmissione appropriata in un formato speciale. Un agente del canale dei messaggi legge quindi il messaggio dalla coda di trasmissione e lo trasporta al gestore code remoto. Per ulteriori informazioni sulla configurazione della gestione remota, fare riferimento a Configurazione dei gestori code per la gestione remota.

Solo le applicazioni privilegiate possono aprire una coda di trasmissione per MQOO_OUTPUT per inserire messaggi direttamente su di essa. Di solito, solo le applicazioni di utilità lo fanno. Verificare che il formato dei dati del messaggio sia corretto (consultare “MQXQH - Intestazione coda di trasmissione” a pagina 635). o potrebbero verificarsi degli errori durante il processo di trasmissione. Il contesto non viene passato o impostato a meno che non venga specificata una delle opzioni di contesto MQPMO_ * _CONTEXT.

Per stabilire il valore di questo attributo, utilizzare il selettore MQIA_USAGE con la chiamata MQINQ.

XmitQName (MQCHAR48)
Questo è il nome della coda di trasmissione. Se questo attributo non è vuoto quando si verifica un'apertura, per una coda remota o per una definizione dell'alias del gestore code, specifica il nome della coda di trasmissione locale da utilizzare per l'inoltro del messaggio.

Tabella 628. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
			X	

Se **XmitQName** è vuoto, la coda locale con un nome uguale a **RemoteQMgrName** viene utilizzata come coda di trasmissione. Se non è presente alcuna coda con il nome **RemoteQMgrName**, viene utilizzata la coda identificata dall'attributo del gestore code **DefXmitQName** .

Questo attributo viene ignorato se la definizione viene utilizzata come alias del gestore code e **RemoteQMgrName** è il nome del gestore code locale. Viene ignorato anche se la definizione è utilizzata come una definizione di alias di coda risposta.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_XMIT_Q_NAME con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da MQ_Q_NAME_LENGTH.

Attributi per gli elenchi nomi

La seguente tabella riepiloga gli attributi specifici degli elenchi nomi. Gli attributi sono descritti in ordine alfabetico.

Gli elenchi nomi sono supportati su tutti i sistemi IBM MQ , più IBM MQ MQI clients connessi a questi sistemi.

Nota: I nomi degli attributi visualizzati in questa sezione sono nomi descrittivi utilizzati con le chiamate MQINQ e MQSET; i nomi sono gli stessi dei comandi PCF. Quando i comandi MQSC vengono utilizzati per definire, modificare o visualizzare gli attributi, vengono utilizzati nomi brevi alternativi; per ulteriori informazioni, consultare Comandi MQSC .

Tabella 629. Attributi per gli elenchi nomi	
Attributo	**Descrizione**
AlterationDate	Data dell'ultima modifica della definizione
AlterationTime	Ora dell'ultima modifica della definizione
NameCount	Numero di nomi nell'elenco nomi
NamelistDesc	Descrizione elenco nomi
NamelistName	Il nome dell'elenco dei nomi
Nomi	Un elenco di nomi *NameCount*
NamelistType	Tipo di elenco nomi
QSGDisp	Disposizione del gruppo di condivisione code

AlterationDate (MQCHAR12)

Questa è la data dell'ultima modifica della definizione. Il formato della data è YYYY-MM-DD, riempito con due spazi finali per rendere la lunghezza di 12 byte.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_ALTERATION_DATE con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Questa è l'ora dell'ultima modifica della definizione. Il formato dell'ora è HH.MM.SS.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_ALTERATION_TIME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_TIME_LENGTH.

NameCount (MQLONG)

Questo è il numero di nomi presenti nell'elenco nomi. È maggiore o uguale a zero. Viene definito il seguente valore:

MQNC_MAX_NAMELIST_NAME_COUNT
Numero massimo di nomi in un elenco nomi.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_NAME_COUNT con la chiamata MQINQ.

NamelistDesc (MQCHAR64)

Utilizzare questo campo per un commento descrittivo; il suo valore viene stabilito dal processo di definizione. Il contenuto del campo non ha alcun significato per il gestore code, ma il gestore code potrebbe richiedere che il campo contenga solo caratteri che possono essere visualizzati. Non può contenere caratteri null; se necessario, viene riempito a destra con spazi. In un'installazione DBCS, questo campo può contenere caratteri DBCS (con una lunghezza massima di 64 byte).

Nota: Se questo campo contiene caratteri non presenti nella serie di caratteri del gestore code (come definito dall'attributo del gestore code **CodedCharSetId**), tali caratteri potrebbero essere tradotti in modo non corretto se questo campo viene inviato a un altro gestore code.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_NAMELIST_DESC con la chiamata MQINQ.

La lunghezza di questo attributo viene fornita da MQ_NAMELIST_DESC_LENGTH.

NamelistName (MQCHAR48)

Questo è il nome di un elenco nomi definito nel gestore code locale. Per ulteriori informazioni sui nomi degli elenchi nomi, consultare la sezione Altri nomi oggetto .

Ogni elenco nomi ha un nome diverso dai nomi di altri elenchi nomi appartenenti al gestore code, ma potrebbe duplicare i nomi di altri oggetti gestore code di tipi differenti (ad esempio, code).

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_NAMELIST_NAME con la chiamata MQINQ.

La lunghezza di questo attributo è fornita da MQ_NAMELIST_NAME_LENGTH.

NamelistType (MQLONG)

Specifica la natura dei nomi nell'elenco nomi e indica come viene utilizzato l'elenco nomi. È una dei seguenti valori:

MQNT_NONE
Elenco nomi senza tipo assegnato.

MQNT_Q
Elenco nomi contenente nomi di code.

MQNT_CLUSTER
Elenco nomi contenente nomi di cluster.

INFO AUTORE MQNT
Elenco nomi contenente i nomi degli oggetti delle informazioni di autenticazione.

Per definire il valore di questo attributo, utilizzare il selettore MQIA_NAMELIST_TYPE con la chiamata MQINQ.

z/OS questo attributo è supportato solo su z/OS.

Nomi (MQCHAR48xNameCount)

Questo è un elenco di nomi *NameCount* , in cui ciascun nome è il nome di un oggetto definito per il gestore code locale. Per ulteriori informazioni sui nomi oggetto, consultare Regole per la denominazione degli oggetti IBM MQ.

Per determinare il valore di questo attributo, utilizzare il programma di selezione MQCA_NAMES con la chiamata MQINQ.

La lunghezza di ciascun nome nell'elenco è fornita da MQ_OBJECT_NAME_LENGTH.

QSGDisp (MQLONG)

Specifica la disposizione dell'elenco nomi. Il valore è uno dei seguenti:

MQQSGD_Q_MGR
L'oggetto ha disposizione del gestore code: la definizione dell'oggetto è nota solo al gestore code locale; la definizione non è nota ad altri gestori code nel gruppo di condivisione code.

Ogni gestore code nel gruppo di condivisione code può avere un oggetto con lo stesso nome e tipo dell'oggetto corrente, ma si tratta di oggetti separati e non vi è alcuna correlazione tra loro. I loro attributi non sono vincolati ad essere gli stessi.

MQQSGD_XX_ENCODE_CASE_ONE copia
L'oggetto è una copia locale di una definizione di oggetto principale che esiste nel repository condiviso. Ogni gestore code nel gruppo di condivisione code può avere la propria copia dell'oggetto. Inizialmente, tutte le copie hanno gli stessi attributi, ma è possibile modificare ciascuna copia utilizzando i comandi MQSC, in modo che i relativi attributi differiscano da quelli delle altre copie. Gli attributi delle copie vengono risincronizzati quando la definizione principale nel repository condiviso viene modificata.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_QSG_DISP con la chiamata MQINQ.

z/OS questo attributo è supportato solo su z/OS.

Attributi per le definizioni di processi

La seguente tabella riepiloga gli attributi specifici delle definizioni di processo. Gli attributi sono descritti in ordine alfabetico.

Nota: i nomi degli attributi in questa sezione sono nomi descrittivi utilizzati con le chiamate MQINQ e MQSET; i nomi sono gli stessi dei comandi PCF. Quando i comandi MQSC vengono utilizzati per definire, modificare o visualizzare gli attributi, vengono utilizzati nomi brevi alternativi; per ulteriori informazioni, consultare Comandi MQSC .

Tabella 630. Attributi per le definizioni di processi

Attributo	Descrizione
AlterationDate	Data dell'ultima modifica della definizione
AlterationTime	Ora dell'ultima modifica della definizione
ApplId	Identificativo applicazione
ApplType	Tipo di applicazione
EnvData	Dati ambiente
ProcessDesc	Descrizione del processo
ProcessName	Nome processo
QSGDisp	Disposizione del gruppo di condivisione code
UserData	Dati utente

AlterationDate (MQCHAR12)

Questa è la data dell'ultima modifica della definizione. Il formato della data è YYYY-MM-DD, riempito con due spazi finali per rendere la lunghezza di 12 byte.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_ALTERATION_DATE con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Questa è l'ora dell'ultima modifica della definizione. Il formato dell'ora è HH.MM.SS.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_ALTERATION_TIME con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_TIME_LENGTH.

ApplId (MQCHAR256)

Si tratta di una stringa di caratteri che identifica l'applicazione da avviare. Queste informazioni vengono utilizzate da un'applicazione di controllo trigger che elabora i messaggi sulla coda di iniziazione; le informazioni vengono inviate alla coda di iniziazione come parte del messaggio trigger.

Il significato di *ApplId* è determinato dall'applicazione trigger - monitor. Il controllo dei trigger fornito da IBM MQ richiede *ApplId* come nome di un programma eseguibile. Le seguenti note si applicano agli ambienti indicati:

- In z/OS, *ApplId* deve essere:
  - Un identificativo di transazione CICS , per le applicazioni avviate utilizzando la transazione CKTI di controllo trigger CICS
  - Un identificativo della transazione IMS , per le applicazioni avviate utilizzando il controllo trigger IMS CSQQTRMN
- Su Windows, il nome del programma può essere preceduto da un percorso di unità e directory.
- Su AIX and Linux, il nome programma può essere preceduto da un percorso di directory.

La stringa di caratteri non può contenere valori null. Se necessario, viene riempito a destra con spazi vuoti.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_APPL_ID con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_PROCESS_APPL_ID_LENGTH.

ApplType (MQLONG)

Identifica la natura del programma da avviare in risposta alla ricezione di un messaggio trigger. Queste informazioni vengono utilizzate da un'applicazione di controllo trigger che elabora i messaggi sulla coda di iniziazione; le informazioni vengono inviate alla coda di iniziazione come parte del messaggio trigger.

ApplType può avere qualsiasi valore, ma i seguenti valori sono consigliati per i tipi standard; limitare i tipi di applicazione definiti dall'utente ai valori compresi tra MQAT_USER_FIRST e MQAT_USER_LAST:

AIX MQAT
Applicazione AIX (stesso valore di MQAT_UNIX).

MQAT_BATCH
Applicazione batch

MQAT_CICS
Transazione CICS .

IMS MQAT
Applicazione IMS .

MQAT_IM_Bridge
Applicazione bridge IMS .

JAVA MQAT
Applicazione Java .

MVS MQAT
Applicazione MVS o TSO (stesso valore di MQAT_ZOS).

MQAT_OS390
Applicazione OS/390 (stesso valore di MQAT_ZOS).

MQAT_OS400
Applicazione IBM i .

UNIX MQAT
Applicazione UNIX .

MQAT_SCONOSCIUTO
Applicazione di tipo sconosciuto.

UTENTE MQAT
Applicazione utente.

WINDOWS MQAT
Applicazione Windows a 64 bit.

MQAT_WINDOWS_NT
Applicazione Windows a 32 bit.

WLM MQAT
Applicazione z/OS workload manager.

ZOS MQAT
Applicazione z/OS .

MQAT_USER_FIRST
Il valore più basso per il tipo di applicazione definito dall'utente.

MQAT_USER_LAST
Il valore più alto per il tipo di applicazione definito dall'utente.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_APPL_TYPE con la chiamata MQINQ.

EnvData (MQCHAR128)

Si tratta di una stringa di caratteri che contiene informazioni relative all'ambiente relative all'applicazione da avviare. Queste informazioni vengono utilizzate da un'applicazione di controllo trigger che elabora i messaggi sulla coda di iniziazione; le informazioni vengono inviate alla coda di iniziazione come parte del messaggio trigger.

Il significato di *EnvData* è determinato dall'applicazione trigger - monitor. Il controllo trigger fornito da IBM MQ accoda *EnvData* all'elenco di parametri passato all'applicazione avviata. L'elenco dei parametri è composto dalla struttura MQTMC2 , seguita da uno spazio vuoto, seguita da *EnvData* con spazi vuoti finali rimossi. Le seguenti note si applicano agli ambienti indicati:

- Su z/OS:
  - *EnvData* non viene utilizzato dalle applicazioni trigger - monitor fornite da IBM MQ.
  - Se ApplType è MQAT_WLM, è possibile fornire i valori predefiniti in EnvData per i campi ServiceName e ServiceStep nell'intestazione delle informazioni di lavoro (MQWIH).
- Su AIX and Linux, *EnvData* può essere impostato sul carattere & per eseguire l'applicazione avviata in background.

La stringa di caratteri non può contenere valori null. Se necessario, viene riempito a destra con spazi vuoti.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_ENV_DATA con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_PROCESS_ENV_DATA_LENGTH.

ProcessDesc (MQCHAR64)

Utilizzare questo campo per commenti descrittivi. Il contenuto del campo non ha alcun significato per il gestore code, ma il gestore code potrebbe richiedere che il campo contenga solo caratteri che possono essere visualizzati. Non può contenere caratteri null; se necessario, viene riempito a destra con spazi. In un'installazione DBCS, il campo può contenere caratteri DBCS (con una lunghezza massima di 64 byte).

Nota: Se questo campo contiene caratteri non presenti nella serie di caratteri del gestore code (come definito dall'attributo del gestore code **CodedCharSetId**), tali caratteri potrebbero essere tradotti in modo non corretto se questo campo viene inviato a un altro gestore code.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_PROCESS_DESC con la chiamata MQINQ.

La lunghezza di questo attributo è fornita da MQ_PROCESS_DESC_LENGTH.

ProcessName (MQCHAR48)

Questo è il nome di una definizione di processo definita nel gestore code locale.

Ciascuna definizione di processo ha un nome diverso da quello di altre definizioni di processi appartenenti al gestore code. Ma il nome della definizione del processo potrebbe essere uguale ai nomi di altri oggetti del gestore code di tipi differenti (ad esempio, code).

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_PROCESS_NAME con la chiamata MQINQ.

La lunghezza di questo attributo è fornita da MQ_PROCESS_NAME_LENGTH.

QSGDisp (MQLONG)

Specifica la disposizione della definizione del processo. Il valore è uno dei seguenti:

MQQSGD_Q_MGR
L'oggetto ha disposizione del gestore code: la definizione dell'oggetto è nota solo al gestore code locale; la definizione non è nota ad altri gestori code nel gruppo di condivisione code.

Ogni gestore code nel gruppo di condivisione code può avere un oggetto con lo stesso nome e tipo dell'oggetto corrente, ma si tratta di oggetti separati e non vi è alcuna correlazione tra loro. I loro attributi non sono vincolati ad essere gli stessi.

MQQSGD_XX_ENCODE_CASE_ONE copia
L'oggetto è una copia locale di una definizione di oggetto principale che esiste nel repository condiviso. Ogni gestore code nel gruppo di condivisione code può avere la propria copia dell'oggetto. Inizialmente, tutte le copie hanno gli stessi attributi, ma è possibile modificare ciascuna copia utilizzando i comandi MQSC, in modo che i relativi attributi differiscano da quelli delle altre copie. Gli attributi delle copie vengono risincronizzati quando la definizione principale nel repository condiviso viene modificata.

Per determinare il valore di questo attributo, utilizzare il selettore MQIA_QSG_DISP con la chiamata MQINQ.

z/OS questo attributo è supportato solo su z/OS.

UserData (MQCHAR128)

UserData è una stringa di caratteri che contiene informazioni utente relative all'applicazione da avviare. Queste informazioni vengono utilizzate da un'applicazione di controllo dei trigger che elabora i messaggi sulla coda di avvio o dall'applicazione avviata dal controllo dei trigger. Le informazioni vengono inviate alla coda di iniziazione come parte del messaggio trigger.

Il significato di *UserData* è determinato dall'applicazione trigger - monitor. Il controllo trigger fornito da IBM MQ passa *UserData* all'applicazione avviata come parte dell'elenco di parametri. L'elenco dei parametri è costituito dalla struttura MQTMC2 (contenente *UserData*), seguita da uno spazio vuoto, seguito da *EnvData* con spazi vuoti finali rimossi.

La stringa di caratteri non può contenere valori null. Se necessario, viene riempito a destra con spazi vuoti. Per Microsoft Windows, la stringa di caratteri non deve contenere virgolette doppie se la definizione del processo sta per essere inoltrata a **runmqtrm**.

Per determinare il valore di questo attributo, utilizzare il selettore MQCA_USER_DATA con la chiamata MQINQ. La lunghezza di questo attributo è fornita da MQ_PROCESS_USER_DATA_LENGTH.

Codici di ritorno

Per ogni chiamata MQI (IBM MQ Message Queue Interface) e MQAI (IBM MQ Administration Interface), un codice di **completamento** e un codice **motivo** vengono restituiti dal gestore code o da una routine di uscita, per indicare l'esito positivo o negativo della chiamata.

Le applicazioni non devono dipendere da errori controllati in un ordine specifico, tranne dove specificamente indicato. Se più di un codice di completamento o di errore può derivare da una chiamata, l'errore particolare riportato dipende dall'implementazione.

Le applicazioni che verificano il corretto completamento dopo una chiamata API IBM MQ devono sempre verificare il codice di completamento. Non assumere il valore del codice di completamento, in base al valore del codice di errore.

Codici di completamento

Il parametro del codice di completamento (*CompCode*) consente al chiamante di vedere rapidamente se la chiamata è stata completata correttamente, è stata completata parzialmente o non è riuscita.Il seguente è un elenco di codici di completamento, con più dettagli di quelli forniti nelle descrizioni delle chiamate:

MQCC_OK
La chiamata è stata completata completamente; tutti i parametri di output sono stati impostati. Il parametro **Reason** ha sempre il valore MQRC_NONE in questo caso.

MQCC_AVVERTENZA
La chiamata è stata completata parzialmente. Alcuni parametri di output potrebbero essere stati impostati in aggiunta ai parametri di output *CompCode* e *Reason* . Il parametro **Reason** fornisce ulteriori informazioni sul completamento parziale.

MQCC_NON RIUSCITO
L'elaborazione della chiamata non è stata completata. Lo stato del gestore code non viene modificato, tranne dove indicato in modo specifico. I parametri di output *CompCode* e *Reason* sono stati impostati; gli altri parametri non vengono modificati, tranne dove indicato.

Il motivo potrebbe essere un errore nel programma applicativo oppure potrebbe essere il risultato di una situazione esterna al programma, ad esempio l'autorizzazione dell'utente potrebbe essere stata revocata. Il parametro **Reason** fornisce ulteriori informazioni sull'errore.

Codici di origine

Il parametro del codice di errore (*Reason*) qualifica il parametro del codice di completamento (*CompCode*).

Se non vi è alcun motivo speciale per eseguire il report, viene restituito MQRC_NONE. Una chiamata eseguita correttamente restituisce MQCC_OK e MQRC_NONE.

Se il codice di completamento è MQCC_WARNING o MQCC_FAILED, il gestore code riporta sempre un motivo valido; i dettagli vengono forniti sotto ogni descrizione della chiamata.

Quando le routine di uscita utente impostano i codici di completamento e i motivi, devono rispettare queste regole. Inoltre, i valori dei motivi speciali definiti dalle uscite utente devono essere inferiori a zero per garantire che non siano in conflitto con i valori definiti dal gestore code. Le uscite possono impostare i motivi già definiti dal gestore code, dove appropriato.

I codici di errore si verificano anche in:

- Il campo *Reason* della struttura MQDLH.
- Il campo *Feedback* della struttura di MQMD

Per una descrizione completa dei codici di errore, vedere Messaggi e codici di errore.

Regole per la convalida delle opzioni MQI

Questa sezione elenca le situazioni che producono un codice motivo MQRC_OPTIONS_ERROR da una chiamata MQOPEN, MQPUT, MQPUT1, MQGET, MQCLOSE o MQSUB.

chiamata MQOPEN

Per le opzioni della chiamata MQOPEN:

- È possibile specificare almeno *uno* dei seguenti valori:
  - MQOO_SFOGLIA
  - MQOO_INPUT_EXCLUSIVE [1]
  - MQOO_INPUT_SHARED [1]
  - MQOO_INPUT_AS_Q_DEF [1]
  - MQOO_INQUIRE
  - OUTPUT MQOO
  - SET MQOO
  - MQOO_BIND_ON_OPEN [2]
  - MQOO_BIND_NON_FISSO [2]
  - MQOO_BIND_ON_XX_ENCODE_CASE_ONE gruppo [2]
  - MQOO_BIND_AS_Q_DEF [2]
- È consentito solo *uno* dei seguenti:
  - MQOO_READ_AHEAD
  - MQOO_NO_READ_AHEAD

- MQOO_READ_AHEAD_AS_Q_DEF

1. È consentito solo *uno* dei seguenti:
   - MQOO_INPUT_EXCLUSIVE
   - MQOO_INPUT_SHARED
   - MQOO_INPUT_AS_Q_DEF
2. È consentito solo *uno* dei seguenti:
   - MQOO_BIND_ON_OPEN
   - MQOO_BIND_NON_FISSO
   - Gruppo_BIND_MQOO
   - MQOO_BIND_AS_Q_DEF

   Nota: Le opzioni precedentemente elencate si escludono a vicenda. Tuttavia, poiché il valore di MQOOO_BIND_AS_Q_DEF è zero, specificarlo con una delle altre due opzioni di bind non risulta nel codice motivo MQRC_OPTIONS_ERROR. MQOO_BIND_AS_Q_DEF viene fornito per la documentazione del programma.

- Se viene specificato MQOO_SAVE_ALL_CONTEXT, è necessario specificare anche una delle opzioni MQOO_INPUT_ *.
- Se è specificata una delle opzioni MQOO_SET_ * _CONTEXT o MQOO_PASS_ * _CONTEXT, è necessario specificare anche MQOO_OUTPUT.
- Se viene specificato MQOO_CO_OP, è necessario specificare anche MQOO_BROWSE
- Se viene specificato MQOO_NO_MULTICAST, è necessario specificare anche MQOO_OUTPUT.

Chiamata MQPUT

Per le opzioni put - message:

- La combinazione di MQPMO_SYNCPOINT e MQPMO_NO_SYNCPOINT non è consentita.
- È consentito solo *uno* dei seguenti:
  - MQPMO_DEFAULT_CONTEXT
  - MQPMO_NO_CONTEXT
  - MQPMO_PASS_ALL_CONTEXT
  - MQPMO_PASS_IDENTITY_CONTEXT
  - MQPMO_SET_ALL_CONTEXT
  - MQPMO_SET_IDENTITY_CONTEXT
- È consentito solo *uno* dei seguenti:
  - MQPMO_ASYNC_RESPONSE
  - MQPMO_SYNC_RESPONSE
  - MQPMO_RESPONSE_AS_TOPIC_DEF
  - MQPMO_RESPONSE_AS_Q_DEF
- MQPMO_ALTERNATE_USER_AUTHORITY non è consentito (è valido soltanto sulla chiamata MQPUT1).

chiamata MQPUT1

Per le opzioni put - message, le regole sono le stesse della chiamata MQPUT, ad eccezione di quanto segue:

- MQPMO_ALTERNATE_USER_AUTHORITY è consentito.
- MQPMO_LOGICAL_ORDER non è consentito.

Chiamata MQGET

Per le opzioni get - message:

- È consentito solo *uno* dei seguenti:
  - MQGMO_NO_SYNCPOINT
  - SYNCPOINT MQGMO
  - MQGMO_SYNCPOINT_IF_PERSISTENT
- È consentito solo *uno* dei seguenti:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_SUCCESSIVO
  - MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT non è consentito con uno dei seguenti:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_SUCCESSIVO
  - LOCK_MQGMO
  - MQGMO_UNLOCK
- MQGMO_SYNCPOINT_IF_PERSISTENT non è consentito con quanto segue:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_SUCCESSIVO
  - MQGMO_COMPLETE_MSG
  - MQGMO_UNLOCK
- MQGMO_MARK_SKIP_BACKOUT richiede la specifica di MQGMO_SYNCPOINT.
- La combinazione di MQGMO_WAIT e MQGMO_SET_SIGNAL non è consentita.
- Se viene specificato MQGMO_LOCK, è necessario specificare anche uno dei seguenti:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_SUCCESSIVO
- Se viene specificato MQGMO_UNLOCK, sono consentiti solo i seguenti valori:
  - MQGMO_NO_SYNCPOINT
  - MQGMO_NO_WAIT

chiamata MQCLOSE

Per le opzioni della chiamata MQCLOSE:

- La combinazione di MQCO_DELETE e MQCO_DELETE_PURGE non è consentita.
- È consentito solo uno dei seguenti:
  - SUB MQCO_KEEP_
  - MQCO_REMOVE_SUB

Chiamata MQSUB

Per le opzioni della chiamata MQSUB:

- È necessario specificare almeno uno dei seguenti valori:
  - ALTER MQSO
  - RESUME MQSO
  - CREA_MQSO
- È consentito solo uno dei seguenti:
  - MQSO_DURATA
  - MQSO_NON_DURABLE

  Nota: Le opzioni precedentemente elencate si escludono a vicenda. Tuttavia, dal momento che il valore di MQSO_NON_DURABLE è zero, specificarlo con MQSO_DURABLE non risulta nel codice motivo MQRC_OPTIONS_ERROR. MQSO_NON_DURABLE viene fornito per la documentazione del programma.
- La combinazione di MQSO_GROUP_SUB e MQSO_MANAGED non è consentita.
- MQSO_GROUP_SUB richiede MQSO_SET_CORREL_ID da specificare.
- È consentito solo uno dei seguenti:
  - IDER_ANY_MQSO
  - IDUSER_FIX_MQSO
- MQSO_NEW_PUBLICATIONS_ONLY è consentito in combinazione con:
  - CREA_MQSO
  - MQSO_ALTER, se MQSO_NEW_PUBLICATIONS_ONLY è stato impostato sulla sottoscrizione originale
- La combinazione di MQSO_PUBLICATIONS_ON_REQUEST e SubLevel maggiore di 1 non è consentita.
- È consentito solo uno dei seguenti:
  - MQSO_WILDCARD_CHAR
  - MQSO_WILDCARD_TOPIC
- MQSO_NO_MULTICAST richiede che sia specificato MQSO_MANAGED.

Messaggi di comando di pubblicazione / sottoscrizione accodati

Un'applicazione può utilizzare i messaggi di comandi MQRFH2 per controllare un'applicazione di pubblicazione / sottoscrizione in coda.

Un'applicazione che utilizza MQRFH2 per la pubblicazione / sottoscrizione può inviare i seguenti messaggi di comando a SYSTEM.BROKER.CONTROL.QUEUE:

- “Messaggio Elimina pubblicazione” a pagina 903
- “Messaggio Annulla registrazione sottoscrittore (subscriber)” a pagina 904
- “Pubblica messaggio” a pagina 908
- “Messaggio Registra sottoscrittore (subscriber)” a pagina 910
- “Messaggio Richiedi aggiornamento” a pagina 915

Se si stanno scrivendo applicazioni di pubblicazione / sottoscrizione accodate, è necessario comprendere questi messaggi, il messaggio di risposta del gestore code e il descrittore del messaggio (MQMD); consultare le seguenti informazioni:

- “Messaggio di risposta gestore code” a pagina 918
- “Impostazioni MQMD per le pubblicazioni inoltrate da un gestore code” a pagina 924
- “Impostazioni MQMD nei messaggi di risposta del gestore code” a pagina 925
- “Codici di errore di pubblicazione / sottoscrizione” a pagina 919

I comandi sono contenuti in una cartella psc nel campo **NameValueData** dell'intestazione MQRFH2 . Il messaggio che può essere inviato da un broker in risposta a un messaggio di comando è contenuto in una cartella pscr .

Nella descrizione di ciascun comando è presente un elenco delle proprietà che possono essere presenti in una cartella. Se non diversamente specificato, le proprietà sono facoltative e possono verificarsi solo una volta.

I nomi delle proprietà vengono visualizzati come <Command>.

I valori devono essere in formato stringa, ad esempio: Publish.

Una costante stringa che rappresenta il valore di una proprietà viene visualizzata tra parentesi, ad esempio: (MQPSC_PUBLISH).

Le costanti di stringa sono definite nel file di intestazione cmqpsc.h fornito con il gestore code.

Messaggio Elimina pubblicazione

Il messaggio di comando **Delete Publication** viene inviato a un gestore code da un publisher o da un altro gestore code per indicare al gestore code di eliminare tutte le pubblicazioni conservate per gli argomenti specificati.

Questo messaggio viene inviato a una coda monitorata dall'interfaccia di pubblicazione / sottoscrizione accodata del gestore code.

La coda di input deve essere la coda a cui è stata inviata la pubblicazione di origine.

Se si dispone dell'autorizzazione per alcuni argomenti, ma non per tutti, specificati nel messaggio di comando **Delete Publication** , solo tali argomenti vengono eliminati. Un messaggio **Broker Response** indica quali argomenti non vengono eliminati.

Allo stesso modo, se un comando **Publish** contiene più di un argomento, un comando **Delete Publication** che corrisponde ad alcuni di questi argomenti, ma non a tutti, elimina solo le pubblicazioni per gli argomenti specificati nel comando **Delete Publication** .

Consultare “Impostazioni MQMD per le pubblicazioni inoltrate da un gestore code” a pagina 924 per dettagli sui parametri MQMD (message descriptor) necessari quando si invia un messaggio di comando al gestore code.

Proprietà

Comando (*MQPSC_COMMAND*)
Il valore è DeletePub (*MQPSC_DELETE_PUBLICATION*).

Questa proprietà deve essere specificata.

Argomento> (*MQPSC_TOPIC*)
Il valore è una stringa che contiene un argomento per il quale devono essere eliminate delle pubblicazioni conservate. È possibile utilizzare i caratteri wildcard nella stringa per eliminare le pubblicazioni su più di un argomento.

Questa proprietà deve essere specificata; può essere ripetuta per tutti gli eventuali argomenti necessari.

DelOpt (*MQPSC_DELETE_OPTION*)
La proprietà delle opzioni di eliminazione può assumere uno dei seguenti valori:

Local (*MQPSC_LOCAL*)

Tutte le pubblicazioni conservate per gli argomenti specificati vengono eliminate nel gestore code locale (ovvero, il gestore code a cui viene inviato questo messaggio), indipendentemente dal fatto che siano state pubblicate con l'opzione Locale o meno.

Le pubblicazioni in altri gestori code non sono interessate.

None (*MQPSC_NONE*)

Tutte le opzioni assumono i relativi valori predefiniti. Ciò ha lo stesso effetto di omettere la proprietà `DelOpt`. Se sono specificate altre opzioni in contemporanea, `None` viene ignorato.

Il valore predefinito se questa proprietà viene omessa è che tutte le pubblicazioni conservate per gli argomenti specificati vengono eliminate in tutti i gestori code nella rete, indipendentemente dal fatto che siano state pubblicate con l'opzione `Locale` .

Esempio

Di seguito viene riportato un esempio di `NameValueData` per un messaggio di comando **`Delete Publication`** . Viene utilizzato dall'applicazione di esempio per eliminare, nel gestore code locale, la pubblicazione conservata che contiene il punteggio più recente nella corrispondenza tra Team1 e Team2.

```
<psc>
  <Command>DeletePub</Command>
  <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
  <DelOpt>Local</DelOpt>
</psc>
```

Messaggio Annulla registrazione sottoscrittore (subscriber)

Il messaggio di comando **`Deregister Subscriber`** viene inviato a un gestore code da un sottoscrittore (subscriber) o da un'altra applicazione per conto di un sottoscrittore (subscriber), per indicare che non desidera più ricevere messaggi che corrispondono ai parametri forniti.

Questo messaggio viene inviato a SYSTEM.BROKER.CONTROL.QUEUE, la coda di controllo del gestore code. L'utente deve disporre dell'autorizzazione necessaria per inserire un messaggio in tale coda.

Consultare Impostazioni MQMD per le pubblicazioni inoltrate da un gestore code per i dettagli dei parametri MQMD (message descriptor) necessari durante l'invio di un messaggio di comando al gestore code.

È possibile annullare una singola sottoscrizione specificando l'argomento corrispondente, il punto di sottoscrizione ed i valori di filtro della sottoscrizione di origine. Se, nella sottoscrizione di origine non vengono specificati dei valori (ovvero, vengono assunti i valori predefiniti), questi devono essere omessi in fase di annullamento della sottoscrizione.

È possibile annullare tutte le sottoscrizioni relative ad uno o più sottoscrittori (subscriber), utilizzando l'opzione `DeregAll`. Ad esempio, se è specificata l'opzione `DeregAll` insieme ad un punto di sottoscrizione (ma non è specificato nessun argomento o filtro), vengono annullate tutte le sottoscrizioni relative ad un sottoscrittore (subscriber) sul punto di sottoscrizione specificato, indipendentemente dall'argomento e dal filtro. È consentita qualsiasi combinazione di argomento, filtro e punto di sottoscrizione; se tutti questi tre elementi sono specificati, può corrispondere solo una sottoscrizione e l'opzione `DeregAll` viene ignorata.

Il messaggio deve essere inviato dal sottoscrittore (subscriber) che ha registrato la sottoscrizione; ciò è confermato verificando l'ID utente del sottoscrittore (subscriber).

La registrazione delle sottoscrizioni può essere annullata anche da un amministratore di sistema utilizzando i comandi MQSC o PCF. Tuttavia, le sottoscrizioni registrate con una coda dinamica temporanea sono associate alla coda, non al nome della coda. Se la coda viene eliminata, esplicitamente o dall'applicazione che si disconnette dal gestore code, non è più possibile utilizzare il comando **`Deregister Subscriber`** per annullare la registrazione delle sottoscrizioni per tale coda. Le sottoscrizioni possono essere annullate utilizzando il workbench dello sviluppatore e vengono rimosse automaticamente dal gestore code alla successiva corrispondenza di una pubblicazione con la sottoscrizione o al successivo riavvio del gestore code. In circostanze normali, le applicazioni devono annullare la registrazione delle relative sottoscrizioni prima di eliminare la coda o nello scollegamento dal gestore code.

Se un sottoscrittore (subscriber) invia un messaggio per annullare la registrazione di una sottoscrizione e riceve un messaggio di risposta per indicare che è stata elaborata correttamente, alcune pubblicazioni

potrebbero comunque raggiungere la coda del sottoscrittore (subscriber) se sono state elaborate dal gestore code contemporaneamente all'annullamento della registrazione della sottoscrizione. Se i messaggi non vengono eliminati dalla coda, potrebbe riscontrarsi un accumulo di messaggi non elaborati sulla coda del sottoscrittore (subscriber). Se l'applicazione esegue un loop che include una chiamata MQGET con l'appropriato `CorrelId` dopo una certa pausa, questi messaggi vengono eliminati dalla coda.

Allo stesso modo, se il sottoscrittore utilizza una coda dinamica permanente e annulla la registrazione e chiude la coda con l'opzione *MQCO_DELETE_PURGE* su una chiamata MQCLOSE, la coda potrebbe non essere vuota. Se le pubblicazioni dal gestore code non sono ancora sottoposte a commit quando la coda viene eliminata, viene emesso un codice di ritorno MQRC_Q_NOT_EMPTY dalla chiamata MQCLOSE. L'applicazione può evitare questo problema ponendosi in pausa e immettendo nuovamente la chiamata MQCLOSE di tanto in tanto.

Proprietà

Comando (*MQPSC_COMMAND*)
Il valore è `DeregSub` (*MQPSC_DEREGISTER_SUBSCRIBER*).

Questa proprietà deve essere specificata.

Argomento (*MQPSC_TOPIC*)
Il valore è una stringa che contiene l'argomento di cui deve essere annullata la registrazione.

Nell'eventualità debba essere annullata la registrazione di più argomenti, è possibile ripetere questa proprietà. Può essere omesso se `DeregAll` è specificato in `<RegOpt>`.

Gli argomenti specificati possono essere una serie secondaria di quelli registrati, se il sottoscrittore (subscriber) desidera conservare le sottoscrizioni per altri argomenti. I caratteri jolly sono consentiti, ma una stringa di argomenti che contiene caratteri jolly deve corrispondere esattamente alla stringa corrispondente specificata nel messaggio di comando **`Deregister Subscriber`** .

SubPoint (*MQPSC_SUBSCRIPTION_POINT*)
Il valore è una stringa che specifica il punto di sottoscrizione da cui deve essere staccata la sottoscrizione.

Non è possibile ripetere questa proprietà. Può essere omesso se viene specificato `< Topic>` o se `DeregAll` è specificato in `<RegOpt>`. Se si omette questa proprietà, si verifica quanto segue:

- Se **non** si specifica `DeregAll`, la registrazione delle sottoscrizioni che corrispondono alla proprietà `< Topic>` (e alla proprietà `< Filter>` , se presente) viene annullata dal punto di sottoscrizione predefinito.
- Se si specifica `DeregAll`, viene annullata la registrazione di tutte le sottoscrizioni (corrispondenti alle proprietà `< Topic>` e `< Filter>` , se presenti) da tutti i punti di sottoscrizione.

Tenere presente che non è possibile specificare esplicitamente il punto di sottoscrizione predefinito. Non è possibile, quindi, annullare le registrazioni di tutte le sottoscrizioni solo da questo punto di sottoscrizione; è necessario specificare gli argomenti.

SubIdentity (*MQPSC_SUBSCRIPTION_IDENTITY*)
Questa è una stringa a lunghezza variabile, la cui lunghezza massima è di 64 caratteri. Viene utilizzata per rappresentare un'applicazione con un interesse in una sottoscrizione. Il gestore code conserva una serie di identità del sottoscrittore per ogni sottoscrizione. Ogni sottoscrizione può fare in modo che la relativa serie di identità ne contenga solo una oppure un numero illimitato.

Se `SubIdentity` si trova nella serie di identità relativa alla sottoscrizione, viene eliminata dalla serie. Se la serie di identità diventa vuota, la sottoscrizione viene rimossa dal gestore code, a meno che `LeaveOnly` non venga specificato come valore della proprietà `RegOpt` . Se la serie di identità contiene ancora altre identità, la sottoscrizione non viene rimossa dal gestore code e il flusso di pubblicazione non viene interrotto.

Se si specifica `SubIdentity` , ma `SubIdentity` non si trova nella serie di identità per la sottoscrizione, il comando **`Deregister Subscriber`** ha esito negativo con il codice di ritorno *MQRCCF_SUB_IDENTITY_ERROR*.

Filtro (*MQPSC_FILTER*)
Il valore è una stringa che specifica il filtro di cui deve essere annullata la registrazione. Deve corrispondere completamente, compreso maiuscolo/minuscolo e gli eventuali spazi, ad un filtro di sottoscrizione che è stato registrato in precedenza.

Nell'eventualità debba essere annullata la registrazione di più filtri, è possibile ripetere questa proprietà. Può essere omesso se viene specificato < Topic> o se DeregAll è specificato in <RegOpt>.

I filtri specificati possono essere una serie secondaria di quelli registrati se il sottoscrittore (subscriber) desidera conservare le sottoscrizioni per altri filtri.

RegOpt (*MQPSC_REGISTRATION_OPTION*)
La proprietà delle opzioni di registrazione può assumere i seguenti valori:

DeregAll
(*MQPSC_DEREGISTER_ALL*)

Devono essere annullate tutte le sottoscrizioni corrispondenti registrate per questo sottoscrittore (subscriber).

Se si specifica DeregAll:

- < Topic>, <SubPoint>e < Filter> possono essere omessi.
- < Argomento> e < Filtro> possono essere ripetuti, se necessario.
- <SubPoint> non deve essere ripetuto.

Se **non** si specifica DeregAll:

- È necessario specificare < Topic> e, se necessario, è possibile ripeterlo.
- <SubPoint> e < Filter> possono essere omessi.
- <SubPoint> non deve essere ripetuto.
- < Filtro> può essere ripetuto, se necessario.

Se gli argomenti e i filtri sono entrambi ripetuti, tutte le sottoscrizioni corrispondenti a tutte le combinazioni dei due vengono rimosse. Ad esempio, un comando **Deregister Subscriber** che specifica tre argomenti e tre filtri tenterà di eliminare nove sottoscrizioni.

CorrelAsId
(*MQPSC_CORREL_ID_AS_IDENTITY*)

Il CorrelId presente nel descrittore di messaggi (MQMD), che non deve essere zero, viene utilizzato per individuare il sottoscrittore (subscriber). Deve corrispondere a quanto utilizzato nella sottoscrizione di origine per CorrelId.

FullResp
(*MQPSC_FULL_RESPONSE*)

Quando si specifica FullResp , tutti gli attributi della sottoscrizione vengono restituiti nel messaggio di risposta, se il comando non ha esito negativo.

Quando si specifica FullResp DeregAll non è consentito nel comando **Deregister Subscriber** . Non è possibile inoltre specificare più argomenti. In entrambi i casi, il comando ha esito negativo con il codice di ritorno *MQRCCF_REG_OPTIONS_ERROR*.

LeaveOnly
(*MQPSC_LEAVE_ONLY*)

Quando lo specifichi con una SubIdentity che è nella serie di identità per la sottoscrizione, SubIdentity viene rimosso dalla serie di identità per la sottoscrizione. La sottoscrizione non viene rimossa dal gestore code, anche se la serie di identità risultante è vuota. Se il valore SubIdentity non è nell'insieme di identità, il comando ha esito negativo con codice di ritorno *MQRCCF_SUB_IDENTITY_ERROR*.

Se LeaveOnly viene specificato senza alcuna SubIdentity, il comando ha esito negativo con codice di ritorno *MQRCCF_REG_OPTIONS_ERROR*.

Se non vengono specificati né LeaveOnly né SubIdentity , la sottoscrizione viene rimossa indipendentemente dal contenuto della serie di identità per la sottoscrizione.

Nessuno
(MQPSC_NONE)

Tutte le opzioni assumono i relativi valori predefiniti. Ciò ha lo stesso effetto di omettere la proprietà delle opzioni di registrazione. Se sono specificate altre opzioni in contemporanea, None viene ignorato.

VariableUserId
(*MQPSC_VARIABLE_USER_ID*)

Quando specificata, l'identità del sottoscrittore (subscriber), coda, gestore code e correlid, non è limitata ad un solo ID utente. Ciò differisce dal comportamento esistente del gestore code che associa l'ID utente del messaggio di registrazione originale all'identità del sottoscrittore e da quel momento in poi impedisce a qualsiasi altro utente di utilizzare tale identità. Se un nuovo sottoscrittore (subscriber) tenta di utilizzare la stessa identità, viene restituito il codice di ritorno *MQRCCF_DUPLICATE_SUBSCRIPTION*.

Qualsiasi utente può modificare o annullare la registrazione della sottoscrizione se in possesso dell'autorizzazione appropriata, evitando la verifica esistente della corrispondenza tra l'ID utente e quello del sottoscrittore (subscriber) di origine.

Per aggiungere questa opzione ad una sottoscrizione esistente il comando deve provenire dallo stesso ID utente della sottoscrizione di origine.

Se la sottoscrizione di cui annullare la registrazione ha VariableUserId impostato, è necessario che sia impostato al momento dell'annullamento della registrazione per indicare quale sottoscrizione è in fase di annullamento della registrazione. In alternativa, l'ID utente del comando **Deregister Subscriber** viene utilizzato per identificare la sottoscrizione. Questo viene sovrascritto, insieme agli altri identificativi del sottoscrittore (subscriber), se viene fornito un nome di sottoscrizione.

Se questa proprietà è omessa, il valore predefinito è che non viene impostata alcuna opzione di registrazione.

QMgrName (*MQPSC_Q_MGR_NAME*)
Il valore è il nome del gestore code per la coda del sottoscrittore (subscriber). Deve corrispondere a quanto utilizzato nella sottoscrizione di origine per QMgrName.

Se questa proprietà è omessa, il valore predefinito è il nome ReplyToQMgr nel descrittore di messaggi (MQMD). Se il nome risultante è vuoto, per impostazione predefinita viene utilizzato il nome del gestore code.

QName (*MQPSC_Q_NAME*)
Il valore è il nome della coda del sottoscrittore (subscriber). Deve corrispondere a quanto utilizzato nella sottoscrizione di origine per QName.

Se questa proprietà è omessa, il valore predefinito è il nome ReplyToQ nel descrittore di messaggi (MQMD), che non deve essere vuoto.

SubName (*MQPSC_SUBSCRIPTION_NAME*)
Se si specifica SubName su un comando **Deregister Subscriber** il valore SubName ha la precedenza su tutti gli altri campi identificativo tranne l'ID utente, a meno che VariableUserId non sia impostato sulla sottoscrizione stessa. Se VariableUserId non è impostato, il comando **Deregister Subscriber** ha esito positivo solo se l'id utente del messaggio di comando corrisponde a quello della sottoscrizione, in caso contrario il comando ha esito negativo con codice di ritorno *MQRCCF_DUPLICATE_IDENTITY*.

Se esiste una sottoscrizione che corrisponde all'identità tradizionale di questo comando ma non ha alcun SubName il comando **Deregister Subscriber** ha esito negativo con codice di ritorno

MQRCCF_SUB_NAME_ERROR. Se viene effettuato un tentativo di annullamento della registrazione di una sottoscrizione che ha un SubName utilizzando un messaggio di comando che corrisponde all'identità tradizionale ma senza SubName specificato, il comando ha esito positivo.

Dati SubUser(*MQPSC_SUBSCRIPTION_USER_DATA*)
Questa è una stringa di testo a lunghezza variabile. Il valore viene memorizzato dal gestore code con la sottoscrizione ma non influisce sulla consegna della pubblicazione al sottoscrittore. Il valore può essere modificato con un nuovo valore eseguendo la nuova registrazione sulla stessa sottoscrizione. Questo attributo è per l'utilizzo dell'applicazione.

SubUserI dati vengono restituiti nelle informazioni metatopiche (MQCACF_REG_SUB_USER_DATA) per una sottoscrizione, se sono presenti i dati SubUser.

Esempio

Di seguito viene riportato un esempio di `NameValueData` per un messaggio di comando **Deregister Subscriber** . In questo esempio, l'applicazione annulla la registrazione della relativa sottoscrizione agli argomenti che contengono il punteggio aggiornato di tutti gli incontri. L'identità del sottoscrittore (subscriber), comprendente `CorrelId`, deriva dai valori predefiniti presenti in MQMD.

```
<psc>
  <Command>DeregSub</Command>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Pubblica messaggio

Il messaggio di comando **Publish** viene inserito in una coda o da un gestore code a un sottoscrittore (subscriber) per pubblicare informazioni su uno o più argomenti specificati.

È necessaria l'autorizzazione a inserire un messaggio in una coda e l'autorizzazione a pubblicare informazioni su uno o più argomenti specificati.

Se l'utente dispone dell'autorizzazione per pubblicare le informazioni su alcuni argomenti, ma non su tutti, solo tali argomenti vengono utilizzati per la pubblicazione; una risposta di avvertenza indica quali argomenti non vengono utilizzati per la pubblicazione.

Se un sottoscrittore (subscriber) ha sottoscrizioni corrispondenti, il gestore code inoltra il messaggio **Publish** alle code del sottoscrittore (subscriber) definite nei corrispondenti messaggi di comando **Register Subscriber** .

Consultare Messaggio di risposta del gestore code per i dettagli dei parametri del descrittore del messaggio (MQMD) necessari per l'invio di un messaggio di comando al gestore code e utilizzati quando un gestore code inoltra una pubblicazione a un sottoscrittore.

Il gestore code inoltra il messaggio **Publish** ad altri gestori code nella rete che hanno sottoscrizioni corrispondenti, a meno che non si tratti di una pubblicazione locale.

I dati di pubblicazione, se presenti, sono inclusi nel contenuto del messaggio. I dati possono essere descritti in una cartella <mcd> nel campo `NameValueData` dell'intestazione MQRFH2 .

Proprietà

Comando (*MQPSC_COMMAND*)
Il valore è `Pubblica` (*MQPSC_PUBLISH*).

Questa proprietà deve essere specificata.

Argomento (*MQPSC_TOPIC*)
Il valore è una stringa che contiene un argomento che classifica questa pubblicazione. Non è consentito l'uso di caratteri wildcard.

È necessario aggiungere l'argomento all'elenco nomi SYSTEM.QPUBSUB.QUEUE.NAMELIST, consultare Aggiunta di un flusso per istruzioni su come completare questa attività.

Questa proprietà deve essere specificata e può essere ripetuta per tutti gli eventuali argomenti necessari.

SubPoint (*MQPSC_SUBSCRIPTION_POINT*)
Il punto di sottoscrizione su cui viene pubblicata la pubblicazione.

In WebSphere Event Broker 6.0, il valore della proprietà <SubPoint> è il valore dell'attributo Subscription Point del nodo Publication che gestisce la pubblicazione.

In IBM WebSphere MQ 7.0.1, il valore della proprietà <SubPoint> deve corrispondere al nome di un punto di sottoscrizione. Consultare l'argomento Aggiunta di un punto di sottoscrizione.

PubOpt (*MQPSC_PUBLICATION_OPTION*)
La proprietà delle opzioni di pubblicazione può assumere i seguenti valori:

RetainPub
(*MQPSC_RETAIN_PUB*)

Il gestore code conserva una copia della pubblicazione. Se questa opzione non è impostata, la pubblicazione viene eliminata non appena il gestore code ha inviato la pubblicazione a tutti i sottoscrittori correnti.

IsRetainedPub
(*MQPSC_IS_RETAINED_PUB*)

(Può essere impostato solo da un gestore code). Questa pubblicazione è stata conservata dal gestore code. Il gestore code imposta questa opzione per notificare a un sottoscrittore che questa pubblicazione è stata pubblicata in precedenza ed è stata conservata, a condizione che la sottoscrizione sia stata registrata con l'opzione `InformIfRetained` . Viene impostata solo in risposta ad un messaggio di comando `Registra sottoscrittore (subscriber)` o `Richiedi aggiornamento`. Questa opzione non viene impostata per le pubblicazioni conservate che vengono inviate direttamente ai sottoscrittori (subscriber).

Local
(*MQPSC_LOCAL*)

Questa opzione indica al gestore code che questa pubblicazione non deve essere inviata ad altri gestori code. Tutti i sottoscrittori registrati in questo gestore code ricevono questa pubblicazione se hanno sottoscrizioni corrispondenti.

OtherSubsOnly
(*MQPSC_OTHER_SUBS_ONLY*)

Questa opzione consente un'elaborazione semplificata di applicazioni di tipo conferenza, in cui un autore (publisher) è anche un sottoscrittore (subscriber) dello stesso argomento. Indica al gestore code di non inviare la pubblicazione alla coda del sottoscrittore (subscriber) dell'autore (publisher) anche se dispone di una sottoscrizione corrispondente. La coda del sottoscrittore (subscriber) del publisher è composta da `QMgrName`, `QNamee` `CorrelId`facoltativi, come descritto nel seguente elenco.

CorrelAsId
(*MQPSC_CORREL_ID_AS_IDENTITY*)

`CorrelId` in MQMD (che non deve essere zero) fa parte della coda del sottoscrittore (subscriber) dell'autore (publisher), in applicazioni in cui l'autore (publisher) è anche un sottoscrittore (subscriber).

Nessuno
(*MQPSC_NONE*)

Tutte le opzioni assumono i relativi valori predefiniti. Ciò ha lo stesso effetto di omettere la proprietà delle opzioni di pubblicazione. Se sono specificate altre opzioni in contemporanea, `None` viene ignorato.

È possibile disporre di più di un'opzione di pubblicazione introducendo ulteriori elementi <PubOpt> .

Se questa proprietà è omessa, il valore predefinito è che non viene impostata alcuna opzione di pubblicazione.

PubTime (*MQPSC_PUBLISH_TIMESTAMP*)
Il valore è l'impostazione facoltativa da parte dell'autore (publisher) di data/ora della pubblicazione. È possibile utilizzare un massimo di 16 caratteri nel formato:

```
YYYYMMDDHHMMSSTH
```

in base all'UT (Universal Time). Queste informazioni non vengono controllate dal gestore code prima di essere inviate ai sottoscrittori.

SeqNum (*MQPSC_SEQUENCE_NUMBER*)
Il valore è l'impostazione facoltativa da parte dell'autore (publisher) del numero di sequenza.

Deve essere incrementato di 1 a ogni pubblicazione. Tuttavia, ciò non viene verificato dal gestore code, che trasmette semplicemente queste informazioni ai sottoscrittori.

Se le pubblicazioni sullo stesso argomento vengono pubblicate su diversi gestori code interconnessi, è responsabilità dei publisher garantire che i numeri di sequenza, se utilizzati, siano significativi.

QMgrName (*MQPSC_Q_MGR_NAME*)
Il valore è una stringa contenente il nome del gestore code per la coda del sottoscrittore (subscriber) del publisher, nelle applicazioni in cui il publisher è anche un sottoscrittore (vedere `OtherSubs`).

Se questa proprietà è omessa, il valore predefinito è il nome `ReplyToQMgr` nel descrittore di messaggi (MQMD). Se il nome risultante è vuoto, per impostazione predefinita viene utilizzato il nome del gestore code.

QName (*MQPSC_Q_NAME*)
Il valore è una stringa contenente il nome della coda del sottoscrittore (subscriber) del publisher, nelle applicazioni in cui il publisher è anche un sottoscrittore (consultare `OtherSubsOnly`).

Se questa proprietà è omessa, il valore predefinito è il nome `ReplyToQ` nel descrittore di messaggi (MQMD), che non deve essere vuoto se presente l'impostazione per `OtherSubsOnly`.

Esempio

Di seguito sono riportati alcuni esempi di *NameValueData* per un messaggio di comando **Publish** .

Il primo esempio è relativo ad una pubblicazione inviata dal simulatore dell'incontro nell'applicazione di esempio per indicare l'inizio di un incontro.

```
<psc>
  <Command>Publish</Command>
  <Topic>Sport/Soccer/Event/MatchStarted</Topic>
</psc>
```

Il secondo esempio è relativo ad una pubblicazione conservata. Viene pubblicato il punteggio aggiornato nell'incontro tra Team1 e Team2.

```
<psc>
  <Command>Publish</Command>
  <PubOpt>RetainPub</PubOpt>
  <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
</psc>
```

Messaggio Registra sottoscrittore (subscriber)

Il messaggio di comando **`Register Subscriber`** viene inviato a un gestore code da un sottoscrittore (subscriber) o da un'altra applicazione per conto di un sottoscrittore (subscriber), per indicare che

desidera sottoscrivere uno o più argomenti in un punto di sottoscrizione. È possibile specificare anche un filtro del contenuto del messaggio.

Nelle espressioni di filtro di pubblicazione / sottoscrizione, la nidificazione delle parentesi causa una diminuzione esponenziale delle prestazioni. Evitare di nidificare le parentesi ad una profondità maggiore di circa 6.

Il messaggio viene inviato a SYSTEM.BROKER.CONTROL.QUEUE, che è la coda di controllo del gestore code. È richiesta l'autorizzazione per inserire un messaggio in questa coda, oltre all'autorizzazione di accesso (impostata dall'amministratore di sistema del gestore code) per l'argomento, o gli argomenti, nella sottoscrizione.

Se l'utente dispone dell'autorizzazione solo su alcuni argomenti, non su tutti, vengono registrati solo quegli argomenti; una risposta di avvertenza indica gli argomenti non registrati.

Consultare "Impostazioni MQMD nei messaggi di comando per il gestore code" a pagina 923 per dettagli sui parametri MQMD (message descriptor) necessari quando si invia un messaggio di comando al gestore code.

Se la risposta alla coda è una coda dinamica temporanea, la registrazione della sottoscrizione viene annullata automaticamente dal gestore code quando la coda viene chiusa.

Proprietà

Comando (*MQPSC_COMMAND*)
Il valore è RegSub (*MQPSC_REGISTER_SUBSCRIBER*). Questa proprietà deve essere specificata.

Argomento (*MQPSC_TOPIC*)
L'argomento per il quale il sottoscrittore (subscriber) desidera ricevere le pubblicazioni. È possibile specificare i caratteri wildcard come parte dell'argomento.

Se si utilizza il comando MQSC **display sub** per esaminare la sottoscrizione creata in questo modo, il valore della tag < Topic> viene visualizzato come proprietà TOPICSTR della sottoscrizione.

Questa proprietà è richiesta e può essere ripetuta per tutti gli eventuali argomenti necessari.

SubPoint (*MQPSC_SUBSCRIPTION_POINT*)
Il valore è il punto di sottoscrizione a cui è collegata la sottoscrizione.

Se questa proprietà è omessa, viene utilizzato il punto di sottoscrizione predefinito.

In WebSphere Event Broker 6.0, il valore della proprietà <SubPoint> deve essere uguale al valore dell'attributo Subscription Point dei nodi Publication sottoscritti.

In IBM WebSphere MQ 7.0.1, il valore della proprietà <SubPoint> deve corrispondere al nome di un punto di sottoscrizione. Consultare l'argomento Aggiunta di un punto di sottoscrizione.

Filtro (*MQPSC_FILTER*)
Il valore è un'espressione SQL utilizzata come filtro del contenuto dei messaggi di pubblicazione. Se una pubblicazione sull'argomento specificato ha una corrispondenza con il filtro, viene inviata al sottoscrittore (subscriber). Questa proprietà corrisponde alla stringa di selezione utilizzata nelle chiamate MQSUB e MQOPEN. Per ulteriori informazioni, consultare Selezione del contenuto di un messaggio

Se questa proprietà è omessa, non viene eseguito alcun filtro del contento.

RegOpt (*MQPSC_REGISTRATION_OPTION*)
La proprietà delle opzioni di registrazione può assumere i seguenti valori:

AddName
(*MQPSC_ADD_NAME*)

Quando specificato per una sottoscrizione esistente che corrisponde all'identità tradizionale di questo comando Registra sottoscrizione, ma senza il valore SubName corrente, il SubName specificato in questo comando viene aggiunto alla sottoscrizione.

Se si specifica AddName il campo SubName è obbligatorio, altrimenti viene restituito MQRCCF_REG_OPTIONS_ERROR.

CorrelAsId
(*MQPSC_CORREL_ID_AS_IDENTITY*)

Il CorrelId presente nel descrittore di messaggi (MQMD), viene utilizzato quando si inviano pubblicazioni corrispondenti alla coda del sottoscrittore (subscriber). CorrelId non deve corrispondere a zero.

FullResp
(*MQPSC_FULL_RESPONSE*)

Quando specificato, tutti gli attributi della sottoscrizione vengono restituiti nel messaggio di risposta, se il comando non ha esito negativo.

FullResp è valido solo quando il messaggio di comando fa riferimento ad una sola sottoscrizione. Quindi, nel comando è consentito un solo un argomento; altrimenti il comando ha esito negativo e viene restituito il codice di ritorno *MQRCCF_REG_OPTIONS_ERROR*.

InformIfRet
(*MQPSC_INFORM_IF_RETAINED*)

Il gestore code informa il sottoscrittore se una pubblicazione viene conservata quando invia un messaggio di pubblicazione in risposta a un messaggio di comando **Register Subscriber** o **Request Update** . Il gestore code esegue questa operazione includendo l'opzione di pubblicazione IsRetainedPub nel messaggio.

JoinExcl
(*MQPSC_JOIN_EXCLUSIVE*)

Questa opzione indica che SubIdentity deve essere aggiunto come membro esclusivo della serie di identità per la sottoscrizione e che non è possibile aggiungere altre identità alla serie.

Se l'identità è già stata unita ed è l'unica voce della serie, tale serie viene cambiata come esclusiva di questa identità. Altrimenti, se la sottoscrizione ha attualmente altre identità nella serie (con accesso condiviso) il comando non ha esito positivo e viene restituito il codice di ritorno *MQRCCF_SUBSCRIPTION_IN_USE*.

JoinShared
(*MQPSC_JOIN_SHARED*)

Questa opzione indica che la SubIdentity specificata deve essere aggiunta alla serie di identità per la sottoscrizione.

Se la sottoscrizione è attualmente bloccata esclusivamente (utilizzando l'opzione JoinExcl), il comando ha esito negativo con il codice di ritorno *MQRCCF_SUBSCRIPTION_LOCKED*, a meno che l'identità che ha la sottoscrizione bloccata non sia la stessa di quella presente in questo messaggio di comando. In tale caso il blocco viene modificato automaticamente in un blocco condiviso.

Local
(*MQPSC_LOCAL*)

La sottoscrizione è locale e non è distribuita ad altri gestori code nella rete. Le pubblicazioni effettuate su altri gestori code non vengono consegnate a questo sottoscrittore, a meno che non disponga anche di una sottoscrizione globale corrispondente.

NewPubsOnly
(*MQPSC_NEW_PUBS_ONLY*)

Le pubblicazioni conservate esistenti al momento della registrazione della sottoscrizione non vengono inviate al sottoscrittore (subscriber); vengono inviate solo quelle nuove.

Se un sottoscrittore (subscriber) esegue una nuova registrazione e modifica questa opzione in modo che non è più impostata, potrebbe verificarsi che una pubblicazione che gli è già è stata inviata venga inoltrata nuovamente.

NoAlter
(*MQPSC_NO_ALTER*)

Non vengono modificati gli attributi di una sottoscrizione corrispondente esistente.

Quando viene creata una sottoscrizione, questa opzione viene ignorata. Sono attive per la nuova sottoscrizione tutte le altre opzioni specificate.

Se una `SubIdentity` ha anche una delle opzioni join (`JoinExcl` o `JoinShared`) l'identità viene aggiunta alla serie di identità indipendentemente dal fatto che `NoAlter` sia specificato.

Nessuno
(*MQPSC_NONE*)

Tutte le opzioni di registrazione assumono i relativi valori predefiniti.

Se il sottoscrittore è già registrato, le relative opzioni vengono reimpostate sui valori predefiniti (notare che ciò non ha lo stesso effetto dell'omissione della proprietà delle opzioni di registrazione) e la scadenza della sottoscrizione viene aggiornata dall'MQMD del messaggio **`Register Subscriber`** .

Se sono specificate altre opzioni di registrazione in contemporanea, None viene ignorato.

NonPers
(*MQPSC_NON_PERSISTENT*)

Le pubblicazioni che corrispondono a questa sottoscrizione vengono distribuite al sottoscrittore (subscriber) come messaggi non permanenti.

Pers
(*MQPSC_PERSISTENT*)

Le pubblicazioni che corrispondono a questa sottoscrizione vengono distribuite al sottoscrittore (subscriber) come messaggi permanenti.

PersAsPub
(*MQPSC_PERSISTENT_AS_PUBLISH*)

Le pubblicazioni che corrispondono a questa sottoscrizione vengono distribuite al sottoscrittore (subscriber) con la persistenza specificata dall'autore (publisher). Questo è il funzionamento predefinito.

PersAsQueue
(*MQPSC_PERSISTENT_AS_Q*)

Le pubblicazioni che corrispondono a questa sottoscrizione vengono distribuite al sottoscrittore (subscriber) con la persistenza specificata nella relativa coda.

PubOnReqOnly
(*MQPSC_PUB_ON_REQUEST_ONLY*)

Il gestore code non invia pubblicazioni al sottoscrittore (subscriber), tranne in risposta a un messaggio di comando **`Request Update`** .

VariableUserId
(*MQPSC_VARIABLE_USER_ID*)

Quando specificata, l'identità del sottoscrittore (subscriber), coda, gestore code e correlid, non è limitata ad un solo ID utente. Ciò differisce dal comportamento esistente del gestore code che associa l'ID utente del messaggio di registrazione originale all'identità del sottoscrittore e da quel momento in poi impedisce a qualsiasi altro utente di utilizzare tale identità.
Se un nuovo sottoscrittore (subscriber) tenta di utilizzare la stessa identità, viene restituito *MQRCCF_DUPLICATE_SUBSCRIPTION*.

Ciò consente ad ogni utente di modificare o annullare la registrazione della sottoscrizione se l'utente dispone dell'autorizzazione appropriata. Non c'è quindi nessuna necessità di verificare che l'ID utente corrisponda a quello del sottoscrittore (subscriber) di origine.

Per aggiungere questa opzione ad una sottoscrizione esistente il comando deve provenire dallo stesso ID utente della sottoscrizione di origine.

Se la sottoscrizione del comando **Request Update** ha VariableUserId impostato, deve essere impostato al momento dell'aggiornamento della richiesta per indicare a quale sottoscrizione si fa riferimento. In alternativa, l'ID utente del comando **Request Update** viene utilizzato per identificare la sottoscrizione. Questo viene sovrascritto, insieme agli altri identificativi del sottoscrittore (subscriber), se viene fornito un nome di sottoscrizione.

Se un messaggio di comando **Register Subscriber** senza questa serie di opzioni fa riferimento a una sottoscrizione esistente che ha questa serie di opzioni, l'opzione viene rimossa da questa sottoscrizione e l'id utente della sottoscrizione viene ora corretto. Se esiste già un sottoscrittore (subscriber) con la stessa identità (coda, gestore code e identificativo di correlazione) ma con un ID utente differente associato ad esso, il comando ha esito negativo con codice di ritorno *MQRCCF_DUPLICATE_IDENTITY* perché può esistere solo un ID utente associato a un'identità del sottoscrittore (subscriber).

Se la proprietà delle opzioni di registrazione viene omessa e il sottoscrittore (subscriber) è già registrato, le relative opzioni di registrazione non vengono modificate e la scadenza della sottoscrizione viene aggiornata dall'MQMD del messaggio **Register Subscriber** .

Se il sottoscrittore (subscriber) non è ancora registrato, viene creata una nuova sottoscrizione in cui tutte le opzioni di registrazione assumono i relativi valori predefiniti.

I valori predefiniti sono PersAsPub e nessuna altra opzione impostata.

QMgrName (*MQPSC_Q_MGR_NAME*)
Il valore è il nome del gestore code per la coda del sottoscrittore, a cui il gestore code invia le pubblicazioni corrispondenti.

Se questa proprietà è omessa, il valore predefinito è il nome ReplyToQMgr nel descrittore di messaggi (MQMD). Se il nome risultante è vuoto, viene utilizzato il valore predefinito QMgrNamedel gestore code.

QName (*MQPSC_Q_NAME*)
Il valore è il nome della coda del sottoscrittore (subscriber), a cui il gestore code invia le pubblicazioni corrispondenti.

Se questa proprietà è omessa, il valore predefinito è il nome ReplyToQ nel descrittore di messaggi (MQMD), che in questo caso non deve essere vuoto.

Se la coda è una coda dinamica temporanea, la distribuzione non persistente delle pubblicazioni (NonPers) deve essere specificato nella proprietà <RegOpt> .

Se la coda è una coda dinamica temporanea, la registrazione della sottoscrizione viene annullata automaticamente dal gestore code quando la coda viene chiusa.

SubName (*MQPSC_SUBSCRIPTION_NAME*)
Questo è il nome assegnato ad una determinata sottoscrizione. È possibile utilizzarlo al posto di gestore code, coda e correlId facoltativo, per fare riferimento ad una sottoscrizione.

Se esiste già una sottoscrizione con questo **SubName** , tutti gli altri attributi della sottoscrizione (Topic, QMgrName, QName, CorrelId, UserId, RegOpts, UserSubData e Scadenza) vengono sovrascritti con gli attributi, se specificati, che vengono passati nel nuovo messaggio di comando Registra sottoscrittore (subscriber) . Tuttavia, se **SubName** viene utilizzato senza la specifica del campo QName e nell'intestazione MQMD è specificato ReplyToQ, la coda del sottoscrittore (subscriber) viene modificata in ReplyToQ.

Se una sottoscrizione che corrisponde all'identità tradizionale di questo comando esiste già, ma non ha **SubName** , il comando di registrazione ha esito negativo con codice di ritorno *MQRCCF_DUPLICATE_SUBSCRIPTION*, a meno che non venga specificata l'opzione **AddName** .

Se si tenta di modificare una sottoscrizione denominata esistente utilizzando un altro comando Registra sottoscrittore che specifica lo stesso **SubName** , e i valori di Topic, QMgrName, QName e CorrelId nel nuovo comando corrispondono a una sottoscrizione esistente differente,

con o senza un SubName definito, il comando ha esito negativo con codice di ritorno *MQRCCF_DUPLICATE_SUBSCRIPTION*. Ciò impedisce a due nomi di sottoscrizione di fare riferimento alla stessa sottoscrizione.

SubIdentity (*MQPSC_SUBSCRIPTION_IDENTITY*)
Questa stringa viene utilizzata per rappresentare un'applicazione con un interesse in una sottoscrizione. È una stringa di caratteri a lunghezza variabile con una lunghezza massima di 64 caratteri ed è facoltativa. Il gestore code conserva una serie di identità del sottoscrittore per ogni sottoscrizione. Ogni sottoscrizione (subscriber) può fare in modo che la relativa serie di identità ne contenga solo una oppure un numero illimitato (fare riferimento alle opzioni **JoinShared** e **JoinExcl**).

Un comando di sottoscrizione che specifica l'opzione **JoinShared** o **JoinExcl** aggiunge **SubIdentity** alla serie di identità della sottoscrizione, se non è già presente e se la serie di identità esistente consente tale azione; ovvero, nessun altro sottoscrittore (subscriber) è stato unito in modo esclusivo o la serie di identità è vuota.

Qualsiasi modifica degli attributi della sottoscrizione come risultato di un comando `Registra sottoscrittore (subscriber)` in cui è presente una specifica **SubIdentity**, ha esito positivo solo se è l'unico membro della serie di identità di questa sottoscrizione. Altrimenti, il comando ha esito negativo e viene restituito il codice di ritorno *MQRCCF_SUBSCRIPTION_IN_USE*. Ciò impedisce la modifica degli attributi della sottoscrizione senza che ne siano a conoscenza altri sottoscrittori (subscriber) interessati.

Se si specifica una stringa di caratteri più lunga di 64 caratteri, il comando ha esito negativo e viene restituito il codice di ritorno*MQRCCF_SUB_IDENTITY_ERROR*.

Dati SubUser(*MQPSC_SUBSCRIPTION_USER_DATA*)
Questa è una stringa di testo a lunghezza variabile. Il valore viene memorizzato dal gestore code con la sottoscrizione, ma non ha alcuna influenza sulla consegna della pubblicazione al sottoscrittore. Il valore può essere modificato con un nuovo valore eseguendo la nuova registrazione sulla stessa sottoscrizione. Questo attributo è presente per essere utilizzato dall'applicazione.

SubUserData viene restituito nelle informazioni meta-argomento (*MQCACF_REG_SUB_USER_DATA*) relative ad una sottoscrizione, se presente.

Se si specifica più di uno dei valori dell'opzione di registrazione `NonPers, PersAsPub, PersAsQueue, and Pers`, viene utilizzato solo l'ultimo. Non è possibile combinare queste opzioni in una sola sottoscrizione.

Esempio

Di seguito viene riportato un esempio di `NameValueData` per un messaggio di comando **Register Subscriber** . Nell'applicazione di esempio, il servizio dei risultati utilizza questo messaggio per registrare una sottoscrizione agli argomenti contenenti i punteggi più recenti in tutte le corrispondenze, con l'opzione 'Persistente come pubblicazione' impostata. L'identità del sottoscrittore (subscriber), comprendente `CorrelId`, deriva dai valori predefiniti presenti in MQMD.

```
<psc>
  <Command>RegSub</Command>
  <RegOpt>PersAsPub</RegOpt>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Messaggio Richiedi aggiornamento

Il messaggio di comando **Request Update** viene inviato da un sottoscrittore (subscriber) a un gestore code per richiedere le pubblicazioni correnti conservate per l'argomento e il punto di sottoscrizione specificati che corrispondono al filtro (facoltativo) fornito.

Questo messaggio viene inviato a *SYSTEM.BROKER.CONTROL.QUEUE*, la coda di controllo del gestore code. È richiesta l'autorizzazione per inserire un messaggio in questa coda, oltre all'autorizzazione di accesso per l'argomento nell'aggiornamento della richiesta; questa è impostata dall'amministratore di sistema del gestore code.

Questo comando è normalmente utilizzato se il sottoscrittore (subscriber) ha specificato l'opzione `Pub0nReq0nly` in fase di registrazione. Se il gestore code dispone di pubblicazioni conservate corrispondenti, vengono inviate al sottoscrittore. Se il gestore code non dispone di pubblicazioni conservate corrispondenti, la richiesta ha esito negativo con codice di ritorno *MQRCCF_NO_RETAINED_MSG*. Il richiedente deve avere registrato in precedenza una sottoscrizione con identici valori Topic, SubPoint, e Filter.

Proprietà

Comando (*MQPSC_COMMAND*)
Il valore è `ReqUpdate` (*MQPSC_REQUEST_UPDATE*). Questa proprietà deve essere specificata.

Argomento (*MQPSC_TOPIC*)
Il valore è l'argomento che il sottoscrittore (subscriber) richiede; è consentito l'uso dei caratteri wildcard.

Questa proprietà deve essere specificata, ma in questo messaggio è consentita una sola ricorrenza.

SubPoint (*MQPSC_SUBSCRIPTION_POINT*)
Il valore è il punto di sottoscrizione a cui è collegata la sottoscrizione.

Se questa proprietà è omessa, viene utilizzato il punto di sottoscrizione predefinito.

Filtro (*MQPSC_FILTER*)
Il valore è un'espressione ESQL utilizzata come filtro del contenuto dei messaggi di pubblicazione. Se una pubblicazione sull'argomento specificato ha una corrispondenza con il filtro, viene inviata al sottoscrittore (subscriber).

La proprietà `< Filtro>` deve avere lo stesso valore specificato nella sottoscrizione originale per cui si sta richiedendo un aggiornamento.

Se questa proprietà è omessa, non viene eseguito alcun filtro del contento.

RegOpt (*MQPSC_REGISTRATION_OPTION*)
La proprietà delle opzioni di registrazione può assumere il seguente valore:

`CorrelAsId`
(*MQPSC_CORREL_ID_AS_IDENTITY*)

Il `CorrelId` presente nel descrittore di messaggi (MQMD), che non deve essere zero, viene utilizzato quando si inviano pubblicazioni corrispondenti alla coda del sottoscrittore (subscriber).

`Nessuno`
(*MQPSC_NONE*)

Tutte le opzioni assumono i relativi valori predefiniti. Ciò ha lo stesso effetto di omettere la proprietà `<RegOpt>` . Se sono specificate altre opzioni in contemporanea, `None` viene ignorato.

`VariableUserId`
(*MQPSC_VARIABLE_USER_ID*)

Quando specificata, l'identità del sottoscrittore (subscriber), coda, gestore code e correlid, non è limitata ad un solo ID utente. Ciò differisce dal comportamento esistente del gestore code che associa l'ID utente del messaggio di registrazione originale all'identità del sottoscrittore e da quel momento in poi impedisce a qualsiasi altro utente di utilizzare tale identità. Se un nuovo sottoscrittore (subscriber) tenta di utilizzare la stessa identità, il comando ha esito negativo e viene restituito il codice di ritorno *MQRCCF_DUPLICATE_SUBSCRIPTION*.

Ciò consente ad ogni utente di modificare o annullare la registrazione della sottoscrizione quando dispone dell'autorizzazione appropriata. Non c'è quindi nessuna necessità di verificare che l'ID utente corrisponda a quello del sottoscrittore (subscriber) di origine.

Per aggiungere questa opzione ad una sottoscrizione esistente il comando deve provenire dallo stesso ID utente della sottoscrizione di origine.

Se la sottoscrizione del comando **Request Update** ha VariableUserId impostato, deve essere impostato al momento dell'aggiornamento della richiesta per indicare a quale sottoscrizione si fa riferimento. In alternativa, l'ID utente del comando **Request Update** viene utilizzato per identificare la sottoscrizione. Questo viene sovrascritto, insieme agli altri identificativi del sottoscrittore (subscriber), se viene fornito un nome di sottoscrizione.

Se questa proprietà è omessa, il valore predefinito è che non viene impostata alcuna opzione di registrazione.

QMgrName (*MQPSC_Q_MGR_NAME*)
Il valore è il nome del gestore code per la coda del sottoscrittore, a cui il gestore code invia la pubblicazione conservata corrispondente.

Se questa proprietà è omessa, il valore predefinito è il nome ReplyToQMgr nel descrittore di messaggi (MQMD). Se il nome risultante è vuoto, viene utilizzato il valore predefinito QMgrNamedel gestore code.

QName (*MQPSC_Q_NAME*)
Il valore è il nome della coda del sottoscrittore, a cui il gestore code invia la pubblicazione conservata corrispondente.

Se questa proprietà è omessa, il valore predefinito è il nome ReplyToQ nel descrittore di messaggi (MQMD), che in questo caso non deve essere vuoto.

SubName (*MQPSC_SUBSCRIPTION_NAME*)
Questo è il nome assegnato ad una determinata sottoscrizione. Se specificato su un comando **Request Update** , il valore SubName ha la precedenza su tutti gli altri campi identificativo tranne l'ID utente, a meno che VariableUserId non sia impostato sulla sottoscrizione stessa. Se VariableUserId non è impostato, il comando *Richiedi aggiornamento* ha esito positivo solo se l'ID utente del messaggio di comando corrisponde a quello della sottoscrizione. Se l'ID utente del messaggio di comando non corrisponde a quello della sottoscrizione, il comando ha esito negativo e viene restituito il codice di ritorno *MQRCCF_DUPLICATE_IDENTITY*.

Se VariableUserId è impostato e l'ID utente è diverso da quello della sottoscrizione, il comando ha esito positivo se l'id utente del nuovo messaggio di comando dispone dell'autorità per sfogliare la coda di flusso e inserirlo nella coda del sottoscrittore della sottoscrizione. Altrimenti, il comando ha esito negativo e viene restituito il codice di ritorno *MQRCCF_NOT_AUTHORIZED*.

Se esiste una sottoscrizione che corrisponde all'identit ... tradizionale di questo comando, ma non dispone di SubName, il comando **Request Update** non riesce con codice di ritorno *MQRCCF_SUB_NAME_ERROR*.

Se viene effettuato un tentativo di richiesta di un aggiornamento per una sottoscrizione che ha un SubName utilizzando un messaggio di comando che corrisponde all'identità tradizionale, ma senza SubName specificato, il comando ha esito positivo.

Esempio

Di seguito viene riportato un esempio di NameValueData per un messaggio di comando **Request Update** . Nell'applicazione di esempio, il servizio di risultati utilizza questo messaggio per richiedere le pubblicazioni conservate contenenti i punteggi aggiornati di tutte le squadre. L'identità del sottoscrittore (subscriber), comprendente CorrelId, deriva dai valori predefiniti presenti in MQMD.

```
<psc>
  <Command>ReqUpdate</Command>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Messaggio di risposta gestore code

Un messaggio **Queue Manager Response** viene inviato da un gestore code alla ReplyToQ di un autore (publisher) o di un sottoscrittore (subscriber) per indicare l'esito positivo o negativo di un messaggio di comando ricevuto dal gestore code se il descrittore del messaggio di comando ha specificato che è richiesta una risposta.

Il messaggio di risposta è contenuto nel campo NameValueData dell'intestazione MQRFH2 , in una cartella <pscr> .

In caso di avvertenza o errore, il messaggio di risposta contiene la cartella <psc> del messaggio di comando e la cartella <pscr> . I dati del messaggio, se presenti, non sono contenuti nel messaggio di risposta del gestore code. In caso di errore, non viene elaborato nessuno dei messaggi che ha provocato l'errore; in caso di avvertenza, alcuni dei messaggi potrebbero essere elaborati con esito positivo.

In presenza di un malfunzionamento con invio di una risposta:

- Per i messaggi di pubblicazione, il gestore code tenta di inviare la risposta alla coda di messaggi non recapitabili IBM MQ se MQPUT ha esito negativo. Ciò consente l'invio della pubblicazione ai sottoscrittori (subscriber) anche se la risposta non può essere rimandata all'autore (publisher).
- Relativamente agli altri messaggi o se la risposta di pubblicazione non può essere inviata alla coda di lettere non recapitate, viene registrato un errore e viene eseguito normalmente il roll back del messaggio di comando. Il verificarsi di ciò dipende da come è stato configurato il nodo MQInput.

Proprietà

Completamento (*MQPSCR_COMPLETION*)
Il codice di completamento, può assumere uno dei tre seguenti valori:

ok
Il comando è stato completato correttamente

warning
Il comando è stato completato, ma con un'avvertenza

error
Comando non riuscito

Risposta (*MQPSCR_RESPONSE*)
La risposta ad un messaggio di comando, se tale comando ha prodotto il codice di completamento warning o error. Contiene una proprietà < Motivo> e potrebbe contenere altre proprietà che indicano la causa dell'avvertenza o dell'errore.

In presenza di uno o più errori, è presente una sola cartella di risposta, che indica solamente la causa del primo errore. In caso di una o più avvertenze, è presente una cartella di risposta per ciascuna avvertenza.

Motivo (*MQPSCR_REASON*)
Il codice di errore che qualifica il codice di completamento, se questo è warning o error. È impostato su uno dei codici di errore elencati nel seguente esempio. La proprietà < Motivo> è contenuta in una cartella < Risposta> . Il codice motivo può essere seguito da qualsiasi proprietà valida dalla cartella <psc> (ad esempio, un nome argomento), che indica la causa dell'errore o dell'avvertenza. Se si ottiene un codice di errore di? ???, Verificare la correttezza dei dati, ad esempio, le parentesi angolari corrispondenti (<>).

Esempi

Di seguito sono riportati alcuni esempi di NameValueData in un messaggio **Queue Manager Response** . Una risposta positiva potrebbe essere la seguente:

```
<pscr>
  <Completion>ok</Completion>
</pscr>
```

Di seguito è riportato un esempio di risposta di errore che consiste in un errore del filtro. La prima stringa NameValueData contiene la risposta; la seconda contiene il comando originale.

```
<pscr>
  <Completion>error</Completion>
  <Response>
    <Reason>3150</Reason>
  </Reponse>
</pscr>

<psc>
 ...
 command message (to which
 the queue manager is responding)
 ...
</psc>
```

Di seguito è riportato un esempio di risposta di avvertenza (dovuta ad argomenti non autorizzati). La prima stringa NameValueData contiene la risposta; la seconda stringa NameValueData contiene il comando originale.

```
<pscr>
  <Completion>warning</Completion>
  <Response>
    <Reason>3081</Reason>
    <Topic>topic1</Topic>
  </Reponse>
  <Response>
    <Reason>3081</Reason>
    <Topic>topic2</Topic>
  </Reponse>
</pscr>

<psc>
 ...
 command message (to which
 the queue manager is responding)
 ...
</psc>
```

Codici di errore di pubblicazione / sottoscrizione

Questi codici di errore potrebbero essere restituiti nel campo Motivo di una cartella di risposta di pubblicazione / sottoscrizione <pscr> . Vengono elencate anche le costanti che possono essere utilizzate per rappresentare questi codici nei linguaggi di programmazione C o C + +.

Le costanti MQRC_ richiedono il file di intestazione IBM MQ cmqc.h . Le costanti MQRCCF_ richiedono il file di intestazione IBM MQ cmqcfc.h (a parte *MQRCCF_FILTER_ERROR* e *MQRCCF_WRONG_USER*, che richiedono il file di intestazione cmqpsc.h).

Codice di errore e testo	Spiegazione	Emesso da
2336 MQRC_RFH_COMMAND_ERROR	I valori validi per il campo <Command> di una cartella <psc> sono: RegSub, DeregSub, Publish, DeletePube ReqUpdate. Qualsiasi altro valore risulta in questo codice di errore.	Qualsiasi comando
2337 MQRC_RFH_ PARM_ERROR	Le cartelle <psc> e <mcd> hanno entrambe una serie di parametri validi che possono essere specificati al loro interno. Verificare le descrizioni di queste cartelle ed accertarsi che non siano stati specificati parametri non corretti.	Qualsiasi comando

Codice di errore e testo	Spiegazione	Emesso da
2338 MQRC_RFH_DUPLICATE_PARM	Alcuni parametri (ad esempio, Topic) possono essere ripetuti all'interno di una cartella<psc>, ma altri (ad esempio, Command) non possono esserlo. Verificare di non avere duplicato un parametro non ripetibile.	Qualsiasi comando
2339 MQRC_RFH_PARM_MISSING	Alcuni parametri all'interno delle cartelle <psc> o <mcd> sono facoltativi e possono essere omessi; alcuni sono obbligatori e non devono essere omessi. Verificare di aver incluso tutti i parametri obbligatori nelle proprie cartelle <psc> e <mcd> .	Qualsiasi comando
2551 SELEZIONE_MQRC_NON_DISPONIBILE	Non era disponibile alcun provider di selezione messaggi esteso per stabilire quali sottoscrittori con un filtro specificato devono ricevere la pubblicazione.	Pubblica, registra sottoscrittore e richiedi aggiornamento
	Nessun provider di selezione messaggi esteso era disponibile per gestire il filtro del sottoscrittore specificato.	Registra sottoscrittore e richiedi aggiornamento
2554 ERRORE CONTENUTO MQRC	Un provider di selezione del messaggio esteso ha rilevato un errore nella pubblicazione corrente o conservata.	Pubblica e richiedi aggiornamento
3008 MQRCCF_COMMAND_FAILED	Si è verificato un errore interno che ha impedito l'esecuzione corretta del comando. L'errore potrebbe verificarsi se il comando viene immesso nuovamente. Il log eventi di sistema per il gestore code contiene informazioni da utilizzare quando si riporta il problema a IBM.	Qualsiasi comando
3072 MQRCCF_TOPIC_ERROR	Uno o più valori forniti per il parametro Topic non sono corretti. Verificare che i valori specificati per il parametro Topic siano conformi alle limitazioni indicate.	Qualsiasi comando
3073 MQRCCF_NOT_REGISTERED	La combinazione relativa a SubPoint, Topic e Filter specificata nel comando DeregSub o ReqUpdate non è una combinazione con cui ci si è registrati in precedenza oppure, per il comando DeregSub se è stata specificata l'opzione DeregAll, una delle proprietà SubPoint, Topic o Filter non è stata utilizzata per annullare la registrazione di una sottoscrizione.	Comandi Annulla registrazione sottoscrittore (subscriber) e Richiedi aggiornamento

Codice di errore e testo	Spiegazione	Emesso da
3074 MQRCCF_Q_MGR_NAME_ERROR	Il gestore code specificato non era valido, non era disponibile oppure non era esistente.	Comandi Annulla registrazione sottoscrittore (subscriber), Pubblica, Registra sottoscrittore (subscriber) e Richiedi aggiornamento
3076 MQRCCF_Q_NAME_ERROR	In nome coda specificato non era valido oppure la coda non esiste sul gestore code indicato.	Comandi Annulla registrazione sottoscrittore (subscriber), Pubblica, Registra sottoscrittore (subscriber) e Richiedi aggiornamento
3077 MQRCCF_NO_RETAINED_MSG	Non erano presenti messaggi conservati per l'argomento specificato. Questo potrebbe costituire un errore, a seconda della struttura del programma applicativo.	Comando Richiedi aggiornamento
3079 MQRCCF_INCORRECT_Q	I comandi RegSub, DeregSube ReqUpdate vengono sempre inviati a SYSTEM.BROKER.CONTROL.QUEUE del gestore code a cui sono destinati. I comandi Pubblica ed Elimina pubblicazione vengono inviati alla coda di input del determinato flusso di messaggi di pubblicazione/ sottoscrizione per il quale sono previsti; ciò è determinato in fase di progettazione del flusso di messaggi. Questo codice di errore viene restituito se un comando viene inviato alla coda sbagliata.	Qualsiasi comando
3080 MQRCCF_CORREL_ID_ERROR	CorrelAsId è stato specificato come uno dei parametri RegOpt in uso. Tuttavia, il campo CorrelId di MQMD non contiene un identificativo di correlazione valido (ovvero, è impostato su MQCI_NONE).	Comandi Annulla registrazione sottoscrittore (subscriber) e Registra sottoscrittore (subscriber)
3081 MQRCCF_NOT_AUTHORIZED	L'utente non è autorizzato a eseguire l'azione richiesta. Le impostazioni di autorizzazione per il gestore code vengono gestite dall'amministratore di sistema utilizzando l'editor Gerarchia argomenti.	Comandi Pubblica e Registra sottoscrittore (subscriber)

Codice di errore e testo	Spiegazione	Emesso da
3083 MQRCCF_REG_OPTIONS_ERROR	È stato specificato un parametro RegOpt non riconosciuto nella cartella <psc> che contiene il comando RegSub o DeregSub .	Comandi Annulla registrazione sottoscrittore (subscriber) e Registra sottoscrittore (subscriber)
3084 MQRCCF_PUB_OPTIONS_ERROR	È stato specificato un parametro PubOpt non riconosciuto nella cartella <psc> che contiene il comando Pubblica.	Comando Pubblica
3087 MQRCCF_DEL_OPTIONS_ERROR	È stato specificato un parametro DelOpt non riconosciuto nella cartella <psc> contenente il comando DeletePub .	Comando Elimina pubblicazione
3150 MQRCCF_FILTER_ERROR	Il valore specificato per il parametro Filter non è valido. Verificare nella sezione che descrive la sintassi valida le espressioni di filtro e assicurarsi che l'espressione in uso sia conforme.	Comandi Annulla registrazione sottoscrittore (subscriber), Registra sottoscrittore (subscriber) e Richiedi aggiornamento
3151 MQRCCF_WRONG_USER	Già esiste una sottoscrizione che corrisponde a questa specificata; tuttavia, è stata registrata da un altro utente. Solo l'utente che ne ha eseguito la registrazione in origine può modificare o annullare la registrazione di una sottoscrizione.	Comandi Annulla registrazione sottoscrittore (subscriber), Registra sottoscrittore (subscriber) e Richiedi aggiornamento
3152 MQRCCF_DUPLICATE_SUBSCRIPTION	Esiste già una sottoscrizione corrispondente con un diverso nome di sottoscrizione.	
3153 MQRCCF_SUB_NAME_ERROR	Il formato del nome di sottoscrizione non è valido oppure esiste già una sottoscrizione corrispondente senza nome.	
3154 MQRCCF_SUB_IDENTITY_ERROR	Errore del parametro di identità della sottoscrizione. Il valore fornito supera la lunghezza massima consentita o l'identità della sottoscrizione non è attualmente un membro della serie di identità della sottoscrizione e non è stata specificata un'opzione di unione della registrazione.	
3155 MQRCCF_SUBSCRIPTION_IN_USE	Un membro della serie di identità ha tentato di modificare o annullare la registrazione di una sottoscrizione, quando questi non era l'unico membro di tale serie.	

Codice di errore e testo	Spiegazione	Emesso da
3156 MQRCCF_SUBSCRIPTION_LOCKED	La sottoscrizione è attualmente bloccata in modo esclusivo da un'altra identità.	
3157 MQRCCF_ALREADY_JOINED	È stata specificata un'opzione di unione della registrazione, ma l'identità del sottoscrittore (subscriber) era già un membro della serie di identità di sottoscrizione.	

Impostazioni MQMD nei messaggi di comando per il gestore code

Le applicazioni che inviano messaggi di comando al gestore code utilizzano le seguenti impostazioni dei campi nel descrittore del messaggio (MQMD). I campi lasciati come valore predefinito o per i quali è possibile impostare qualsiasi valore valido nel modo solito, non sono riportati di seguito.

Report
Vedere MsgType e CorrelId.

MsgType
MsgType deve essere impostato su *MQMT_REQUEST* o *MQMT_DATAGRAM*. *MQRC_MSG_TYPE_ERROR* verrà restituito se MsgType non è impostato su uno di questi valori.

MsgType deve essere impostato su *MQMT_REQUEST* per un messaggio di comando, se è sempre richiesta una risposta. Gli indicatori MQRO_PAN e MQRO_NAN nel campo Report non sono in questo caso significativi.

Se MsgType è impostato su *MQMT_DATAGRAM*, le risposte dipendono dall'impostazione degli indicatori MQRO_PAN e MQRO_NAN nel campo Report :

- MQRO_PAN da solo significa che il gestore code invia una risposta solo se il comando ha esito positivo.
- MQRO_NAN da solo significa che il gestore code invia una risposta solo se il comando non riesce.
- Se un comando termina con un'avvertenza, viene inviata una risposta se è impostato MQRO_PAN o MQRO_NAN.
- MQRO_PAN + MQRO_NAN significa che il gestore code invia una risposta se il comando ha esito positivo o negativo. Ciò ha lo stesso effetto, dal punto di vista del gestore code, dell'impostazione di MsgType su MQMT_REQUEST.
- Se non è impostato né MQRO_PAN né MQRO_NAN, non viene inviata alcuna risposta.

Format
Impostazione su MQFMT_RF_HEADER_2

MsgId
Questo campo è normalmente impostato su MQMI_NONE, cosicché il gestore code genera un valore univoco.

CorrelId
Questo campo può essere impostato su qualsiasi valore. Se l'identità del mittente include un CorrelId, specificare questo valore, insieme a MQRO_PASS_CORREL_ID nel campo Report , per assicurarsi che sia impostato in tutti i messaggi di risposta inviati dal gestore code al mittente.

ReplyToQ
Questo campo definisce la coda a cui devono essere inviate le risposte, se presenti. Questa potrebbe essere la coda del mittente; ciò ha il vantaggio che il parametro QName può essere omesso dal messaggio. Se, tuttavia, le risposte devono essere inviate ad una diversa coda, il parametro QName è necessario.

ReplyToQMgr
Questo campo definisce il gestore code per le risposte. Se questo campo viene lasciato vuoto (valore predefinito), il gestore code locale vi inserisce il proprio nome.

Impostazioni MQMD per le pubblicazioni inoltrate da un gestore code

Un gestore code utilizza queste impostazioni dei campi nel descrittore di messaggi (MQMD) quando invia una pubblicazione a un sottoscrittore. Tutti gli altri campi in MQMD sono impostati sui relativi valori predefiniti.

Report
Report è impostato su MQRO_NONE.

MsgType
MsgType è impostato su MQMT_DATAGRAM.

Expiry
Scadenza è impostato sul valore nel messaggio Pubblica ricevuto dal publisher. Nel caso di un messaggio conservato, il tempo in sospeso viene ridotto del tempo approssimativo in cui il messaggio è stato sul gestore code.

Format
Format è impostato su MQFMT_RF_HEADER_2

MsgId
MsgId è impostato su un valore univoco.

CorrelId
Se CorrelId fa parte dell'identità del sottoscrittore (subscriber), questo è il valore specificato dal sottoscrittore (subscriber) in fase di registrazione. Altrimenti, è un valore diverso da zero scelto dal gestore code.

Priority
Priority assume il valore impostato dall'autore (publisher) o come risolto se l'autore (publisher) ha specificato MQPRI_PRIORITY_AS_Q_DEF.

Persistence
Persistence assume il valore impostato dall'autore (publisher) o come risolto se l'autore (publisher) ha specificato MQPER_PERSISTENCE_AS_Q_DEF, a meno che non è specificato diversamente nel messaggio Registra sottoscrittore (subscriber) per il sottoscrittore (subscriber) a cui viene inviata questa pubblicazione.

ReplyToQ
ReplyToQ è impostato su spazi vuoti.

ReplyToQMgr
ReplyToGestore code è impostato sul nome del gestore code.

UserIdentifier
UserIdentifier è l'identificativo utente del sottoscrittore (subscriber), come impostato quando questo è stato registrato.

AccountingToken
AccountingToken è il token di account del sottoscrittore (subscriber) come impostato quando questo è stato registrato per la prima volta.

ApplIdentityData
ApplIdentityData sono i dati di identità dell'applicazione del sottoscrittore (subscriber) come impostato quando questo è stato registrato per la prima volta.

PutApplType
PutApplType è impostato su MQAT_BROKER.

PutApplName
PutApplNome è impostato sui primi 28 caratteri del nome del gestore code.

PutDate
PutDate è la data in cui è stato inserito il messaggio.

PutTime
PutTime è l'ora in cui è stato inserito il messaggio.

ApplOriginData
ApplOriginData è impostato su spazi vuoti.

Impostazioni MQMD nei messaggi di risposta del gestore code

Un gestore code utilizza queste impostazioni dei campi nel descrittore del messaggio (MQMD) quando invia una risposta a un messaggio di pubblicazione. Tutti gli altri campi in MQMD sono impostati sui relativi valori predefiniti.

Report
Report è impostato su tutti zero.

MsgType
MsgType è impostato su MQMT_REPLY.

Format
Format è impostato su MQFMT_RF_HEADER_2

MsgId
L'impostazione di MsgId dipende dalle opzioni Report nel messaggio di comando di origine. Per impostazione predefinita, è impostato su MQMI_NONE, cosicché il gestore code genera un valore univoco.

CorrelId
L'impostazione di CorrelId dipende dalle opzioni Report nel messaggio di comando di origine. Come impostazione predefinita, ciò significa che CorrelId è impostato sullo stesso valore di MsgId del messaggio di comando. Può essere utilizzato per correlare i comandi con le relative risposte.

Priority
Priorità è impostato sullo stesso valore del messaggio di comando originale.

Persistence
Persistence è impostato sullo stesso valore del messaggio di comando di origine.

Expiry
Scadenza è impostato sullo stesso valore del messaggio di comando originale ricevuto dal gestore code.

PutApplType
PutApplType è impostato su MQAT_BROKER.

PutApplName
PutApplName è impostato sui primi 28 caratteri del nome del gestore code.

Altri campi di contesto sono impostati come se fossero generati con MQPMO_PASS_IDENTITY_CONTEXT.

Codifiche macchina

Questa sezione descrive la struttura del campo *Encoding* nel descrittore del messaggio.

Consultare “MQMD - Descrittore messaggi” a pagina 431 per un riepilogo dei campi nella struttura.

Il campo *Encoding* è un numero intero a 32 bit diviso in quattro sottocampi separati; questi sottocampi identificano:

- La codifica utilizzata per i numeri interi binari
- La codifica utilizzata per i numeri interi decimali compressi
- La codifica utilizzata per i numeri a virgola mobile
- Bit riservati

Ogni sottocampo è identificato da una maschera di bit che ha 1 - bit nelle posizioni corrispondenti al sottocampo e 0 - bit altrove. I bit sono numerati in modo che il bit 0 sia il bit più significativo e il bit 31 il bit meno significativo. Sono definite le seguenti maschere:

MASCHERA_NUMERO_INTERO_MQENC_
Maschera per la codifica binario - intero.

Questo sottocampo occupa le posizioni da 28 a 31 nel campo *Encoding* .

MQEN_DECIM_MASK
Maschera per la codifica packed - decimal - integer.

Questo sottocampo occupa le posizioni da 24 a 27 all'interno del campo *Encoding* .

MASCHERA_MQENC_MOBILE
Maschera per la codifica a virgola mobile.

Questo campo secondario occupa le posizioni da 20 a 23 nel campo *Encoding* .

MASK MQENC_RESERVED_
Maschera per bit riservati.

Questo sottocampo occupa le posizioni da 0 a 19 all'interno del campo *Encoding* .

Codifica numero intero binario

I seguenti valori sono validi per la codifica binario - intero:

MQEN_INTEGER_UNDEFINED
I numeri interi binari vengono rappresentati utilizzando una codifica non definita.

MQENC_INTEGER_NORMAL
I numeri interi binari sono rappresentati nel modo convenzionale:

- Il byte meno significativo nel numero ha l'indirizzo più alto di uno dei byte nel numero; il byte più significativo ha l'indirizzo più basso
- Il bit meno significativo in ogni byte è adiacente al byte con il successivo indirizzo più alto; il bit più significativo in ogni byte è adiacente al byte con il successivo indirizzo più basso

MQENC_INTEGER_REVERSED
I numeri interi binari sono rappresentati nello stesso modo di MQENC_INTEGER_NORMAL, ma con i byte disposti in ordine inverso. I bit all'interno di ogni byte sono disposti nello stesso modo di MQENC_INTEGER_NORMAL.

Codifica numero intero decimale compresso

I seguenti valori sono validi per la codifica numero intero decimale compresso:

MQEN_DECIM_UNDEFINED
I numeri interi decimali compressi vengono rappresentati utilizzando una codifica non definita.

MQENC_DECIMAL_NORMAL
I numeri interi decimali compressi sono rappresentati nel modo convenzionale:

- Ogni cifra decimale nel formato stampabile del numero è rappresentata in decimale compresso da una singola cifra esadecimale compresa nell'intervallo compreso tra X' 0 'e X' 9'. Ogni cifra esadecimale occupa quattro bit e quindi ogni byte nel numero decimale compresso rappresenta due cifre decimali nel formato stampabile del numero.
- Il byte meno significativo nel numero decimale compresso è il byte che contiene la cifra decimale meno significativa. All'interno di quel byte, i quattro bit più significativi contengono la cifra decimale meno significativa, mentre i quattro bit meno significativi contengono il segno. Il segno è X'C '(positivo), X 'D' (negativo) o X'F ' (senza segno).
- Il byte meno significativo nel numero ha l'indirizzo più alto di uno dei byte nel numero; il byte più significativo ha l'indirizzo più basso.
- Il bit meno significativo in ogni byte è adiacente al byte con il successivo indirizzo più alto; il bit più significativo in ogni byte è adiacente al byte con il successivo indirizzo più basso.

MQENC_DECIMAL_REVERSED
I numeri interi decimali compressi sono rappresentati nello stesso modo di MQENC_DECIMAL_NORMAL, ma con i byte disposti in ordine inverso. I bit all'interno di ciascun byte sono disposti nello stesso modo di MQENC_DECIMAL_NORMAL.

Codifica a virgola mobile

I seguenti valori sono validi per la codifica a virgola mobile:

MQEN_FLOAT_UNDEFINED
I numeri a virgola mobile vengono rappresentati utilizzando una codifica non definita.

MQENC_FLOAT_IEEE_NORMAL
I numeri a virgola mobile sono rappresentati utilizzando lo standard IEEE[4]formato a virgola mobile, con i byte disposti come segue:

- Il byte meno significativo nella mantissa ha l'indirizzo più alto di uno qualsiasi dei byte nel numero; il byte contenente l'esponente ha l'indirizzo più basso
- Il bit meno significativo in ogni byte è adiacente al byte con il successivo indirizzo più alto; il bit più significativo in ogni byte è adiacente al byte con il successivo indirizzo più basso

I dettagli della codifica a virgola mobile IEEE sono disponibili nello standard IEEE 754.

MQENC_FLOAT_IEEE_REVERSED
I numeri a virgola mobile sono rappresentati nello stesso modo di MQENC_FLOAT_IEEE_NORMAL, ma con i byte disposti in ordine inverso. I bit all'interno di ciascun byte sono disposti nello stesso modo di MQENC_FLOAT_IEEE_NORMAL.

MQENC_FLOAT_S390
I numeri a virgola mobile vengono rappresentati utilizzando il formato a virgola mobile System/390 standard; questo viene utilizzato anche da System/370.

Costruzione di codifiche

Per creare un valore per il campo *Encoding* in MQMD, le costanti rilevanti che descrivono le codifiche richieste possono essere aggiunte insieme (non aggiungere la stessa costante più di una volta) o combinate utilizzando l'operazione OR bitwise (se il linguaggio di programmazione supporta le operazioni bit).

Indipendentemente dal metodo utilizzato, combinare solo una delle codifiche MQEN_INTEGER_ * con una delle codifiche MQENC_DECIMAL_ * e una delle codifiche MQENC_FLOAT_ *.

Analisi delle codifiche

Il campo *Encoding* contiene campi secondari; per questo motivo, le applicazioni che devono esaminare la codifica integer, packed decimal o float devono utilizzare una delle tecniche descritte.

Utilizzo delle operazioni bit

Se il linguaggio di programmazione supporta le operazioni bit, effettuare le seguenti operazioni:

1. Selezionare uno dei seguenti valori, in base al tipo di codifica richiesto:
   - MQENC_INTEGER_MASK per la codifica di numeri interi binari
   - MQENC_DECIMAL_MASK per la codifica del numero intero decimale compresso
   - MQENC_FLOAT_MASK per la codifica a virgola mobile

   Richiamare il valore A.
2. Combinare il campo *Encoding* con A utilizzando l'operazione AND bit per bit; richiamare il risultato B.
3. B è la codifica richiesta e può essere testata per l'uguaglianza con ciascuno dei valori validi per quel tipo di codifica.

Utilizzo dell'aritmetica

Se il linguaggio di programmazione *non* supporta le operazioni bit, effettuare le seguenti operazioni utilizzando l'aritmetica dei numeri interi:

[4] L'Istituto degli Ingegneri Elettrici ed Elettronici

1. Selezionare uno dei seguenti valori, in base al tipo di codifica richiesto:
   - 1 per la codifica di numeri interi binari
   - 16 per la codifica del numero intero decimale compresso
   - 256 per la codifica a virgola mobile

   Richiamare il valore A.
2. Dividere il valore del campo *Encoding* per A ; richiamare il risultato B.
3. Dividere B per 16; richiamare il risultato C.
4. Moltiplicare C per 16 e sottrarre da B ; richiamare il risultato D.
5. Moltiplicare D per A ; richiamare il risultato E.
6. E è la codifica richiesta e può essere testata per l'uguaglianza con ciascuno dei valori validi per quel tipo di codifica.

Riepilogo delle codifiche dell'architettura della macchina

Le codifiche per le architetture delle macchine sono mostrate in Tabella 631 a pagina 928.

Tabella 631. Riepilogo delle codifiche per le architetture delle macchine

Architettura macchina	Codifica numero intero binario	Codifica numero intero decimale compresso	Codifica virgola mobile
IBM i	normale	normale	IEEE normale
Intel x86	Inverso	Inverso	IEEE invertito
PowerPC	normale	normale	IEEE normale
System/390	normale	normale	System/390

Opzioni di report e indicatori di messaggi

Questa sezione descrive i campi di *Report* e *MsgFlags* che fanno parte del descrittore del messaggio MQMD specificato nelle chiamate MQGET, MQPUT e MQPUT1 .

Gli argomenti in questa sezione descrivono:

- La struttura del campo del report e il modo in cui il gestore code lo elabora
- Come un'applicazione analizza il campo del report
- La struttura del campo message - flags

Per ulteriori informazioni sul descrittore del messaggio MQMD, consultare “MQMD - Descrittore messaggi” a pagina 431.

Struttura del campo del report

Queste informazioni descrivono la struttura del campo del report.

Il campo *Report* è un numero intero a 32 bit diviso in tre sottocampi separati. Questi sottocampi identificano:

- Opzioni di report rifiutate se il gestore code locale non le riconosce
- Opzioni di report sempre accettate, anche se il gestore code locale non le riconosce
- Opzioni di report accettate solo se sono soddisfatte determinate altre condizioni

Ogni sottocampo è identificato da una maschera di bit che ha 1 - bit nelle posizioni corrispondenti al sottocampo e 0 - bit altrove. I bit in un sottocampo non sono necessariamente adiacenti. I bit sono numerati in modo che il bit 0 sia il bit più significativo e il bit 31 il bit meno significativo. Le seguenti maschere sono definite per identificare i sottocampi:

MQRO_REJECT_UNSUP_MASK
Questa maschera identifica le posizioni di bit all'interno del campo *Report* in cui le opzioni di report non supportate dal gestore code locale causano l'esito negativo della chiamata MQPUT o MQPUT1 con codice di completamento MQCC_FAILED e codice motivo MQRC_REPORT_OPTIONS_ERROR.

Questo sottocampo occupa le posizioni bit 3 e da 11 a 13.

MQRO_ACCEPT_UNSUP_MASK
Questa maschera identifica le posizioni di bit all'interno del campo *Report* in cui le opzioni di report non supportate dal gestore code locale vengono tuttavia accettate sulle chiamate MQPUT o MQPUT1 . In questo caso, viene restituito il codice di completamento MQCC_WARNING con codice motivo MQRC_UNKNOWN_REPORT_OPTION.

Questo sottocampo occupa le posizioni da 0 a 2, da 4 a 10 e da 24 a 31.

Le seguenti opzioni del report sono incluse in questo campo secondario:

- ATTIVITÀ MQRO
- ID_COPY_MQRO_MSG_TO_CORREL_ID
- MQRO_DEAD_LETTER_Q
- MQRO_DISCARD_MSG
- MQRO_ECCEZIONE
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA
- SCADENZA_MQRO
- MQRO_EXPIRATION_WITH_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA
- MQRO_NAN
- ID_MSG_NEW_MQRO
- MQRO_NONE
- MQRO_PAN
- ID CORREL_PASS_MQRO_
- MQRO_PASS_MSG_ID

MQRO_ACCEPT_UNSUP_IF_XMIT_MASK
Questa maschera identifica le posizioni di bit all'interno del campo *Report* in cui le opzioni di report non supportate dal gestore code locale vengono comunque accettate sulle chiamate MQPUT o MQPUT1 *purché* siano soddisfatte entrambe le condizioni riportate di seguito:

- Il messaggio è destinato a un gestore code remoto.
- L'applicazione non sta inserendo il messaggio direttamente in una coda di trasmissione locale (ossia, la coda identificata dai campi *ObjectQMgrName* e *ObjectName* nel descrittore oggetto specificato nella chiamata MQOPEN o MQPUT1 non è una coda di trasmissione locale).

Il codice di completamento MQCC_WARNING con il codice motivo MQRC_UNKNOWN_REPORT_OPTION viene restituito se queste condizioni sono soddisfatte e MQCC_FAILED con il codice motivo MQRC_REPORT_OPTIONS_ERROR in caso contrario.

Questo sottocampo occupa le posizioni di bit da 14 a 23.

Le seguenti opzioni del report sono incluse in questo campo secondario:

- COA MQRO
- DATA_COA_WITH_MQRO
- DATI_COA_WITH_MQRO_FULL_DATA
- COD MQRO

- DATI MQRO_COD_WITH_
- DAD_COD MQRO_WITH_FULL_DATA

Se nel campo *Report* sono specificate opzioni che il gestore code non riconosce, il gestore code controlla ogni campo secondario a sua volta utilizzando l'operazione AND bit per bit per combinare il campo *Report* con la maschera per tale campo secondario. Se il risultato di tale operazione è diverso da zero, vengono restituiti il codice di completamento e i codici di errore descritti in precedenza.

Se viene restituito MQCC_WARNING, non viene definito quale codice di errore viene restituito se esistono altre condizioni di avvertenza.

La possibilità di specificare e accettare opzioni di report non riconosciute dal gestore code locale è utile quando si invia un messaggio con un'opzione di report riconosciuta e elaborata da un gestore code *remoto* .

Analisi del campo del report

Il campo *Report* contiene campi secondari; per questo motivo, le applicazioni che devono controllare se il mittente del messaggio ha richiesto un particolare report devono utilizzare una delle tecniche descritte.

Utilizzo delle operazioni bit

Se il linguaggio di programmazione supporta le operazioni bit, effettuare le seguenti operazioni:

1. Selezionare uno dei seguenti valori, in base al tipo di report da controllare:
   - MQRO_COA_WITH_FULL_DATA per report COA
   - Report MQRO_COD_WITH_FULL_DATA per COD
   - MQRO_EXCEPTION_WITH_FULL_DATA per report di eccezioni
   - MQRO_EXPIRATION_WITH_FULL_DATA per report di scadenza

   Richiamare il valore A.

   Su z/OS, utilizzare i valori MQRO_ * _WITH_DATA invece dei valori MQRO_ * WITH_FULL_DATA.
2. Combinare il campo *Report* con A utilizzando l'operazione AND bit per bit; richiamare il risultato B.
3. Verificare B per l'uguaglianza con ogni valore possibile per quel tipo di report.

   Ad esempio, se A è MQRO_EXCEPTION_WITH_FULL_DATA, verificare B per l'uguaglianza con ciascuno dei seguenti elementi per stabilire cosa è stato specificato dal mittente del messaggio:

   - MQRO_NONE
   - MQRO_ECCEZIONE
   - MQRO_EXCEPTION_WITH_DATA
   - MQRO_EXCEPTION_WITH_FULL_DATA

   I test possono essere eseguiti in qualsiasi ordine sia più conveniente per la logica dell'applicazione.

Utilizzare un metodo simile per eseguire il test per le opzioni MQRO_PASS_MSG_ID o MQRO_PASS_CORREL_ID; selezionare come valore A qualunque di queste due costanti sia appropriato e procedere come descritto in precedenza.

Utilizzo dell'aritmetica

Se il linguaggio di programmazione *non* supporta le operazioni bit, effettuare le seguenti operazioni utilizzando l'aritmetica dei numeri interi:

1. Selezionare uno dei seguenti valori, in base al tipo di report da controllare:
   - Report MQRO_COA per COA
   - MQRO_COD per report COD
   - MQRO_EXCEPTION per report di eccezioni

- MQRO_EXPIRATION per report di scadenza

Richiamare il valore A.

2. Dividere il campo *Report* per A ; richiamare il risultato B.
3. Dividere B per 8 ; richiamare il risultato C.
4. Moltiplicare C per 8 e sottrarre da B ; richiamare il risultato D.
5. Moltiplicare D per A ; richiamare il risultato E.
6. Verificare E per l'uguaglianza con ogni valore possibile per quel tipo di report.

   Ad esempio, se A è MQRO_EXCEPTION, verificare E per l'uguaglianza con ciascuno dei seguenti elementi per stabilire cosa è specificato dal mittente del messaggio:

   - MQRO_NONE
   - MQRO_ECCEZIONE
   - MQRO_EXCEPTION_WITH_DATA
   - MQRO_EXCEPTION_WITH_FULL_DATA

   I test possono essere eseguiti in qualsiasi ordine sia più conveniente per la logica dell'applicazione.

Il seguente pseudocodice illustra questa tecnica per i messaggi di report di eccezioni:

```
A = MQRO_EXCEPTION
B = Report/A
C = B/8
D = B - C*8
E = D*A
```

Utilizzare un metodo simile per eseguire il test per le opzioni MQRO_PASS_MSG_ID o MQRO_PASS_CORREL_ID; selezionare come valore A qualunque di queste due costanti sia appropriato, quindi procedere come descritto in precedenza, ma sostituendo il valore 8 nei passi precedenti con il valore 2.

Struttura del campo message - flags

Queste informazioni descrivono la struttura del campo message - flags.

Il campo *MsgFlags* è un numero intero a 32 bit diviso in tre sottocampi separati. Questi sottocampi identificano:

- Indicatori di messaggio rifiutati se il gestore code locale non li riconosce
- Indicatori di messaggi sempre accettati, anche se il gestore code locale non li riconosce
- Indicatori di messaggio accettati solo se sono soddisfatte determinate altre condizioni

Nota: Tutti i campi secondari in *MsgFlags* sono riservati per l'utilizzo da parte del gestore code.

Ogni sottocampo è identificato da una maschera di bit che ha 1 - bit nelle posizioni corrispondenti al sottocampo e 0 - bit altrove. I bit sono numerati in modo che il bit 0 sia il bit più significativo e il bit 31 il bit meno significativo. Le seguenti maschere sono definite per identificare i sottocampi:

MQMF_REJECT_UNSUP_MASK
Questa maschera identifica le posizioni dei bit all'interno del campo *MsgFlags* in cui gli indicatori dei messaggi non supportati dal gestore code locale causano l'esito negativo della chiamata MQPUT o MQPUT1 con codice di completamento MQCC_FAILED e codice motivo MQRC_MSG_FLAGS_ERROR.

Questo sottocampo occupa le posizioni da 20 a 31.

I seguenti indicatori di messaggio sono inclusi in questo campo secondario:

- MQMF_LAST_MSG_IN_GROUP
- MQMF_LAST_SEGMENT
- MQMF_MSG_IN_GROUP

- ISCRIZIONE MQMF_SE
- MQMF_SEGMENTAZIONE_CONSENTITA
- MQMF_SEGMENTATION_INIBITO

MQMF_ACCEPT_UNSUP_MASK
Questa maschera identifica le posizioni dei bit all'interno del campo *MsgFlags* in cui gli indicatori dei messaggi non supportati dal gestore code locale vengono tuttavia accettati nelle chiamate MQPUT o MQPUT1 . Il codice di completamento è MQCC_OK.

Questo sottocampo occupa le posizioni di bit da 0 a 11.

MQMF_ACCEPT_UNSUP_IF_XMIT_MASK
Questa maschera identifica le posizioni dei bit all'interno del campo *MsgFlags* in cui gli indicatori dei messaggi non supportati dal gestore code locale vengono tuttavia accettati sulle chiamate MQPUT o MQPUT1 *purché* siano soddisfatte entrambe le condizioni riportate di seguito:

- Il messaggio è destinato a un gestore code remoto.
- L'applicazione non sta inserendo il messaggio direttamente in una coda di trasmissione locale (ossia, la coda identificata dai campi *ObjectQMgrName* e *ObjectName* nel descrittore oggetto specificato nella chiamata MQOPEN o MQPUT1 non è una coda di trasmissione locale).

Il codice di completamento MQCC_OK viene restituito se queste condizioni sono soddisfatte e MQCC_FAILED con il codice motivo MQRC_MSG_FLAGS_ERROR in caso contrario.

Questo sottocampo occupa le posizioni bit da 12 a 19.

Se ci sono indicatori specificati nel campo *MsgFlags* che il gestore code non riconosce, il gestore code controlla ogni campo secondario a turno utilizzando l'operazione AND bit per bit per combinare il campo *MsgFlags* con la maschera per tale campo secondario. Se il risultato di tale operazione è diverso da zero, vengono restituiti il codice di completamento e i codici di errore descritti in precedenza.

Uscita conversione dati

Questa raccolta di argomenti descrive l'interfaccia per l'uscita di conversione dati e l'elaborazione eseguita dal gestore code quando è richiesta la conversione dei dati.

Per ulteriori informazioni sulla conversione dei dati, consultare *Data Conversion in IBM MQ* all'indirizzo https://www.ibm.com/support/pages/node/317869

L'uscita di conversione dati viene richiamata come parte dell'elaborazione della chiamata MQGET per convertire i dati del messaggio dell'applicazione nella rappresentazione richiesta dall'applicazione ricevente. La conversione dei dati del messaggio dell'applicazione è facoltativa; richiede che l'opzione MQGMO_CONVERT sia specificata nella chiamata MQGET.

Vengono descritti i seguenti argomenti:

- L'elaborazione eseguita dal gestore code in risposta all'opzione MQGMO_CONVERT; consultare “Elaborazione conversione” a pagina 933.
- Convenzioni di elaborazione utilizzate dal gestore code quando si elabora un formato integrato; queste convenzioni sono consigliate anche per le uscite scritte dall'utente. Consultare “Convenzioni di elaborazione” a pagina 934.
- Considerazioni speciali sulla conversione dei messaggi di report; consultare “Conversione dei messaggi di report” a pagina 938.
- I parametri passati all'exit di conversione dati; consultare “MQ_DATA_CONV_EXIT - Uscita conversione dati” a pagina 951.
- Una chiamata che può essere utilizzata dall'uscita per convertire i dati carattere tra rappresentazioni differenti; consultare “MQXCNVC - Converti caratteri” a pagina 945.
- Il parametro della struttura dati specifico per l'uscita; consultare “MQDXP - Parametro di uscita conversione dati” a pagina 939.

Elaborazione conversione

Queste informazioni descrivono l'elaborazione eseguita dal gestore code in seguito all'opzione MQGMO_CONVERT.

Il gestore code effettua le seguenti azioni se l'opzione MQGMO_CONVERT viene specificata nella chiamata MQGET e viene restituito un messaggio all'applicazione:

1. Se si verifica una o più delle seguenti condizioni, non è necessaria alcuna conversione:
   - I dati del messaggio sono già nella serie di caratteri e nella codifica richiesta dall'applicazione che emette la chiamata MQGET. L'applicazione deve impostare i campi *CodedCharSetId* e *Encoding* nel parametro **MsgDesc** della chiamata MQGET sui valori richiesti, prima di emettere la chiamata.
   - La lunghezza dei dati del messaggio è zero.
   - La lunghezza del parametro **Buffer** della chiamata MQGET è zero.

   In questi casi, il messaggio viene restituito senza conversione all'applicazione che emette la chiamata MQGET; i valori *CodedCharSetId* e *Encoding* nel parametro **MsgDesc** sono impostati sui valori nelle informazioni di controllo nel messaggio e la chiamata viene completata con una delle seguenti combinazioni di codice di completamento e codice motivo:

Tabella 632. Combinazioni di codice di completamento e codice motivo

Codice di completamento	Codice di errore
MQCC_OK	MQRC_NONE
MQCC_AVVERTENZA	MQRC_TRUNCATED_MSG_ACCEPTED
MQCC_AVVERTENZA	MQRC_TRUNCATED_MSG_NON RIUSCITO

   Le seguenti operazioni vengono eseguite solo se la serie di caratteri o la codifica dei dati del messaggio differiscono dal valore corrispondente nel parametro **MsgDesc** e vi sono dati da convertire:

2. Se il campo *Format* nelle informazioni di controllo nel messaggio ha il valore MQFMT_NONE, il messaggio viene restituito non convertito, con codice di completamento MQCC_WARNING e codice motivo MQRC_FORMAT_ERROR.

   In tutti gli altri casi l'elaborazione della conversione continua.

3. Il messaggio viene rimosso dalla coda e inserito in un buffer temporaneo della stessa dimensione del parametro **Buffer** . Per le operazioni di ricerca, il messaggio viene copiato nel buffer temporaneo, invece di essere rimosso dalla coda.
4. Se il messaggio deve essere troncato per adattarsi al buffer, viene eseguito quanto segue:
   - Se l'opzione MQGMO_ACCEPT_TRUNCATED_MSG non è stata specificata, il messaggio viene restituito non convertito, con il codice di completamento MQCC_WARNING e il codice motivo MQRC_TRUNCATED_MSG_FAILED.
   - Se l'opzione MQGMO_ACCEPT_TRUNCATED_MSG *è stata* specificata, il codice di completamento è impostato su MQCC_WARNING, il codice motivo è impostato su MQRC_TRUNCATED_MSG_ACCEPTED e l'elaborazione della conversione continua.
5. Se il messaggio può essere inserito nel buffer senza troncamento o se è stata specificata l'opzione MQGMO_ACCEPT_TRUNCATED_MSG, viene eseguita la seguente procedura:
   - Se il formato è un formato integrato, il buffer viene passato al servizio di conversione dati del gestore code.
   - Se il formato non è un formato integrato, il buffer viene passato a un'uscita scritta dall'utente con lo stesso nome del formato. Se non è possibile trovare l'uscita, il messaggio viene restituito non convertito, con codice di completamento MQCC_WARNING e codice motivo MQRC_FORMAT_ERROR.

   Se non si verifica alcun errore, l'output del servizio di conversione dati o dell'uscita scritta dall'utente è il messaggio convertito, più il codice di completamento e il codice motivo da restituire all'applicazione che emette la chiamata MQGET.

6. Se la conversione ha esito positivo, il gestore code restituisce il messaggio convertito all'applicazione. In questo caso, il codice di completamento e il codice motivo restituiti dalla chiamata MQGET sono una delle seguenti combinazioni:

Tabella 633. Combinazioni di codice di completamento e codice motivo

Codice di completamento	**Codice di errore**
MQCC_OK	MQRC_NONE
MQCC_AVVERTENZA	MQRC_TRUNCATED_MSG_ACCEPTED

Tuttavia, se la conversione viene eseguita da un'uscita scritta dall'utente, possono essere restituiti altri codici di errore, anche quando la conversione ha esito positivo.

Se la conversione non riesce, il gestore code restituisce il messaggio non convertito all'applicazione, con i campi *CodedCharSetId* e *Encoding* nel parametro **MsgDesc** impostati sui valori nelle informazioni di controllo nel messaggio e con il codice di completamento MQCC_WARNING.

Convenzioni di elaborazione

Quando si converte un formato integrato, il gestore code segue le convenzioni di elaborazione descritte.

Anche le uscite scritte dall'utente devono seguire queste convenzioni, anche se non vengono applicate dal gestore code. I formati integrati convertiti dal gestore code sono:

- MMQFMT_ADMIN
- MQFMT_CICS (solo z/OS)
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- INTESTAZIONE_DIST_MQFM
- MQFMT_EVENT versione 1
- MQFMT_EVENT versione 2
- IMS MQFMT
- MQFMT_IMS_VAR_STRING
- MQFMT_MD_EXTENSIONE
- MQFMT_PCF
- MQFMT_REF_MSG_HEADER
- MQFMT_RF_HEADER
- MQFMT_RF_HEADER_2
- MQFMT_STRING
- MQFMT_TRIGGER
- MQFMT_WORK_INFO_HEADER (solo z/OS)
- MQFMT_XMIT_Q_HEADER

1. Se il messaggio si espande durante la conversione e supera la dimensione del parametro **Buffer** , viene eseguito quanto segue:
   - Se non è stata specificata l'opzione MQGMO_ACCEPT_TRUNCATED_MSG, il messaggio viene restituito non convertito, con codice di completamento MQCC_WARNING e codice motivo MQRC_CONVERTED_MSG_TOO_BIG.
   - Se l'opzione MQGMO_ACCEPT_TRUNCATED_MSG *è stata* specificata, il messaggio viene troncato, il codice di completamento è impostato su MQCC_WARNING, il codice di errore è impostato su MQRC_TRUNCATED_MSG_ACCEPTED e l'elaborazione della conversione continua.

2. Se si verifica un troncamento (prima o durante la conversione), il numero di byte validi restituiti nel parametro **Buffer** può essere inferiore alla lunghezza del buffer.

   Ciò può verificarsi, ad esempio, se un numero intero a 4 byte o un carattere DBCS si trova alla fine del buffer. L'elemento incompleto delle informazioni non viene convertito e i byte nel messaggio restituito non contengono informazioni valide. Ciò può verificarsi anche se un messaggio troncato prima della conversione si riduce durante la conversione.

   Se il numero di byte validi restituiti è inferiore alla lunghezza del buffer, i byte inutilizzati alla fine del buffer vengono impostati su valori null.
3. Se una schiera o una stringa si trova a cavallo della fine del buffer, viene convertita la maggior parte dei dati possibile; non viene convertito solo il particolare elemento della schiera o il carattere DBCS incompleto; vengono convertiti i caratteri o gli elementi della schiera precedenti.
4. Se si verifica un troncamento (prima o durante la conversione), la lunghezza restituita per il parametro **DataLength** è la lunghezza del messaggio non convertito prima del troncamento.
5. Quando le stringhe vengono convertite tra serie di caratteri a byte singolo (SBCS), serie di caratteri a byte doppio (DBCS) o serie di caratteri a più byte (MBCS), le stringhe possono espandersi o contrarsi.
   - Nei formati PCF MQFMT_ADMIN, MQFMT_EVENT e MQFMT_PCF, le stringhe nelle strutture MQCFST e MQCFSL si espandono o si contraggono in base alle necessità per adattare la stringa dopo la conversione.

     Per la struttura dell'elenco di stringhe MQCFSL, le stringhe nell'elenco potrebbero espandersi o contrarsi in base a quantità differenti. Se ciò si verifica, il gestore code riempirebbe le stringhe più corte con spazi vuoti per renderle della stessa lunghezza della stringa più lunga dopo la conversione.
   - Nel formato MQFMT_REF_MSG_HEADER, le stringhe indirizzate dai campi `SrcEnvOffset`, `SrcNameOffset`, `DestEnvOffsete` `DestNameOffset` si espandono o si contraggono in base alle necessità per adattare le stringhe dopo la conversione.
   - Nel formato MQFMT_RF_HEADER, il campo `NameValueString` si espande o si contrae come necessario per contenere le coppie nome - valore dopo la conversione.
   - Nelle strutture con dimensioni di campo fisse, il gestore code consente alle stringhe di espandersi o di contrarsi all'interno dei relativi campi fissi, a condizione che non vengano perse informazioni significative. A questo proposito, gli spazi vuoti finali e i caratteri che seguono il primo carattere null nel campo vengono trattati come non significativi.
     - Se la stringa si espande, ma solo i caratteri non significativi devono essere scartati per contenere la stringa convertita nel campo, la conversione ha esito positivo e la chiamata viene completata con MQCC_OK e codice motivo MQRC_NONE (supponendo che non vi siano altri errori).
     - Se la stringa si espande, ma la stringa convertita richiede l'eliminazione di caratteri significativi per poter essere inserita nel campo, il messaggio viene restituito non convertito e la chiamata viene completata con MQCC_WARNING e codice motivo MQRC_CONVERTED_STRING_TOO_BIG.

       Nota: In questo caso, il codice motivo MQRC_CONVERTED_STRING_TOO_BIG indica se è stata specificata o meno l'opzione MQGMO_ACCEPT_TRUNCATED_MSG.
     - Se la stringa si contrae, il gestore code riempiendo la stringa con spazi vuoti fino alla lunghezza del campo.
6. Per i messaggi costituiti da una o più strutture di intestazione MQ seguite da dati utente, è possibile che una o più strutture di intestazione vengano convertite, mentre il resto del messaggio non lo è. Tuttavia, (con due eccezioni) i campi *CodedCharSetId* e *Encoding* in ogni struttura di intestazione indicano sempre correttamente la serie di caratteri e la codifica dei dati che seguono la struttura di intestazione.

   Le due eccezioni sono le strutture MQCIH e MQIIH, in cui i valori nei campi *CodedCharSetId* e *Encoding* in tali strutture non sono significativi. Per tali strutture, i dati che seguono la struttura si trovano nella stessa serie di caratteri e nella stessa codifica della struttura MQCIH o MQIIH.
7. Se i campi *CodedCharSetId* o *Encoding* nelle informazioni di controllo del messaggio richiamato o nel parametro **MsgDesc** , specificano valori non definiti o non supportati, il gestore code potrebbe

ignorare l'errore se non è necessario utilizzare il valore non definito o non supportato nella conversione del messaggio.

Ad esempio, se il campo *Encoding* nel messaggio specifica una codifica a virgola mobile non supportata, ma il messaggio contiene solo dati interi o contiene dati a virgola mobile che non richiedono la conversione (poiché le codifiche a virgola mobile di origine e di destinazione sono identiche), l'errore potrebbe non essere diagnosticato.

Se l'errore viene diagnosticato, il messaggio viene restituito non convertito, con codice di completamento MQCC_WARNING e uno dei codici di errore MQRC_SOURCE_ * _ERROR o MQRC_TARGET_ * _ERROR (come appropriato); i campi *CodedCharSetId* e *Encoding* del parametro **MsgDesc** sono impostati sui valori nelle informazioni di controllo nel messaggio.

Se l'errore non viene diagnosticato e la conversione viene completata correttamente, i valori restituiti nei campi *CodedCharSetId* e *Encoding* nel parametro **MsgDesc** sono quelli specificati dall'applicazione che emette la chiamata MQGET.

8. In tutti i casi, se il messaggio viene restituito all'applicazione non convertita, il codice di completamento viene impostato su MQCC_WARNING e i campi *CodedCharSetId* e *Encoding* nel parametro **MsgDesc** vengono impostati sui valori appropriati per i dati non convertiti. Questa operazione viene eseguita anche per MQFMT_NONE.

   Il parametro **Reason** è impostato su un codice che indica il motivo per cui non è stato possibile eseguire la conversione, a meno che non sia stato necessario troncare anche il messaggio; i codici di errore relativi al troncamento hanno la precedenza sui codici di errore relativi alla conversione. (Per determinare se un messaggio troncato è stato convertito, controllare i valori restituiti nei campi *CodedCharSetId* e *Encoding* nel parametro **MsgDesc** .)

   Quando viene diagnosticato un errore, viene restituito un codice di errore specifico oppure il codice di errore generale MQRC_NOT_CONVERTED. Il codice motivo restituito dipende dalle capacità diagnostiche del servizio di conversione dati sottostante.

9. Se viene restituito il codice di completamento MQCC_WARNING e più di un codice di errore è rilevante, l'ordine di precedenza è il seguente:

   a. I seguenti motivi hanno la precedenza su tutti gli altri; solo uno dei motivi in questo gruppo può sorgere:

      - MQRC_SIGNAL_REQUEST_ACCEPTED
      - MQRC_TRUNCATED_MSG_ACCEPTED

   b. L'ordine di precedenza all'interno dei codici di errore rimanenti non è definito.

10. Al completamento della chiamata MQGET:

    - Il seguente codice di errore indica che il messaggio è stato convertito correttamente:
      - MQRC_NONE
    - I seguenti codici di errore indicano che il messaggio *potrebbe* essere stato convertito correttamente (controllare i campi *CodedCharSetId* e *Encoding* nel parametro **MsgDesc** per scoprirlo):
      - MQRC_MSG_MARKED_BROWSE_CO_OP
      - MQRC_TRUNCATED_MSG_ACCEPTED
    - Tutti gli altri codici di errore indicano che il messaggio non è stato convertito.

    La seguente elaborazione è specifica per i formati integrati; non si applica ai formati definiti dall'utente:

11. Ad eccezione dei formati seguenti:

    - MMQFMT_ADMIN
    - MQFMT_COMMAND_1
    - MQFMT_COMMAND_2
    - EVENTO MQFMT

- MQFMT_IMS_VAR_STRING
- MQFMT_PCF
- MQFMT_STRING

nessuno dei formati integrati può essere convertito da o in serie di caratteri che non dispongono di caratteri SBCS per i caratteri validi nei nomi delle code. Se viene effettuato un tentativo di eseguire tale conversione, il messaggio viene restituito non convertito, con il codice di completamento MQCC_WARNING e il codice motivo MQRC_SOURCE_CCSID_ERROR o MQRC_TARGET_CCSID_ERROR, come appropriato.

La serie di caratteri Unicode UTF-16 è un esempio di una serie di caratteri che non dispone di caratteri SBCS per i caratteri validi nei nomi coda.

12. Se i dati del messaggio per un formato integrato vengono troncati, i campi all'interno del messaggio che contengono lunghezze di stringhe o conteggi di elementi o strutture non vengono regolati in modo da riflettere la lunghezza dei dati effettivamente restituiti all'applicazione; i valori restituiti per tali campi all'interno dei dati del messaggio sono i valori applicabili al messaggio *prima del troncamento*.

    Quando si elaborano messaggi come ad esempio un messaggio MQFMT_ADMIN troncato, assicurarsi che l'applicazione non tenti di accedere ai dati oltre la fine dei dati restituiti.

13. Se il nome del formato è MQFMT_DEAD_LETTER_HEADER, i dati del messaggio iniziano con una struttura MQDLH, possibilmente seguita da zero o più byte di dati del messaggio dell'applicazione. Il formato, la serie di caratteri e la codifica dei dati del messaggio dell'applicazione sono definiti dai campi `Format`, `CodedCharSetId`e `Encoding` nella struttura MQDLH all'inizio del messaggio. Poiché la struttura MQDLH e i dati del messaggio dell'applicazione possono avere diverse serie di caratteri e codifiche, una, un'altra o entrambe le strutture MQDLH e i dati del messaggio dell'applicazione potrebbero richiedere la conversione.

    Il gestore code converte prima la struttura MQDLH, come necessario. Se la conversione ha esito positivo o se la struttura MQDLH non richiede la conversione, il gestore code controlla i campi `CodedCharSetId` e `Encoding` nella struttura MQDLH per verificare se è richiesta la conversione dei dati del messaggio dell'applicazione. Se la conversione è richiesta, il gestore code richiama l'uscita scritta dall'utente con il nome fornito dal campo `Format` nella struttura MQDLH oppure esegue la conversione stessa (se `Format` è il nome di un formato integrato).

    Se la chiamata MQGET restituisce un codice di completamento MQCC_WARNING e il codice motivo è uno di quelli che indicano che la conversione non è stata eseguita correttamente, si applica una delle seguenti condizioni:

    - Non è possibile convertire la struttura MQDLH. In questo caso, anche i dati del messaggio dell'applicazione non saranno stati convertiti.
    - La struttura MQDLH è stata convertita, ma non i dati del messaggio dell'applicazione.

    L'applicazione può esaminare i valori restituiti nei campi `CodedCharSetId` e `Encoding` nel parametro **MsgDesc** e quelli nella struttura MQDLH, al fine di determinare quale dei precedenti si applica.

14. Se il nome del formato è MQFMT_XMIT_Q_HEADER, i dati del messaggio iniziano con una struttura MQXQH, possibilmente seguita da zero o più byte di dati aggiuntivi. Questi dati aggiuntivi sono di solito i dati del messaggio dell'applicazione (che possono essere di lunghezza zero), ma possono essere presenti anche una o più ulteriori strutture di intestazione MQ , all'inizio dei dati aggiuntivi.

    La struttura MQXQH deve essere nella serie di caratteri e nella codifica del gestore code. Il formato, la serie di caratteri e la codificazione dei dati che seguono la struttura MQXQH sono forniti dai campi `Format`, `CodedCharSetId`e `Encoding` nella struttura MQMD contenuta in MQXQH. Per ogni struttura di intestazione MQ successiva presente, i campi `Format`, `CodedCharSetId`e `Encoding` nella struttura descrivono i dati che seguono tale struttura; tali dati sono un'altra struttura di intestazioni MQ o i dati del messaggio dell'applicazione.

Se l'opzione MQGMO_CONVERT viene specificata per un messaggio MQFMT_XMIT_Q_HEADER, i dati del messaggio dell'applicazione e alcune delle strutture dell'intestazione MQ vengono convertite, *ma i dati nella struttura MQXQH non sono*. Al ritorno dalla chiamata MQGET, quindi:

- I valori dei campi `Format`, `CodedCharSetId`e `Encoding` nel parametro **MsgDesc** descrivono i dati nella struttura MQXQH e non i dati del messaggio dell'applicazione; i valori, quindi, non sono uguali a quelli specificati dall'applicazione che ha emesso la chiamata MQGET.

  L'effetto di questa operazione è che un'applicazione che riceve ripetutamente i messaggi da una coda di trasmissione con l'opzione MQGMO_CONVERT specificata deve reimpostare i campi `CodedCharSetId` e `Encoding` nel parametro **MsgDesc** sui valori richiesti per i dati del messaggio dell'applicazione, prima di ogni chiamata MQGET.
- I valori dei campi `Format`, `CodedCharSetId`e `Encoding` presenti nell'ultima struttura di intestazioni MQ descrivono i dati del messaggio dell'applicazione. Se non sono presenti altre strutture di intestazione MQ , i dati del messaggio dell'applicazione vengono descritti da questi campi nella struttura MQMD all'interno della struttura MQXQH. Se la conversione ha esito positivo, i valori saranno gli stessi specificati nel parametro **MsgDesc** dall'applicazione che ha emesso la chiamata MQGET.

Se il messaggio è un messaggio elenco di distribuzione, la struttura MQXQH è seguita da una struttura MQDH (più i relativi array di record MQOR e MQPMR), che a sua volta potrebbe essere seguita da zero o più ulteriori strutture di intestazione MQ e zero o più byte di dati del messaggio dell'applicazione. Come la struttura MQXQH, la struttura MQDH deve essere nella serie di caratteri e nella codifica del gestore code e non viene convertita nella chiamata MQGET, anche se è specificata l'opzione MQGMO_CONVERT.

L'elaborazione delle strutture MQXQH e MQDH descritte in precedenza è destinata principalmente all'utilizzo da parte degli agent del canale dei messaggi quando ricevono i messaggi dalle code di trasmissione.

Conversione dei messaggi di report

In generale un messaggio di report può contenere quantità variabili di dati del messaggio dell'applicazione, in base alle opzioni del report specificate dal mittente del messaggio originale. Tuttavia, un report di attività può contenere dati ma senza l'opzione di report che indica * _WITH_DATA nella costante.

In particolare, un messaggio di report può contenere:

1. Nessun dato del messaggio dell'applicazione
2. Alcuni dati del messaggio dell'applicazione dal messaggio originale

   Ciò si verifica quando il mittente del messaggio originale specifica MQRO_ * _WITH_DATA e il messaggio è più lungo di 100 byte.
3. Tutti i dati del messaggio dell'applicazione dal messaggio originale

   Ciò si verifica quando il mittente del messaggio originale specifica MQRO_ * _WITH_FULL_DATA o specifica MQRO_ * _WITH_DATA e il messaggio è di 100 byte o più breve.

Quando il gestore code o l'agent del canale dei messaggi genera un messaggio di report, copia il nome del formato dal messaggio originale nel campo *`Format`* nelle informazioni di controllo nel messaggio di report. Il nome del formato nel messaggio di report potrebbe quindi implicare una lunghezza di dati diversa dalla lunghezza effettivamente presente nel messaggio di report (casi 1 e 2 in precedenza).

Se l'opzione MQGMO_CONVERT viene specificata quando viene richiamato il messaggio di report:

- Per il caso 1 precedente, l'uscita di conversione dati non viene richiamata (poiché il messaggio di report non contiene dati).
- Per il caso precedente 3, il nome del formato implica correttamente la lunghezza dei dati del messaggio.
- Ma per il caso 2 precedente, l'uscita di conversione dati viene richiamata per convertire un messaggio *più breve* della lunghezza implicita dal nome formato.

Inoltre, il codice motivo trasmesso all'uscita è di solito MQRC_NONE (ossia, il codice motivo non indica che il messaggio è stato troncato). Ciò si verifica perché i dati del messaggio sono stati troncati dal *mittente* del messaggio di report e non dal gestore code del ricevente in risposta alla chiamata MQGET.

A causa di queste possibilità, l'exit di conversione dati non deve utilizzare il nome del formato per dedurre la lunghezza dei dati ad esso trasmessi; invece l'exit deve controllare la lunghezza dei dati forniti ed essere preparata a convertire meno dati della lunghezza implicita dal nome del formato. Se i dati possono essere convertiti correttamente, il codice di completamento MQCC_OK e il codice motivo MQRC_NONE devono essere restituiti dall'uscita. La lunghezza dei dati del messaggio da convertire viene passata all'exit come parametro **InBufferLength** .

Interfaccia di programmazione sensibile al prodotto

MQDXP - Parametro di uscita conversione dati

La struttura MQDXP è un parametro che il gestore code passa all'uscita di conversione dati quando l'uscita viene richiamata per convertire i dati del messaggio come parte dell'elaborazione della chiamata MQGET. Consultare la descrizione della chiamata MQ_DATA_CONV_EXIT per i dettagli dell'uscita di conversione dati.

I dati di caratteri in MQDXP si trovano nella serie di caratteri del gestore code locale; ciò viene fornito dall'attributo gestore code **CodedCharSetId** . I dati numerici in MQDXP sono nella codifica della macchina nativa; ciò viene fornito da MQENC_NATIVE.

Solo i campi *DataLength*, *CompCode*, *Reason*, e *ExitResponse* in MQDXP possono essere modificati dall'uscita; le modifiche agli altri campi vengono ignorate. Tuttavia, il campo *DataLength* non può essere modificato se il messaggio che si sta convertendo è un segmento che contiene solo una parte di un messaggio logico.

Quando il controllo ritorna al gestore code dall'uscita, il gestore code verifica i valori restituiti in MQDXP. Se i valori restituiti non sono validi, il gestore code continua l'elaborazione come se l'uscita avesse restituito MQXDR_CONVERSION_FAILED in *ExitResponse* ; tuttavia, il gestore code ignora i valori dei campi *CompCode* e *Reason* restituiti dall'exit in questo caso e utilizza invece i valori che tali campi avevano in *input* per l'exit. I seguenti valori in MQDXP causano questa elaborazione:

- Il campo *ExitResponse* non è MQXDR_OK e non è MQXDR_CONVERSION_FAILED
- *CompCode* campo non MQCC_OK e non MQCC_WARNING
- Il campo *DataLength* minore di zero oppure il campo *DataLength* è stato modificato quando il messaggio da convertire è un segmento che contiene solo una parte di un messaggio logico.

La seguente tabella riepiloga i campi nella struttura.

Tabella 634. Campi in MQDXP

Campo	Descrizione	Argomento
StrucId	Identificativo struttura	StrucId
Version	Numero di versione della struttura	Versione
AppOptions	Opzioni applicazione	AppOptions
Encoding	Codifica numerica richiesta dall'applicazione	Codifica
CodedCharSetId	Serie di caratteri richiesta dall'applicazione	CodedCharSetId
DataLength	Lunghezza in byte dei dati del messaggio	DataLength
CompCode	Codice di completamento	CompCode

Tabella 634. Campi in MQDXP (Continua)		
Campo	**Descrizione**	**Argomento**
Reason	Codice di errore qualificante *CompCode*	Motivo
ExitResponse	Risposta dall'uscita	ExitResponse
Hconn	ID interno connessioni	Hconn
pEntryPoints	Indirizzo della struttura MQIEP	pEntryPunti

Campi

La struttura MQDXP contiene i seguenti campi, descritti in ordine alfabetico.

AppOptions
Tipo: MQLONG

Questa è una copia del campo *Options* della struttura MQGMO specificata dall'applicazione che emette la chiamata MQGET. L'uscita potrebbe aver bisogno di esaminarli per verificare se è stata specificata l'opzione MQGMO_ACCEPT_TRUNCATED_MSG.

Questo è un campo di immissione per l'uscita.

CodedCharSetId
Tipo: MQLONG

Questo è il CCSID (coded character set identifier) della serie di caratteri richiesta dall'applicazione che emette la chiamata MQGET; consultare il campo *CodedCharSetId* nella struttura MQMD per ulteriori dettagli. Se l'applicazione specifica il valore speciale MQCCSI_Q_MGR sulla chiamata MQGET, il gestore code lo modifica con l'identificativo del set di caratteri effettivo del set di caratteri utilizzato dal gestore code, prima di richiamare l'exit.

Se la conversione ha esito positivo, l'uscita deve copiarlo nel campo *CodedCharSetId* nel descrittore del messaggio.

Questo è un campo di immissione per l'uscita.

CompCode
Tipo: MQLONG

Quando viene richiamata l'uscita, questa contiene il codice di completamento restituito all'applicazione che ha emesso la chiamata MQGET, se l'uscita non esegue alcuna operazione. È sempre MQCC_WARNING, perché il messaggio è stato troncato o il messaggio richiede la conversione e questa operazione non è stata ancora eseguita.

Nell'output dell'uscita, questo campo contiene il codice di completamento da restituire all'applicazione nel parametro **CompCode** della chiamata MQGET; sono validi soltanto MQCC_OK e MQCC_WARNING. Consultare la descrizione del campo *Reason* per suggerimenti su come l'uscita può impostare questo campo sull'output.

Questo è un campo di immissione / emissione per l'uscita.

DataLength
Tipo: MQLONG

Quando si richiama l'uscita, questo campo contiene la lunghezza originale dei dati del messaggio dell'applicazione. Se il messaggio è stato troncato per adattarsi al buffer fornito dall'applicazione, la dimensione del messaggio fornito all'uscita è *minore* del valore di *DataLength*. La dimensione del messaggio fornito all'uscita è sempre fornita dal parametro **InBufferLength** dell'uscita, indipendentemente dal troncamento che si è verificato.

Il troncamento viene indicato dal campo *Reason* con il valore MQRC_TRUNCATED_MSG_ACCEPTED sull'input all'uscita.

La maggior parte delle conversioni non ha bisogno di modificare questa lunghezza, ma un'uscita può farlo se necessario; il valore impostato dall'uscita viene restituito all'applicazione nel parametro **DataLength** della chiamata MQGET. Tuttavia, questa lunghezza non può essere modificata se il messaggio che si sta convertendo è un segmento che contiene solo una parte di un messaggio logico. Ciò è dovuto al fatto che la modifica della lunghezza potrebbe causare l'errore degli offset dei segmenti successivi nel messaggio logico.

Tenere presente che, se l'uscita desidera modificare la lunghezza dei dati, tenere presente che il gestore code ha già deciso se i dati del messaggio si adattano al buffer dell'applicazione, in base alla lunghezza dei dati *non convertiti* . Questa decisione determina se il messaggio viene rimosso dalla coda (o il cursore di ricerca spostato, per una richiesta di ricerca) e non viene influenzato da alcuna modifica alla lunghezza dei dati causata dalla conversione. Per questo motivo si consiglia che le uscite di conversione non causino una modifica nella lunghezza dei dati del messaggio dell'applicazione.

Se la conversione dei caratteri implica una modifica della lunghezza, una stringa può essere convertita in un'altra stringa con la stessa lunghezza in byte, troncando gli spazi finali o riempiendo gli spazi come necessario.

L'uscita non viene richiamata se il messaggio non contiene dati del messaggio dell'applicazione; quindi, *DataLength* è sempre maggiore di zero.

Questo è un campo di immissione / emissione per l'uscita.

Encoding
Tipo: MQLONG

Codifica numerica richiesta dall'applicazione.

Si tratta della codifica numerica richiesta dall'applicazione che emette la chiamata MQGET; per ulteriori dettagli, consultare il campo *Encoding* nella struttura MQMD.

Se la conversione ha esito positivo, l'uscita lo copia nel campo *Encoding* nel descrittore del messaggio.

Questo è un campo di immissione per l'uscita.

ExitOptions
Tipo: MQLONG

Questo è un campo riservato; il valore è `0`.

ExitResponse
Tipo: MQLONG

Risposta dall'uscita.Questo valore viene impostato dall'uscita per indicare l'esito positivo o meno della conversione. Deve essere uno dei seguenti:

MQXDR_OK
Conversione riuscita correttamente.

Se l'uscita specifica questo valore, il gestore code restituisce quanto segue all'applicazione che ha emesso la chiamata MQGET:

- Il valore del campo *CompCode* sull'output dall'uscita
- Il valore del campo *Reason* sull'output dall'uscita
- Il valore del campo *DataLength* sull'output dall'uscita
- Il contenuto del buffer di output dell'exit *OutBuffer*. Il numero di byte restituiti è minore del parametro **OutBufferLength** dell'exit e il valore del campo *DataLength* sull'output dall'exit.

  Se i campi *Encoding* e *CodedCharSetId* nel parametro del descrittore del messaggio dell'uscita sono *entrambi* non modificati, il gestore code restituisce:
- Il valore dei campi *Encoding* e *CodedCharSetId* nella struttura MQDXP su *input* per l'uscita.

  Se uno o entrambi i campi *Encoding* e *CodedCharSetId* nel parametro del descrittore del messaggio dell'exit sono stati modificati, il gestore code restituisce:

- Il valore dei campi *Encoding* e *CodedCharSetId* nel parametro del descrittore del messaggio dell'uscita sull'output dall'uscita

MQXDR_CONVERSION_FAILED
Conversione non riuscita.

Se l'uscita specifica questo valore, il gestore code restituisce quanto segue all'applicazione che ha emesso la chiamata MQGET:

- Il valore del campo *CompCode* sull'output dall'uscita
- Il valore del campo *Reason* sull'output dall'uscita
- Il valore del campo *DataLength* in *input* per l'exit
- Il contenuto del buffer di input dell'uscita *InBuffer*. Il numero di byte restituiti viene fornito dal parametro **InBufferLength**

Se l'uscita ha modificato *InBuffer*, i risultati non sono definiti.

ExitResponse è un campo di output dall'uscita.

Hconn
Tipo: MQHCONN

Questo è un handle di connessione che può essere utilizzato sulla chiamata MQXCNVC. Questo handle non è necessariamente uguale a quello specificato dall'applicazione che ha emesso la chiamata MQGET.

pEntryPoints
Tipo: PMQIEP

L'indirizzo di una struttura MQIEP tramite cui è possibile effettuare chiamate MQI e DCI.

Reason
Tipo: MQLONG

Codice di errore *CompCode*.

Quando l'uscita viene richiamata, contiene il codice motivo restituito all'applicazione che ha emesso la chiamata MQGET, se l'uscita sceglie di non eseguire alcuna operazione. Tra i valori possibili ci sono MQRC_TRUNCATED_MSG_ACCEPTED, che indica che il messaggio è stato troncato per adattarsi al buffer fornito dall'applicazione, e MQRC_NOT_CONVERTED, che indica che il messaggio richiede la conversione, ma che questa operazione non è stata ancora eseguita.

All'output dall'uscita, questo campo contiene il motivo per cui deve essere restituito all'applicazione nel parametro **Reason** della chiamata MQGET; si consiglia di:

- Se *Reason* aveva il valore MQRC_TRUNCATED_MSG_ACCEPTED sull'input per l'uscita, i campi *Reason* e *CompCode* non devono essere modificati, indipendentemente dal fatto che la conversione abbia esito positivo o negativo.

  (Se il campo *CompCode* non è MQCC_OK, l'applicazione che richiama il messaggio può identificare un errore di conversione confrontando i valori *Encoding* e *CodedCharSetId* restituiti nel descrittore del messaggio con i valori richiesti; al contrario, l'applicazione non può distinguere un messaggio troncato da un messaggio che ha adattato il buffer. Per questo motivo, MQRC_TRUNCATED_MSG_ACCEPTED deve essere restituito come preferenza rispetto a uno dei motivi che indicano un errore di conversione.)
- Se *Reason* aveva un altro valore nell'input per l'uscita:
  - Se la conversione ha esito positivo, *CompCode* deve essere impostato su MQCC_OK e *Reason* su MQRC_NONE.
  - Se la conversione non riesce o il messaggio si espande e deve essere troncato per adattarsi al buffer, *CompCode* deve essere impostato su MQCC_WARNING (o lasciato invariato) e *Reason* deve essere impostato su uno dei valori elencati, per indicare la natura dell'errore.

Tenere presente che se il messaggio dopo la conversione è troppo grande per il buffer, deve essere troncato solo se l'applicazione che ha emesso la chiamata MQGET ha specificato l'opzione MQGMO_ACCEPT_TRUNCATED_MSG:

- Se è stata specificata tale opzione, viene restituito il motivo MQRC_TRUNCATED_MSG_ACCEPTED.
- Se non è stata specificata tale opzione, il messaggio viene restituito non convertito, con codice motivo MQRC_CONVERTED_MSG_TOO_BIG.

I codici di errore elencati sono consigliati per l'utilizzo da parte dell'uscita per indicare il motivo per cui la conversione non è riuscita, ma l'uscita può restituire altri valori dalla serie di codici MQRC_ *, se ritenuto appropriato. Inoltre, l'intervallo di valori da MQRC_APPL_FIRST a MQRC_APPL_LAST viene assegnato per essere utilizzato dall'exit per indicare le condizioni che l'exit desidera comunicare con l'applicazione che emette la chiamata MQGET.

Nota: Se il messaggio non può essere convertito correttamente, l'uscita deve restituire MQXDR_CONVERSION_FAILED nel campo *ExitResponse* , per far sì che il gestore code restituisca il messaggio non convertito. Ciò è vero indipendentemente dal codice di errore restituito nel campo *Reason* .

MQRC_APPL_PRIMO
(900, X'384 ') Valore più basso per il codice di errore definito dall'applicazione.

LAST MQRC_APPL
(999, X'3E7') Valore più alto per il codice di errore definito dall'applicazione.

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848 ') Dati convertiti troppo grandi per il buffer.

MQRC_NOT_CONVERTED
(2119, X'847 ') Dati del messaggio non convertiti.

ERRORE CCSID DI MQRC_SOURCE_
(2111, X'83F') Identificativo serie di caratteri codificati origine non valido.

ERRORE DI RIMOZIONE MQRC_SOURCE_DECIMAL_ENC_
(2113, X'841 ') Codifica decimale compresso nel messaggio non riconosciuta.

ERRORE_ERRORE_ORIGINE_RISORSE MQRC
(2114, X'842 ') La codifica a virgola mobile nel messaggio non è stata riconosciuta.

ERRORE DI INIZIALIZZAZIONE MQRC_SOURCE_INTEGER_
(2112, X'840 ') Numero intero di origine non riconosciuto.

ERRORE MQRC_TARGET_CCSID_
(2115, X'843 ') Identificativo serie di caratteri codificati di destinazione non valido.

ERRORE DI RETE MQRC_TARGET_DECIMAL_ENC_ERROR
(2117, X'845 ') La codifica decimale compresso specificata dal ricevitore non è stata riconosciuta.

ERRORE DI RETE MQRC_TARGET_FLOAT_
(2118, X'846 ') La codifica a virgola mobile specificata dal ricevitore non è stata riconosciuta.

ERRORE MQRC_TARGET_INTEGER_ENC
(2116, X'844 ') Codifica numero intero di destinazione non riconosciuta.

MQRC_TRUNCATED_MSG_ACCEPTED
(2079, X'81F') Messaggio troncato restituito (elaborazione completata).

Questo è un campo di immissione / emissione per l'uscita.

StrucId
Tipo: MQCHAR4

Identificatore struttura.Il valore deve essere:

ID_STRUC_MQDXP
Identificativo per la struttura del parametro di uscita conversione dati.

Per il linguaggio di programmazione C, viene definita anche la costante MQDXP_STRUC_ID_ARRAY, che ha lo stesso valore di MQDXP_STRUC_ID, ma è un array di caratteri invece di una stringa.

Questo è un campo di immissione per l'uscita.

Version
Tipo: MQLONG

Numero di versione della struttura.Il valore deve essere:

MQDXP_VERSION_1
Numero di versione per la struttura del parametro di uscita conversione dati.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQDXP_CURRENT_
Versione corrente della struttura del parametro di uscita conversione dati.

Nota: Quando viene introdotta una nuova versione di questa struttura, il layout della parte esistente non viene modificato. L'uscita deve quindi verificare che il campo di *Version* sia uguale o superiore alla versione più bassa che contiene i campi che l'uscita deve utilizzare.

Questo è un campo di immissione per l'uscita.

Dichiarazione C

```
typedef struct tagMQDXP MQDXP;
struct tagMQDXP {
  MQCHAR4  StrucId;         /* Structure identifier */
  MQLONG   Version;         /* Structure version number */
  MQLONG   ExitOptions;     /* Reserved */
  MQLONG   AppOptions;      /* Application options */
  MQLONG   Encoding;        /* Numeric encoding required by
                               application */
  MQLONG   CodedCharSetId;  /* Character set required by application */
  MQLONG   DataLength;      /* Length in bytes of message data */
  MQLONG   CompCode;        /* Completion code */
  MQLONG   Reason;          /* Reason code qualifying CompCode */
  MQLONG   ExitResponse;    /* Response from exit */
  MQHCONN  Hconn;           /* Connection handle */
  PMQIEP   pEntryPoints;    /* Address of the MQIEP structure */
};
```

Dichiarazione COBOL (solo IBM i)

```
**   MQDXP structure
  10 MQDXP.
**    Structure identifier
   15 MQDXP-STRUCID        PIC X(4).
**    Structure version number
   15 MQDXP-VERSION        PIC S9(9) BINARY.
**    Reserved
   15 MQDXP-EXITOPTIONS    PIC S9(9) BINARY.
**    Application options
   15 MQDXP-APPOPTIONS     PIC S9(9) BINARY.
**    Numeric encoding required by application
   15 MQDXP-ENCODING       PIC S9(9) BINARY.
**    Character set required by application
   15 MQDXP-CODEDCHARSETID PIC S9(9) BINARY.
**    Length in bytes of message data
   15 MQDXP-DATALENGTH     PIC S9(9) BINARY.
**    Completion code
   15 MQDXP-COMPCODE       PIC S9(9) BINARY.
**    Reason code qualifying COMPCODE
   15 MQDXP-REASON         PIC S9(9) BINARY.
**    Response from exit
   15 MQDXP-EXITRESPONSE   PIC S9(9) BINARY.
**    Connection handle
   15 MQDXP-HCONN          PIC S9(9) BINARY.
```

Dichiarazione assembler System/390

```
MQDXP                 DSECT
MQDXP_STRUCID         DS   CL4  Structure identifier
MQDXP_VERSION         DS   F    Structure version number
MQDXP_EXITOPTIONS     DS   F    Reserved
MQDXP_APPOPTIONS      DS   F    Application options
MQDXP_ENCODING        DS   F    Numeric encoding required by application
MQDXP_CODEDCHARSETID  DS   F    Character set required by application
MQDXP_DATALENGTH      DS   F    Length in bytes of message data
MQDXP_COMPCODE        DS   F    Completion code
MQDXP_REASON          DS   F    Reason code qualifying COMPCODE
MQDXP_EXITRESPONSE    DS   F    Response from exit
MQDXP_HCONN           DS   F    Connection handle
*
MQDXP_LENGTH          EQU  *-MQDXP
                      ORG  MQDXP
MQDXP_AREA            DS   CL(MQDXP_LENGTH)
```

MQXCNVC - Converti caratteri

La chiamata MQXCNVC converte i caratteri da una serie di caratteri ad un'altra utilizzando il linguaggio di programmazione C.

Questa chiamata fa parte di DCI (Data Conversion Interface) IBM MQ , che è una delle interfacce del framework IBM MQ .

Nota: la chiamata può essere utilizzata sia dall'applicazione che dall'ambiente di uscita di conversione dati.

Sintassi

MQXCNVC (*hconn*, *options*, *SourceCCSID*, *SourceLength*, *SourceBuffer*, *TargetCCSID*, *TargetLength*, *TargetBuffer*, *DataLength*, *CompCode*, *Reason*)

Parametri

Hconn

Tipo: MQHCONN - input

Questo handle rappresenta la connessione al gestore code.

In un'uscita di conversione dati, Hconn è in genere l'handle che viene passato all'uscita di conversione dati nel campo Hconn della struttura MQDXP; questo handle non è necessariamente uguale all'handle specificato dall'applicazione che ha emesso la chiamata MQGET.

IBM i In IBM i, è possibile specificare il seguente valore speciale per Hconn:

DEF_MQH_HCONN
Handle di connessione predefinito.

Se si esegue un'applicazione CICS TS 3.2 o superiore, assicurarsi che il programma di uscita conversione caratteri, che richiama la chiamata MQXCNVC, sia definito come OPENAPI. Questa definizione impedisce l'errore MQRC_HCONN_ERROR del 2018 causato da una connessione non corretta e consente il completamento di MQGET.

Opzioni

Tipo: MQLONG - input

Opzioni che controllano l'azione di MQXCNVC.

È possibile specificare zero o più delle opzioni descritte. Per specificare più di un'opzione, aggiungere i valori insieme (non aggiungere la stessa costante più di una volta) o combinare i valori utilizzando l'operazione OR bit per bit (se il linguaggio di programmazione supporta le operazioni bit).

Opzione di conversione predefinita: la seguente opzione controlla l'utilizzo della conversione dei caratteri predefinita:

MQDCC_DEFAULT_CONVERSION
Conversione predefinita.

Questa opzione specifica che è possibile utilizzare la conversione caratteri predefinita se una o entrambe le serie di caratteri specificate nella chiamata non sono supportate. Ciò consente al gestore code di utilizzare una serie di caratteri predefinita specificata dall'installazione che si avvicina alla serie di caratteri specificata, durante la conversione della stringa.

Nota: Il risultato dell'utilizzo di una serie di caratteri approssimativi per convertire la stringa è che alcuni caratteri possono essere convertiti in maniera non corretta. Ciò può essere evitato utilizzando nella stringa solo i caratteri comuni sia alla serie di caratteri specificata che alla serie di caratteri predefinita.

I set di caratteri predefiniti vengono definiti da un'opzione di configurazione quando il gestore code viene installato o riavviato.

Se MQDCC_DEFAULT_CONVERSION non è specificato, il gestore code utilizza solo le serie di caratteri specificate per convertire la stringa e la chiamata ha esito negativo se una o entrambe le serie di caratteri non sono supportate.

Questa opzione è supportata nei seguenti ambienti:

- 

AIX
- 

IBM i
- 

Linux
- 

Windows

Opzione di riempimento: la seguente opzione consente al gestore code di riempire la stringa convertita con spazi vuoti o eliminare caratteri finali non significativi, in modo che la stringa convertita si adatti al buffer di destinazione:

MQDCC_FILL_TARGET_BUFFER
Riempimento buffer di destinazione.

Questa opzione richiede che la conversione avvenga in modo tale che il buffer di destinazione venga riempito completamente:

- Se la stringa si contrae quando viene convertita, vengono aggiunti spazi finali per riempire il buffer di destinazione.
- Se la stringa si espande quando viene convertita, i caratteri finali non significativi vengono eliminati per rendere la stringa convertita adatta al buffer di destinazione. Se questa operazione può essere eseguita correttamente, la chiamata viene completata con MQCC_OK e il codice motivo MQRC_NONE.

  Se il numero di caratteri finali non significativi è troppo basso, la maggior parte della stringa viene inserita nel buffer di destinazione e la chiamata viene completata con MQCC_WARNING e codice motivo MQRC_CONVERTED_MSG_TOO_BIG.

  I caratteri non significativi sono:

  – Spazi vuoti finali
  – I caratteri che seguono il primo carattere null nella stringa (ma escludendo il primo carattere null stesso)
- Se la stringa, `TargetCCSID`e `TargetLength` sono tali che il buffer di destinazione non può essere impostato completamente con caratteri validi, la chiamata ha esito negativo con MQCC_FAILED e codice motivo MQRC_TARGET_LENGTH_ERROR. Ciò può verificarsi quando `TargetCCSID` è una serie di caratteri DBCS puri (come UTF-16), ma `TargetLength` specifica una lunghezza che è un numero dispari di byte.

- `TargetLength` può essere minore o maggiore di `SourceLength`. Al ritorno da MQXCNVC, `DataLength` ha lo stesso valore di `TargetLength`.

Se questa opzione non viene specificata:

- La stringa è consentita per contrarre o espandere all'interno del buffer di destinazione come richiesto. I caratteri finali non significativi non vengono aggiunti o eliminati.

  Se la stringa convertita si adatta al buffer di destinazione, la chiamata viene completata con MQCC_OK e il codice motivo MQRC_NONE.

  Se la stringa convertita è troppo grande per il buffer di destinazione, la maggior parte della stringa viene inserita nel buffer di destinazione e la chiamata viene completata con MQCC_WARNING e codice motivo MQRC_CONVERTED_MSG_TOO_BIG. Notare che in questo caso possono essere restituiti meno di `TargetLength` byte.
- `TargetLength` può essere minore o maggiore di `SourceLength`. Alla restituzione da MQXCNVC, `DataLength` è minore o uguale a `TargetLength`.

Questa opzione è supportata nei seguenti ambienti:

- 

AIX
- 

IBM i
- 

Linux
- 

Windows

Opzioni di codifica: le opzioni descritte possono essere utilizzate per specificare le codificazioni intere delle stringhe di origine e di destinazione. La codifica rilevante viene utilizzata solo quando l'identificativo della serie di caratteri corrispondente indica che la rappresentazione della serie di caratteri nella memoria principale dipende dalla codifica utilizzata per i numeri interi binari. Ciò riguarda solo alcune serie di caratteri multibyte (ad esempio, le serie di caratteri UTF-16).

La codifica viene ignorata se la serie di caratteri è una serie di caratteri a byte singolo (SBCS) o una serie di caratteri a più byte con rappresentazione nella memoria principale che non dipende dalla codifica del numero intero.

È necessario specificare solo uno dei valori MQDCC_SOURCE_ *, combinato con uno dei valori MQDCC_TARGET_ *:

MQDCC_SOURCE_ENC_NATIVE
La codifica di origine è quella predefinita per l'ambiente e il linguaggio di programmazione.

MQDCC_SOURCE_EN_NORMAL
La codifica origine è normale.

MQDCC_SOURCE_ENC_REVERSED
La codifica di origine è invertita.

MQDCC_SOURCE_ENC_UNDEFINED
La codifica di origine non è definita.

MQDCC_TARGET_ENC_NATIVE
La codifica di destinazione è quella predefinita per l'ambiente e il linguaggio di programmazione.

MQDCC_XX_ENCODE_CASE_ONE destinazione - NORMAL
La codifica di destinazione è normale.

MQDCC_TARGET_ENC_REVERSED
La codifica di destinazione è invertita.

MQDCC_TARGET_ENC_UNDEFINED
La codifica di destinazione non è definita.

I valori di codifica definiti precedentemente possono essere aggiunti direttamente al campo `Options` . Tuttavia, se la codifica di origine o di destinazione viene ottenuta dal campo `Encoding` in MQMD o in un'altra struttura, è necessario eseguire la seguente elaborazione:

1. La codifica di numeri interi deve essere estratta dal campo Encoding eliminando le codifiche a virgola mobile e decimale compresso; consultare “Analisi delle codifiche” a pagina 927 per dettagli su come eseguire questa operazione.
2. La codifica di numeri interi risultante dal passo 1 deve essere moltiplicata per il fattore appropriato prima di essere aggiunta al campo Options . Questi fattori sono:
   - MQDCC_SOURCE_ENC_FACTOR per la codifica di origine
   - MQDCC_TARGET_ENC_FACTOR per la codifica di destinazione

Il seguente codice di esempio illustra come codificarlo nel linguaggio di programmazione C:

```
Options = (MsgDesc.Encoding & MQENC_INTEGER_MASK)
           * MQDCC_SOURCE_ENC_FACTOR
        + (DataConvExitParms.Encoding & MQENC_INTEGER_MASK)
           * MQDCC_TARGET_ENC_FACTOR;
```

Se non specificato, le opzioni di codifica vengono impostate per impostazione predefinita su non definito (MQDCC_ * _ENC_UNDEFINED). Nella maggior parte dei casi, ciò non influisce sul corretto completamento della chiamata MQXCNVC. Tuttavia, se la serie di caratteri corrispondente è una serie di caratteri multibyte con rappresentazione dipendente dalla codifica (ad esempio, una serie di caratteri UTF-16), la chiamata ha esito negativo con codice motivo MQRC_SOURCE_INTEGER_ENC_ERROR o MQRC_TARGET_INTEGER_ENC_ERROR come appropriato.

Le opzioni di codifica sono supportate nei seguenti ambienti:

- AIX
- IBM i
- Linux
- Windows
- z/OS

Opzione predefinita: se nessuna delle opzioni descritte precedentemente viene specificata, è possibile utilizzare la seguente opzione:

MQDCC_NONE
Nessuna opzione specificata.

MQDCC_NONE è definito per la documentazione del programma. Non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

SourceCCSID
Tipo: MQLONG - input

Questo è il CCSID (coded character set identifier) della stringa di immissione in SourceBuffer.

SourceLength
Tipo: MQLONG - input

Questa è la lunghezza in byte della stringa di input in SourceBuffer ; deve essere zero o maggiore.

SourceBuffer
Tipo: MQCHAR x SourceLength - input

Questo è il buffer contenente la stringa da convertire da una serie di caratteri ad un'altra.

TargetCCSID
Tipo: MQLONG - input

Questo è il CCSID (coded character set identifier) della serie di caratteri in cui `SourceBuffer` deve essere convertito.

TargetLength
Tipo: MQLONG - input

È la lunghezza in byte del buffer di output `TargetBuffer` ; deve essere zero o maggiore. Può essere minore o maggiore di `SourceLength`.

TargetBuffer
Tipo: MQCHAR x TargetLength - output

Questa è la stringa dopo che è stata convertita nella serie di caratteri definita da `TargetCCSID`. La stringa convertita può essere più corta o più lunga della stringa non convertita. Il parametro **`DataLength`** indica il numero di byte validi restituiti.

DataLength
Tipo: MQLONG - output

Questa è la lunghezza della stringa restituita nel buffer di output `TargetBuffer`. La stringa convertita può essere più corta o più lunga della stringa non convertita.

CompCode
Tipo: MQLONG - output

Il valore è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice di errore `CompCode`.

Se `CompCode` è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se `CompCode` è MQCC_WARNING:

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848 ') Dati convertiti troppo grandi per il buffer.

Se `CompCode` è MQCC_FAILED:

ERRORE MQRC_DATA_LENGTH
(2010, X'7DA') Parametro di lunghezza dati non valido.

ERRORE MQRC_DBCS
(2150, X'866 ') Stringa DBCS non valida.

ERRORE MQRC_HCONN
(2018, X'7E2') Handle di connessione non valido.

ERRORE MQRC_OPTIONS_
(2046, X'7FE') Opzioni non valide o non congruenti.

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Risorse di sistema insufficienti.

ERRORE_ORIGINE_RISORSE MQRC
(2145, X'861 ') Parametro del buffer di origine non valido.

ERRORE CCSID DI MQRC_SOURCE_
(2111, X'83F') Identificativo serie di caratteri codificati origine non valido.

ERRORE DI INIZIALIZZAZIONE MQRC_SOURCE_INTEGER_
(2112, X'840 ') Numero intero di origine non riconosciuto.

ERRORE_ORIGINE_RISORSE MQRC
(2143, X'85F') Parametro di lunghezza origine non valido.

MQRC_STORAGE_NON_DISPONIBILE
(2071, X'817 ') Memoria disponibile insufficiente.

ERRORE - BUFFER_MQRC_TARGET_
(2146, X'862 ') Parametro buffer di destinazione non valido.

ERRORE MQRC_TARGET_CCSID_
(2115, X'843 ') Identificativo serie di caratteri codificati di destinazione non valido.

ERRORE MQRC_TARGET_INTEGER_ENC
(2116, X'844 ') Codifica numero intero di destinazione non riconosciuta.

ERRORE MQRC_TARGET_LENGTH
(2144, X'860 ') Parametro di lunghezza di destinazione non valido.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, vedere Messaggi e codici di errore.

Richiamo C

```
MQXCNVC (Hconn, Options, SourceCCSID, SourceLength, SourceBuffer,
        TargetCCSID, TargetLength, TargetBuffer, &DataLength,
        &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;            /* Connection handle */
MQLONG   Options;          /* Options that control the action of
                              MQXCNVC */
MQLONG   SourceCCSID;      /* Coded character set identifier of string
                              before conversion */
MQLONG   SourceLength;     /* Length of string before conversion */
MQCHAR   SourceBuffer[n];  /* String to be converted */
MQLONG   TargetCCSID;      /* Coded character set identifier of string
                              after conversion */
MQLONG   TargetLength;     /* Length of output buffer */
MQCHAR   TargetBuffer[n];  /* String after conversion */
MQLONG   DataLength;       /* Length of output string */
MQLONG   CompCode;         /* Completion code */
MQLONG   Reason;           /* Reason code qualifying CompCode */
```

Dichiarazione COBOL (solo IBM i)

IBM i

```
     CALL 'MQXCNVC' USING HCONN, OPTIONS, SOURCECCSID, SOURCELENGTH,
                          SOURCEBUFFER, TARGETCCSID, TARGETLENGTH,
                          TARGETBUFFER, DATALENGTH, COMPCODE, REASON.
```

Dichiarare i parametri come segue:

```
**   Connection handle
 01  HCONN         PIC S9(9) BINARY.
**   Options that control the action of MQXCNVC
 01  OPTIONS       PIC S9(9) BINARY.
**   Coded character set identifier of string before conversion
 01  SOURCECCSID   PIC S9(9) BINARY.
```

```
**   Length of string before conversion
 01  SOURCELENGTH  PIC S9(9) BINARY.
**   String to be converted
 01  SOURCEBUFFER  PIC X(n).
**   Coded character set identifier of string after conversion
 01  TARGETCCSID   PIC S9(9) BINARY.
**   Length of output buffer
 01  TARGETLENGTH  PIC S9(9) BINARY.
**   String after conversion
 01  TARGETBUFFER  PIC X(n).
**   Length of output string
 01  DATALENGTH    PIC S9(9) BINARY.
**   Completion code
 01  COMPCODE      PIC S9(9) BINARY.
**   Reason code qualifying COMPCODE
 01  REASON        PIC S9(9) BINARY.
```

Dichiarazione assembler S/390

```
        CALL MQXCNVC,(HCONN,OPTIONS,SOURCECCSID,SOURCELENGTH,         X
              SOURCEBUFFER,TARGETCCSID,TARGETLENGTH,TARGETBUFFER,     X
              DATALENGTH,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN         DS  F      Connection handle
OPTIONS       DS  F      Options that control the action of MQXCNVC
SOURCECCSID   DS  F      Coded character set identifier of string before
*                        conversion
SOURCELENGTH  DS  F      Length of string before conversion
SOURCEBUFFER  DS  CL(n)  String to be converted
TARGETCCSID   DS  F      Coded character set identifier of string after
*                        conversion
TARGETLENGTH  DS  F      Length of output buffer
TARGETBUFFER  DS  CL(n)  String after conversion
DATALENGTH    DS  F      Length of output string
COMPCODE      DS  F      Completion code
REASON        DS  F      Reason code qualifying COMPCODE
```

MQ_DATA_CONV_EXIT - Uscita conversione dati

La chiamata MQ_DATA_CONV_EXIT descrive i parametri passati all'uscita di conversione dati.

Nessun punto di entrata denominato MQ_DATA_CONV_EXIT viene fornito dal gestore code (consultare la nota sull'uso 11).

Questa definizione fa parte di DCI (Data Conversion Interface) IBM MQ , che è una delle interfacce del framework IBM MQ .

Sintassi

MQ_DATA_CONV_EXIT (*DataConvExitParms*, *MsgDesc*, *InBufferLength*, *InBuffer*, *OutBufferLength*, *OutBuffer*)

Parametri

DataConvExitParms
Tipo: MQDXP - input/output

Questa struttura contiene informazioni relative al richiamo dell'exit. L'uscita imposta le informazioni in questa struttura per indicare il risultato della conversione. Consultare “MQDXP - Parametro di uscita conversione dati” a pagina 939 per i dettagli dei campi in questa struttura.

MsgDesc
Tipo: MQMD - input/output

Nell'input all'uscita, questo è il descrittore del messaggio associato ai dati del messaggio passati all'uscita nel parametro **InBuffer** .

Nota: Il parametro **MsgDesc** inoltrato all'exit è sempre la versione più recente di MQMD supportata dal gestore code che richiama l'exit. Se l'uscita è destinata ad essere trasportabile tra ambienti differenti, l'uscita controllerà il campo `Version` in `MsgDesc` per verificare che i campi a cui l'uscita deve accedere siano presenti nella struttura.

Nei seguenti ambienti, all'uscita viene passato un MQMD version-2 :

- AIX
- IBM i
- Linux
- Windows

In tutti gli altri ambienti che supportano l'uscita di conversione dati, all'uscita viene passato un MQMD version-1 .

In fase di output, l'uscita modificherà i campi `Encoding` e `CodedCharSetId` con i valori richiesti dall'applicazione, se la conversione ha avuto esito positivo; queste modifiche si rifletteranno nuovamente sull'applicazione. Tutte le altre modifiche apportate dall'exit alla struttura vengono ignorate; non vengono riportate nell'applicazione.

Se l'uscita restituisce MQXDR_OK nel campo `ExitResponse` della struttura MQDXP, ma non modifica i campi `Encoding` o `CodedCharSetId` nel descrittore del messaggio, il gestore code restituisce per tali campi i valori che i campi corrispondenti nella struttura MQDXP avevano nell'input dell'uscita.

Lunghezza InBuffer
Tipo: MQLONG - input

Lunghezza in byte di `InBuffer`.

Si tratta della lunghezza del buffer di input `InBuffere` specifica il numero di byte che devono essere elaborati dall'uscita. `InBufferLength` è la lunghezza minore dei dati del messaggio prima della conversione e la lunghezza del buffer fornito dall'applicazione sulla chiamata MQGET.

Il valore è sempre maggiore di zero.

InBuffer
Tipo: MQBYTExInBufferLength - input

Buffer contenente il messaggio non convertito.

Contiene i dati del messaggio prima della conversione. Se l'uscita non è in grado di convertire i dati, il gestore code restituisce il contenuto di questo buffer all'applicazione dopo che l'uscita è stata completata.

Nota: L'uscita non deve modificare `InBuffer` ; se questo parametro viene modificato, i risultati non sono definiti.

Nel linguaggio di programmazione C, questo parametro è definito come un puntatore a vuoto.

Lunghezza OutBuffer
Tipo: MQLONG - input

Lunghezza in byte di `OutBuffer`.

Questa è la lunghezza del buffer di output `OutBuffered` è uguale alla lunghezza del buffer fornito dall'applicazione sulla chiamata MQGET.

Il valore è sempre maggiore di zero.

OutBuffer
Tipo: MQBYTExOutBufferLength - output

Buffer contenente il messaggio convertito.

All'output dall'uscita, se la conversione ha avuto esito positivo (come indicato dal valore MQXDR_OK nel campo `ExitResponse` del parametro **DataConvExitParms**), `OutBuffer` contiene i dati del messaggio da consegnare all'applicazione, nella rappresentazione richiesta. Se la conversione ha avuto esito negativo, tutte le modifiche che l'uscita ha apportato a questo buffer vengono ignorate.

Nel linguaggio di programmazione C, questo parametro è definito come un puntatore a vuoto.

Note d'utilizzo

1. Un'uscita di conversione dati è un'uscita scritta dall'utente che riceve il controllo durante l'elaborazione di una chiamata MQGET. La funzione eseguita dall'uscita di conversione dati è definita dal provider dell'uscita; tuttavia, l'uscita deve essere conforme alle regole qui descritte e nella struttura di parametro associata MQDXP.

   I linguaggi di programmazione che possono essere utilizzati per un'uscita di conversione dati sono determinati dall'ambiente.
2. L'uscita viene richiamata solo se tutte le seguenti istruzioni sono vere:
   - L'opzione MQGMO_CONVERT è specificata nella chiamata MQGET
   - Il campo `Format` nel descrittore del messaggio non è MQFMT_NONE
   - Il messaggio non è già nella rappresentazione richiesta; ovvero, uno o entrambi i `CodedCharSetId` e `Encoding` del messaggio sono diversi dal valore specificato dall'applicazione nel descrittore del messaggio fornito nella chiamata MQGET
   - Il gestore code non ha ancora eseguito correttamente la conversione
   - La lunghezza del buffer dell'applicazione è maggiore di zero
   - La lunghezza dei dati del messaggio è maggiore di zero
   - Il codice di errore fino ad ora durante l'operazione MQGET è MQRC_NONE o MQRC_TRUNCATED_MSG_ACCEPTED
3. Quando si sta scrivendo un'uscita, considerare la possibilità di codificare l'uscita in un modo che gli consenta di convertire i messaggi che sono stati troncati. I messaggi troncati possono verificarsi nei modi seguenti:
   - L'applicazione ricevente fornisce un buffer più piccolo del messaggio, ma specifica l'opzione MQGMO_ACCEPT_TRUNCATED_MSG sulla chiamata MQGET.

     In tal caso, il campo `Reason` nel parametro **DataConvExitParms** sull'input per l'exit ha il valore MQRC_TRUNCATED_MSG_ACCEPTED.
   - Il mittente del messaggio lo ha troncato prima di inviarlo. Ciò può verificarsi con i messaggi di report, ad esempio (consultare “Conversione dei messaggi di report” a pagina 938 per ulteriori dettagli).

     In questo caso, il campo `Reason` nel parametro **DataConvExitParms** sull'input per l'uscita ha il valore MQRC_NONE (se l'applicazione ricevente ha fornito un buffer sufficientemente grande per il messaggio).

   Pertanto, il valore del campo `Reason` sull'input per l'uscita non può essere sempre utilizzato per decidere se il messaggio è stato troncato.

   La caratteristica distintiva di un messaggio troncato è che la lunghezza fornita all'uscita nel parametro **InBufferLength** è inferiore alla lunghezza implicita dal nome formato contenuto nel campo `Format` nel descrittore del messaggio. L'uscita deve quindi controllare il valore di `InBufferLength` prima di tentare la conversione di uno qualsiasi dei dati; l'uscita non deve presumere che sia stata fornita l'intera quantità di dati impliciti nel nome del formato.

   Se l'uscita non è stata scritta per convertire i messaggi troncati e `InBufferLength` è inferiore al valore previsto, l'uscita restituirà MQXDR_CONVERSION_FAILED nel campo `ExitResponse` del parametro **DataConvExitParms** , con i campi `CompCode` e `Reason` impostati su MQCC_WARNING e MQRC_FORMAT_ERROR.

Se l'uscita è stata scritta per convertire i messaggi troncati, l'uscita convertirà la maggior parte dei dati possibile (consultare la nota di utilizzo successiva), facendo attenzione a non tentare di esaminare o convertire i dati oltre la fine di `InBuffer`. Se la conversione viene completata correttamente, l'uscita lascerà invariato il campo `Reason` nel parametro **DataConvExitParms** . Restituisce MQRC_TRUNCATED_MSG_ACCEPTED se il messaggio è stato troncato dal gestore code del destinatario e MQRC_NONE se il messaggio è stato troncato dal mittente del messaggio.

È anche possibile che un messaggio si espanda durante la conversione, al punto in cui è più grande di `OutBuffer`. In questo caso l'uscita deve decidere se troncare il messaggio; il campo `AppOptions` nel parametri **DataConvExitParms** indica se l'applicazione ricevente ha specificato l'opzione MQGMO_ACCEPT_TRUNCATED_MSG.

4. In genere, tutti i dati nel messaggio forniti all'uscita in `InBuffer` vengono convertiti o non lo sono. Un'eccezione a questo, tuttavia, si verifica se il messaggio viene troncato, prima della conversione o durante la conversione; in questo caso può esserci un elemento incompleto alla fine del buffer (ad esempio: 1 byte di un carattere a doppio byte o 3 byte di un numero intero a 4 byte). In questa situazione, considerare la possibilità di omettere l'elemento incompleto e impostare i byte non utilizzati in `OutBuffer` su valori null. Tuttavia, gli elementi o i caratteri completi all'interno di una schiera o stringa devono essere convertiti.
5. Quando un'uscita è necessaria per la prima volta, il gestore code tenta di caricare un oggetto con lo stesso nome del formato (a parte le estensioni). L'oggetto caricato deve contenere l'uscita che elabora i messaggi con tale nome formato. Considerare la possibilità di rendere identici il nome dell'uscita e il nome dell'oggetto che contiene l'uscita, anche se non tutti gli ambienti lo richiedono.
6. Una nuova copia dell'uscita viene caricata quando un'applicazione tenta di recuperare il primo messaggio che utilizza `Format` da quando l'applicazione si è connessa al gestore code. Per le applicazioni CICS o IMS , ciò significa quando il sottosistema CICS o IMS si connette al gestore code. Una nuova copia può essere caricata anche in altri momenti, se il gestore code ha eliminato una copia precedentemente caricata. Per questo motivo, un'uscita non deve tentare di utilizzare la memoria statica per comunicare le informazioni da un richiamo dell'uscita al successivo - l'uscita può essere scaricata tra i due richiami.
7. Se è presente un'uscita fornita dall'utente con lo stesso nome di uno dei formati integrati supportati dal gestore code, l'uscita fornita dall'utente non sostituisce la routine di conversione integrata. Le uniche circostanze in cui tale uscita viene richiamata sono:
   - Se la routine di conversione integrata non è in grado di gestire le conversioni verso o da `CodedCharSetId` o `Encoding` coinvolti, oppure
   - Se la routine di conversione incorporata non è riuscita a convertire i dati (ad esempio, perché esiste un campo o un carattere che non può essere convertito).
8. L'ambito dell'uscita dipende dall'ambiente. I nomi `Format` devono essere scelti per ridurre il rischio di conflitti con altri formati. Si consiglia di iniziare con i caratteri che identificano l'applicazione che definisce il nome del formato.
9. L'uscita di conversione dati viene eseguita in un ambiente simile a quello del programma che ha emesso la chiamata MQGET; l'ambiente include lo spazio di indirizzo e il profilo utente (dove applicabile). Il programma potrebbe essere un agente del canale dei messaggi che invia messaggi a un gestore code di destinazione che non supporta la conversione dei messaggi. L'uscita non può compromettere l'integrità del gestore code, poiché non viene eseguita nell'ambiente del gestore code.
10. L'unica chiamata MQI che può essere utilizzata dall'uscita è MQXCNVC; il tentativo di utilizzare altre chiamate MQI ha esito negativo con codice motivo MQRC_CALL_IN_PROGRESS o altri errori imprevedibili.
11. Nessun punto di immissione denominato MQ_DATA_CONV_EXIT è fornito dal gestore code. Tuttavia, viene fornito un `typedef` per il nome MQ_DATA_CONV_EXIT nel linguaggio di programmazione C, e questo può essere utilizzato per dichiarare l'uscita scritta dall'utente, per garantire che i parametri siano corretti. Il nome dell'uscita deve essere uguale al nome del formato (il nome contenuto nel campo `Format` in MQMD), anche se non è richiesto in tutti gli ambienti.

Il seguente esempio illustra come l'uscita che elabora il formato MYFORMAT può essere dichiarata nel linguaggio di programmazione C:

```
#include "cmqc.h"
#include "cmqxc.h"

MQ_DATA_CONV_EXIT MYFORMAT;

void MQENTRY MYFORMAT(
     PMQDXP   pDataConvExitParms, /* Data-conversion exit parameter
                                     block */
     PMQMD    pMsgDesc,           /* Message descriptor */
     MQLONG   InBufferLength,     /* Length in bytes of InBuffer */
     PMQVOID  pInBuffer,          /* Buffer containing the unconverted
                                     message */
     MQLONG   OutBufferLength,    /* Length in bytes of OutBuffer */
     PMQVOID  pOutBuffer)         /* Buffer containing the converted
                                     message */
{
  /* C language statements to convert message */
}
```

12. z/OS Su z/OS, se è attiva anche un'uscita di attraversamento API, viene richiamata dopo l'uscita di conversione dati.

Richiamo C

```
exitname (&DataConvExitParms, &MsgDesc, InBufferLength,
        InBuffer, OutBufferLength, OutBuffer);
```

I parametri passati all'uscita vengono dichiarati come segue:

```
MQDXP   DataConvExitParms;  /* Data-conversion exit parameter block */
MQMD    MsgDesc;            /* Message descriptor */
MQLONG  InBufferLength;     /* Length in bytes of InBuffer */
MQBYTE  InBuffer[n];        /* Buffer containing the unconverted
                               message */
MQLONG  OutBufferLength;    /* Length in bytes of OutBuffer */
MQBYTE  OutBuffer[n];       /* Buffer containing the converted
                               message */
```

Dichiarazione COBOL (soloIBM i)

IBM i

```
     CALL 'exitname' USING DATACONVEXITPARMS, MSGDESC, INBUFFERLENGTH,
                           INBUFFER, OUTBUFFERLENGTH, OUTBUFFER.
```

I parametri passati all'uscita vengono dichiarati come segue:

```
**   Data-conversion exit parameter block
 01  DATACONVEXITPARMS.
     COPY CMQDXPV.
**   Message descriptor
 01  MSGDESC.
     COPY CMQMDV.
**   Length in bytes of INBUFFER
 01  INBUFFERLENGTH     PIC S9(9) BINARY.
**   Buffer containing the unconverted message
 01  INBUFFER           PIC X(n).
**   Length in bytes of OUTBUFFER
 01  OUTBUFFERLENGTH    PIC S9(9) BINARY.
**   Buffer containing the converted message
 01  OUTBUFFER          PIC X(n).
```

Dichiarazione assembler System/390

```
         CALL EXITNAME,(DATACONVEXITPARMS,MSGDESC,INBUFFERLENGTH,      X
               INBUFFER,OUTBUFFERLENGTH,OUTBUFFER)
```

I parametri passati all'uscita vengono dichiarati come segue:

```
DATACONVEXITPARMS  CMQDXPA  ,      Data-conversion exit parameter block
MSGDESC            CMQMDA   ,      Message descriptor
INBUFFERLENGTH     DS       F      Length in bytes of INBUFFER
INBUFFER           DS       CL(n)  Buffer containing the unconverted
*                                  message
OUTBUFFERLENGTH    DS       F      Length in bytes of OUTBUFFER
OUTBUFFER          DS       CL(n)  Buffer containing the converted
*                                  message
```

Proprietà specificate come elementi MQRFH2

Le proprietà del descrittore non di messaggi possono essere specificate come elementi nelle cartelle di intestazione MQRFH2 . Panoramica degli elementi MQRFH2 specificati come proprietà.

Ciò conserva la compatibilità con le versioni precedenti dei client IBM MQ JMS e XMS . Questa sezione descrive come specificare le proprietà nelle intestazioni MQRFH2 .

Per utilizzare gli elementi di MQRFH2 come proprietà, specificare gli elementi come descritto in Utilizzo IBM MQ classes for Java . Queste informazioni integrano le informazioni descritte in “MQRFH2 - Regole e intestazione di formattazione 2” a pagina 542.

Associazione dei tipi di dati delle propriet ... ai tipi di dati MQRFH2

Questo argomento fornisce informazioni sui tipi di proprietà dei messaggi associati ai tipi di dati MQRFH2 corrispondenti.

Tabella 635. Tipi di dati MQRFH2 supportati

Tipo di proprietà del messaggio	**Tipo di dati MQRFH2**
MQBYTE []	bin.hex
MQBOOL	booleano
MQINT8	i1
MQINT16	i2
MQINT32	i4
MQINT64	i8
MQFLOAT32	r4
MQFLOAT64	r8
MQCHAR []	stringa

Si presuppone che qualsiasi elemento senza un tipo di dati sia di tipo "string".

Un tipo di dati MQRFH2 `int`, che indica un numero intero di dimensione non specificata, viene considerato come se fosse un `i8`.

Un valore nullo è indicato dall'attributo dell'elemento `xsi:nil='true'`. Non utilizzare l'attributo `xsi:nil='false'` per valori non null.

Ad esempio, la seguente proprietà ha un valore null:

```
<NullProperty xsi:nil='true'></NullProperty>
```

Una proprietà byte o stringa di caratteri può avere un valore vuoto. Viene rappresentato da un elemento MQRFH2 con un valore di elemento di lunghezza zero.

Ad esempio, la seguente proprietà ha un valore vuoto:

```
<EmptyProperty></EmptyProperty>
```

Cartelle MQRFH2 supportate

Panoramica sull'utilizzo dei campi descrittore del messaggio come proprietà.

Le cartelle <jms>, <mcd>, <mqext>e <usr> sono descritte in L'intestazione MQRFH2 e JMS. La cartella <usr> viene utilizzata per trasportare tutte le proprietà JMS definite dall'applicazione associate a un messaggio. I gruppi non sono consentiti nella cartella <usr> .

L'intestazione MQRFH2 e JMS supportano le seguenti cartelle aggiuntive:

- <mq>

  Questa cartella è utilizzata e riservata per le proprietà definite da MQutilizzate da IBM MQ.
- <mq_usr>

  Questa cartella può essere utilizzata per trasportare tutte le proprietà definite dall'applicazione che non sono esposte come JMS proprietà definite dall'utente, poiché le proprietà potrebbero non soddisfare i requisiti di una proprietà JMS . Questa cartella può contenere gruppi che la cartella <usr> non può contenere.
- Qualsiasi cartella contrassegnata con l'attributo content='properties' .

  Tale cartella è equivalente alla cartella <mq_usr> nel contenuto.
- <mqps>

  Questa cartella viene utilizzata per proprietà di pubblicazione / sottoscrizione di IBM MQ .

IBM MQ supporta anche le seguenti cartelle già utilizzate da WAS/SIB:

- <sib>

  Questa cartella viene utilizzata e riservata per le proprietà dei messaggi di sistema WAS/SIB che non sono esposte come proprietà JMS o sono associate alle proprietà JMS_IBM_ *, ma sono esposte alle applicazioni WAS/SIB; queste includono le proprietà dei percorsi di instradamento inverso e di inoltro.

  Almeno alcuni non possono essere esposti come proprietà JMS , perché sono array di byte. Se l'applicazione aggiunge proprietà a questa cartella, il valore viene ignorato o rimosso.
- <sib_usr>

  Questa cartella viene utilizzata e riservata per le proprietà del messaggio utente WAS/SIB che non possono essere esposte come proprietà utente JMS perché non sono di tipi supportati; sono esposte alle applicazioni WAS/SIB.

  Si tratta di proprietà utente, che è possibile ottenere o impostare tramite l'interfaccia SIMessage, ma il contenuto dell'array di byte è associato al valore della proprietà richiesto.

  Se l'applicazione IBM MQ scrive un elemento bin.hex arbitrario nella cartella, l'applicazione probabilmente riceve un IOException, poiché non è del formato previsto per il ripristino. Se si aggiunge un elemento diverso da un elemento bin.hex , si riceve un ClassCastException.

  Non tentare di rendere le proprietà disponibili per WAS/SIB utilizzando questa cartella; utilizzare invece la cartella <usr> per tale scopo.
- <sib_context>

Questa cartella viene utilizzata per le proprietà dei messaggi di sistema WAS/SIB non esposte alle applicazioni utente WAS/SIB o come proprietà JMS . Tali proprietà includono la sicurezza e le proprietà transazionali utilizzate per i servizi Web e simili.

L'applicazione non deve aggiungere proprietà a questa cartella.

- <mqema>

  Questa cartella è stata utilizzata da WAS/SIB invece della cartella <mqext> .

I nomi delle cartelle MQRFH2 sono sensibili al maiuscolo / minuscolo.

Le seguenti cartelle sono riservate, in qualsiasi combinazione di caratteri minuscoli o maiuscoli:

- Qualsiasi cartella con prefisso mq o wmq ; riservato per l'utilizzo da parte di IBM MQ.
- Qualsiasi cartella con prefisso sib ; riservato per essere utilizzato da WAS/SIB.
- Cartelle <Root> e <Body> ; riservate ma non utilizzate.

Le seguenti cartelle non sono riconosciute come contenenti le proprietà del messaggio:

- <psc>

  Utilizzato da IBM Integration Bus per trasmettere messaggi di comando di pubblicazione / sottoscrizione al broker.

- <pscr>

  Utilizzato da IBM Integration Bus per contenere le informazioni dal broker, in risposta ai messaggi di comando di pubblicazione / sottoscrizione.

- Qualsiasi cartella non definita da IBM, non contrassegnata con l'attributo content='properties' .

Non specificare content='properties' nelle cartelle <psc> o <pscr> . In questo caso, queste cartelle vengono considerate come proprietà e è probabile che IBM Integration Bus smetta di funzionare come previsto.

Se l'applicazione sta creando messaggi con proprietà, nelle intestazioni MQRFH2 per essere riconosciuta come un'intestazione MQRFH2 contenente proprietà, l'intestazione deve trovarsi nell'elenco di intestazioni che possono essere concatenate all'inizio del messaggio.

MQRFH2 può essere preceduto da un numero qualsiasi di intestazioni standard MQH o da un MQCIH, un MQDLH, un MQIIH, un MQTM, un MQTMC2o un MQXQH. Una stringa o un MQCFH termina l'analisi perché non possono essere concatenati.

È possibile che un messaggio contenga più intestazioni MQRFH2 contenenti tutte le proprietà del messaggio. Le cartelle con lo stesso nome possono coesistere in intestazioni differenti a meno che non sia diversamente limitato, ad esempio da WAS/SIB. Le cartelle vengono considerate come una cartella logica, se tutte si trovano in intestazioni significative.

Mentre le cartelle dalle intestazioni significative non possono essere unite con quelle cartelle in intestazioni non significative, le cartelle con lo stesso nome all'interno delle intestazioni significative possono essere unite, rimuovendo tutte le proprietà in conflitto. Le applicazioni non devono dipendere dal layout delle proprietà all'interno del messaggio.

I gruppi MQRFH2 vengono analizzati per le proprietà nelle cartelle definite dall'utente, ossia non nelle cartelle <wmq>, <jms>, <mcd>, <usr>, <mqext>, <sib>, <sib_usr>, <sib_context>e <mqema> .

I gruppi nelle cartelle delle proprietà definite da IBM, ad eccezione delle cartelle <wmq> e <mq> , vengono analizzati per le proprietà.

Una cartella MQRFH2 non può contenere contenuto misto; una cartella o un gruppo può contenere gruppi o proprietà o un valore, ma non entrambi.

Un segmento di un messaggio, il primo o un segmento successivo, non può contenere proprietà definite da IBM MQ diverse da quelle nel descrittore del messaggio. Pertanto, l'inserimento di un messaggio contenente tali proprietà con l'impostazione MQMF_SEGMENT o MQMF_SEGMENTATION_ALLOWED causa un errore di inserimento con MQRC_SEGMENTATION_NOT_ALLOWED.

Tuttavia, i gruppi di messaggi possono contenere proprietà definite da IBM MQ .

Generazione di intestazioni di MQRFH2

Se IBM MQ converte le proprietà del messaggio nella loro rappresentazione MQRFH2 , deve aggiungere MQRFH2 al messaggio. Aggiunge MQRFH2 come intestazione separata o lo unisce con un'intestazione esistente.

La creazione di nuove intestazioni MQRFH2 da parte di IBM MQ potrebbe interrompere le intestazioni esistenti in un messaggio. Le applicazioni che analizzano un buffer di messaggi per le intestazioni devono essere consapevoli che il numero e la posizione delle intestazioni in un buffer potrebbero cambiare in alcune circostanze. IBM MQ tenta di ridurre l'impatto dell'aggiunta di proprietà a un messaggio unendo le proprietà del messaggio in un'intestazione MQRFH2 esistente, dove è possibile. Inoltre, tenta di ridurre l'impatto inserendo un MQRFH2 generato in una posizione fissa rispetto alle altre intestazioni nel buffer di messaggi.

Un'intestazione MQRFH2 generata viene posizionata dopo MQMDe qualsiasi numero di intestazioni MQXQH, MQRFHe MQDLH , indipendentemente dall'ordine in cui si trovano. L'intestazione MQRFH2 generata viene collocata immediatamente prima della prima intestazione che non è un'intestazione MQMD, MQXQH, MQDLHo MQRFH .

z/OS Sui sistemi z/OS , l'intestazione MQRFH2 generata viene creata nel CCSIDdell'applicazione. Ciò è definito come segue:

- Per le applicazioni LE batch che utilizzano l'interfaccia DLL, CCSID è il CODESET associato alla locale corrente nel momento in cui viene emesso **MQCONN** (il valore predefinito è 1047).
- Per le applicazioni LE batch associate a uno degli stub MQ batch, CCSID è il CODESET associato alla locale corrente al momento della prima chiamata MQI emessa dopo **MQCONN** (il valore predefinito è 1047).
- Per le applicazioni batch non LE in esecuzione su un thread z/OS UNIX System Services (z/OS UNIX), CCSID è il valore di THLICCSID al momento della prima chiamata MQI emessa dopo **MQCONN** (il valore predefinito è 1047).
- Per altre applicazioni batch, CCSID è il CCSID del gestore code.

Per le applicazioni LE, la locale può essere modificata utilizzando il servizio richiamabile `setlocale()` / CEESETL LE . Per le applicazioni non LE in esecuzione su thread z/OS UNIX , il valore di THLICCSID può essere modificato utilizzando la z/OS UNIX macro di associazione **BPXYTHLI**.

Regole per l'unione generate MQRFH2

Le seguenti regole si applicano all'unione di un MQRFH2 generato con un MQRFH2esistente. L'intestazione MQRFH2 generata viene unita a un'intestazione MQRFH2 esistente, se:

1. Il MQRFH2 esistente si trova nella stessa posizione in cui IBM MQ posizionerebbe un MQRFH2generato o precedente nella catena di intestazioni.
2. Il CCSID delle proprietà generate è uguale al NameValueCCSID del MQRFH2esistente.

Altrimenti, l'intestazione generata viene posizionata separatamente nel buffer, nella posizione descritta in precedenza.

Regole per l'unione di cartelle in un MQRFH2 esistente

Se le proprietà del messaggio sono unite in un MQRFH2esistente, il MQRFH2 esistente viene sottoposto a scansione per le cartelle che corrispondono alle proprietà del messaggio e le unisce. Se una cartella corrispondente non esiste, viene aggiunta una nuova cartella alla fine delle cartelle esistenti. Se esiste una cartella corrispondente, viene eseguita la ricerca nella cartella. Tutte le proprietà corrispondenti vengono sovrascritte. Tutti i nuovi vengono aggiunti alla fine della cartella.

Limitazioni della cartella MQRFH2

Panoramica delle limitazioni delle cartelle nelle intestazioni MQRFH2

Le limitazioni MQRFH2 vengono applicate alle cartelle seguenti:

- I nomi elemento nella cartella <usr> non devono iniziare con il prefisso JMS ; tali nomi di proprietà sono riservati per l'utilizzo da parte di JMS e non sono validi per le proprietà definite dall'utente.

  Tale nome elemento non causa l'errore di analisi di MQRFH2 , ma non è accessibile alle API della proprietà del messaggio IBM MQ .
- I nomi elemento nella cartella <usr> non devono essere, in una combinazione di caratteri minuscoli o maiuscoli, NULL, TRUE, FALSE, NOT E, O, BETWEEN, LIKE, IN, IS e ESCAPE. Questi nomi corrispondono alle parole chiave SQL e rendono più difficile l'analisi dei selettori, poiché <usr> è la cartella predefinita utilizzata quando non viene specificata alcuna cartella per una particolare proprietà in un selettore.

  Tale nome elemento non causa l'errore di analisi di MQRFH2 , ma non è accessibile alle API della proprietà del messaggio IBM MQ .
- Il modello di contenuto della cartella <usr> è il seguente:
  - Qualsiasi nome XML valido può essere utilizzato come nome elemento, purché non contenga i due punti.
  - Sono consentiti solo elementi semplici, non cartelle nidificate.
  - Tutti gli elementi assumono il tipo predefinito di stringa, a meno che non siano modificati da un attributo dt="xxx" .
  - Tutti gli elementi sono facoltativi, ma non devono essere presenti più di una volta in una cartella.
- I nomi elemento in qualsiasi cartella considerata contenente le proprietà del messaggio non devono contenere un punto (.) (Carattere Unicode U+002E), perché viene utilizzato nei nomi delle proprietà per indicare la gerarchia.

  Tale nome elemento non causa l'errore di analisi di MQRFH2 , ma non è accessibile alle API della proprietà del messaggio IBM MQ .

In generale, le intestazioni MQRFH2 che contengono dati di stile XML validi possono essere analizzate da IBM MQ senza errori, anche se alcuni elementi di MQRFH2 non sono accessibili tramite API della proprietà del messaggio IBM MQ .

Conflitti di nomi elemento MQRFH2

Panoramica dei conflitti all'interno dei nomi elemento MQRFH2 .

Solo un valore può essere allegato a una proprietà del messaggio. Se un tentativo di accedere a una proprietà porta a un conflitto di valori, ne viene scelto uno di preferenza rispetto a un altro.

La sintassi IBM MQ per l'accesso agli elementi MQRFH2 consente l'identificazione univoca di un elemento, se una cartella non contiene elementi con lo stesso nome. Se una cartella contiene più di un elemento con lo stesso nome, il valore della proprietà utilizzata è quello più vicino all'intestazione del messaggio.

Ciò si applica se due o più cartelle con lo stesso nome sono contenute in intestazioni MQRFH2 significative differenti all'interno dello stesso messaggio.

Un conflitto può verificarsi quando la chiamata MQGET viene elaborata dopo che una proprietà del descrittore non di messaggi è stata impostata due volte: sia tramite una chiamata MQSETMP che direttamente nell'intestazione MQRFH2 non elaborata.

Se ciò si verifica, la proprietà associata al messaggio da una chiamata API assume la preferenza rispetto a una nei dati del messaggio, ovvero, quella nell'intestazione MQRFH2 non elaborata. Se si verifica un conflitto, si considera che si verifichi logicamente prima dei dati del messaggio.

Associazione dai nomi proprietà ai nomi elemento e cartella MQRFH2

Panoramica delle differenze tra nomi di proprietà e nomi di elementi nell'intestazione MQRFH2 .

Quando si utilizza una delle API definite che alla fine generano intestazioni MQRFH2 , per specificare le proprietà del messaggio (ad esempio, MQ JMS), il nome della proprietà non è necessariamente il nome dell'elemento nella cartella MQRFH2 .

Pertanto, una mappatura si verifica dal nome della proprietà all'elemento MQRFH2 e in modo inverso, tenendo conto sia del nome della cartella che contiene l'elemento che del nome dell'elemento. Alcuni esempi da IBM MQ classes for JMS sono già documentati in Utilizzo di IBM MQ classes for Java.

Tabella 636. Nomi di proprietà associati alla cartella MQRFH2 e nomi di elemento

Nome proprietà	Nome cartella MQRFH2	Nome elemento MQRFH2
`JMSDestination`	`jms`	`Dst`
`JMSType`	`mcd`	`Type`, `Set`, `Fmt`
xxx (definito dall'utente, dove xxx non inizia con JMS)	`usr`	`xxx`

Pertanto, quando un'applicazione JMS accede alla proprietà `JMSDestination` , questa viene associata all'elemento `Dst` nella cartella <jms> .

Quando si specificano le proprietà come elementi MQRFH2 , IBM MQ definisce i relativi elementi come segue:

Tabella 637. Nomi proprietà associati ai nomi elemento, gruppo e cartella MQRFH2

Nome proprietà	Nome cartella MQRFH2	Nome gruppo MQRFH2	Nome elemento MQRFH2
<Property>	<usr>	n/a	<Property>
<folder>. <Property>	<folder>	n/a	<Property>
<folder>. <group>. <Property>	<folder>	<group>	<Property>

Ad esempio, quando un'applicazione IBM MQ tenta di accedere alla proprietà `Property1` , questa viene associata all'elemento `Property1` nella cartella <usr> . La proprietà `wmq.Property2` è associata alla proprietà `Property2` nella cartella <wmq> .

Se il nome della proprietà ne contiene più di uno. il nome dell'elemento MQRFH2 utilizzato è quello che segue il nome finale. e i gruppi MQRFH2 vengono utilizzati per formare una gerarchia; i gruppi MQRFH2 nidificati sono consentiti.

Le proprietà specifiche del fornitore e dell'intestazione JMS contenute in MQRFH2 nelle cartelle <mcd>, <jms>e <mqext> sono accessibili da un'applicazione IBM MQ utilizzando i nomi brevi definiti in Utilizzo di IBM MQ classes for Java .

JMS le proprietà definite dall'utente sono accessibili dalla cartella <usr> . Un'applicazione IBM MQ può utilizzare la cartella <usr> per le proprietà dell'applicazione se è accettabile che la proprietà venga visualizzata dalle applicazioni JMS come una delle proprietà definite dall'utente.

Se non è accettabile, scegliere un'altra cartella; la cartella <wmq_usr> viene fornita come ubicazione standard per tali proprietà nonJMS .

Le applicazioni possono specificare e utilizzare qualsiasi cartella MQRFH2 con un utilizzo ben definito, non documentato in “Proprietà specificate come elementi MQRFH2” a pagina 956 se si nota quanto segue:

1. La cartella potrebbe essere già in uso o potrebbe essere utilizzata in futuro da un'altra applicazione che fornisce un accesso non definito alle proprietà contenute al suo interno; consultare Nomi delle proprietà per la convenzione di denominazione consigliata per i nomi delle proprietà.
2. Le proprietà non sono accessibili alle versioni precedenti del client IBM MQ classes for JMS o XMS che può accedere solo alla cartella <usr> per le proprietà definite dall'utente

3. La cartella deve essere contrassegnata con l'attributo `content` con il valore impostato su `properties`, ad esempio `content='properties'`.

   "MQSETMP - Imposta proprietà messaggio" a pagina 799 aggiunge automaticamente questo attributo come richiesto. Questo attributo non deve essere aggiunto a nessuna delle cartelle definite da IBM, ad esempio <jms> e <usr>. In questo modo, il messaggio viene rifiutato dal client IBM MQ classes for JMS prima di IBM WebSphere MQ 7.0. con un `MessageFormatException`.

Poiché la cartella <usr> è l'ubicazione predefinita per le proprietà della sintassi <Property> , un'applicazione IBM MQ e un'applicazione JMS per accedere allo stesso valore della proprietà definita dall'utente utilizzando lo stesso nome.

Nomi cartella riservati

Esistono diversi nomi di cartelle riservate. Non è possibile utilizzare tali nomi come prefissi di cartella; ad esempio, `Root.Property1` non accede ad una proprietà valida perché `Root` è riservato. Il seguente elenco contiene nomi di cartelle riservate:

- Radice
- Corpo
- Proprietà
- Ambiente
- LocalEnvironment
- DestinationList
- ExceptionList
- InputBody
- InputRoot
- InputProperties
- InputLocalEnvironment
- InputDestinationList
- InputExceptionList
- OutputRoot
- OutputLocalEnvironment
- OutputDestinationList
- OutputExceptionList

Associazione dei campi descrittore delle proprietà nelle intestazioni MQRFH2

Quando una proprietà viene convertita in un elemento MQRFH2 , i seguenti attributi dell'elemento vengono utilizzati per specificare i campi significativi del descrittore della proprietà: descrive il modo in cui i campi MQPD vengono tradotti in attributi dell'elemento MQRFH2 .

Supporto

Il campo descrittore della proprietà di supporto è suddiviso in tre attributi elemento

- L'attributo dell'elemento **sr** specifica valori nella maschera di bit MQPD_REJECT_UNSUP_MASK.
- L'attributo dell'elemento **sa** specifica i valori nella maschera di bit MQPD_ACCEPT_UNSUP_MASK.
- L'attributo dell'elemento **sx** specifica i valori nella maschera di bit MQPD_ACCEPT_UNSUP_IF_XMIT_MASK.

Questi attributi dell'elemento sono validi solo nella cartella < mq> e vengono ignorati se impostati sugli elementi nelle altre cartelle che contengono le proprietà.

Tabella 638. Campi MQPD associati agli attributi dell'elemento MQRFH2		
Valore di supporto	**attributo elemento MQRFH2**	**Valore dell'attributo MQRFH2**
MQPD_SUPPORT_OPZIONALE	sa	facoltativo Questo è il valore predefinito.
MQPD_SUPPORT_XX_ENCODE_CASE_ONE obbligatorio	sr	obbligatorio
MQPD_SUPPORT_REQUIRED_IF_LOCAL	sx	locale

Contesto

Utilizzare l'attributo dell'elemento **context** per indicare il contesto del messaggio a cui appartiene una proprietà. Utilizzare un solo valore. Questo attributo dell'elemento è valido su una proprietà in qualsiasi cartella contenente le proprietà.

Tabella 639. Valori di contesto associati a valori di attributo MQRFH2	
Valore contesto	**Valore dell'attributo MQRFH2**
MQPD_NO_CONTEXT	Nessuna Questo è il valore predefinito.
Contesto_UTENTE MQPD_	utente

CopyOptions

Utilizzare l'attributo dell'elemento **copy** per indicare i messaggi in cui deve essere copiata una proprietà. È consentito più di un valore; separare più valori con una virgole. Ad esempio, **copy='reply'** e **copy='publish,report'** sono entrambi validi. Questo attributo dell'elemento è valido su una proprietà in qualsiasi cartella contenente le proprietà.

Nota: Nella definizione dell'attributo, le virgolette singole o doppie sono un uso valido, ad esempio **copy='reply'** o **copy="report"**

Tabella 640. Valori CopyOption associati ai valori dell'attributo MQRFH2	
valore CopyOption	**Valore dell'attributo MQRFH2**
MQPD_COPY_FORWARD	inoltro
MQPD_COPY_REPLY	risposta
REPORT MQPD_COPY_REPORT	report
MQPD_COPY_PUBLISH	pubblicare
MQPD_COPY_ALL	tutti Non specificarlo con altri valori. Quando viene utilizzato con un altro valore, questo ha la precedenza su qualsiasi valore tranne **none**.
MQPD_COPY_DEFAULT	valore predefinito Questo è il valore predefinito. Equivale a specificare i tre valori MQCOPY_FORWARD, MQCOPY_REPORT e MQCOPY_PUBLISH. Non specificarlo con altri valori.

Tabella 640. Valori CopyOption associati ai valori dell'attributo MQRFH2 (Continua)	
valore CopyOption	**Valore dell'attributo MQRFH2**
MQPD_COPY_NONE	Nessuna Non specificarlo con altri valori. Se utilizzato con un altro valore, ha la precedenza.

Restrizioni per la cartella < mq> MQRFH2

Quando un messaggio viene inserito in una coda, viene ricercata una cartella < mq> in modo che il messaggio possa essere elaborato in base alle relative proprietà definite da MQ. Per consentire l'analisi efficiente delle proprietà definite da MQ, alla cartella si applicano le seguenti restrizioni:

- Solo le proprietà nella prima cartella < mq> significativa del messaggio vengono utilizzate da MQ; le proprietà in qualsiasi altra cartella < mq> del messaggio vengono ignorate.
- Se la cartella è in UTF-8, nella cartella sono consentiti solo UTF-8 caratteri a byte singolo. Un carattere multibyte nella cartella può causare un errore di analisi e il messaggio può essere rifiutato.
- Non includere i gruppi MQRFH2 nella cartella < mq>. La presenza del carattere Unicode U+003C in un valore della proprietà causerà il rifiuto del messaggio.
- Non utilizzare le stringhe di escape nella cartella. Una stringa di escape viene considerata come il valore effettivo dell'elemento.
- Solo il carattere Unicode U+0020 viene trattato come spazio vuoto all'interno della cartella. Tutti gli altri caratteri vengono trattati come significativi e possono causare l'esito negativo dell'analisi della cartella e il rifiuto del messaggio.

Se l'analisi della cartella < mq> non riesce o se la cartella non rispetta queste limitazioni, il messaggio viene rifiutato con CompCode **MQCC_FAILED** e Reason **MQRC_RFH_RESTRICTED_FORMAT_ERR**.

Intestazioni MQRFH2 non valide

Al momento dell'elaborazione di una chiamata MQPUT, MQPUT1o MQGET, può verificarsi un'analisi parziale di tutte le intestazioni MQRFH2 nel messaggio per controllare quali cartelle sono incluse e per determinare se le cartelle contengono proprietà. Panoramica delle intestazioni MQRFH2 non valide.

Se l'analisi parziale del messaggio non può essere completata correttamente perché la struttura non è valida, ad esempio, il campo `StrucLength` è troppo piccolo:

- La chiamata MQPUT o MQPUT1 ha esito negativo con codice motivo MQRC_RFH_ERROR, se è possibile determinare che l'applicazione include alcune opzioni IBM WebSphere MQ 7 , in modo che le applicazioni esistenti non abbiano esito negativo.
- La chiamata MQGET viene restituita correttamente e la MQRFH2 contenente l'errore viene restituita nel buffer fornito.

Se l'analisi parziale ha esito negativo perché non è possibile rilevare se una particolare cartella contiene proprietà o meno, ad esempio, la cartella inizia con `<<jms`, quindi l'analisi ha esito negativo prima che venga determinato il nome della cartella, quindi:

- La chiamata MQPUT o MQPUT1 ha esito negativo con codice motivo MQRC_RFH_FORMAT_ERROR, se è possibile determinare che l'applicazione include alcune opzioni IBM WebSphere MQ 7 , in modo che le applicazioni esistenti non abbiano esito negativo.
- La chiamata MQGET viene restituita correttamente e la MQRFH2 contenente l'errore viene restituita nel buffer fornito.
- Mentre è internamente all'interno del gestore code, il messaggio non viene rifiutato a causa della cartella formattata in modo non corretto, ma la cartella viene sempre trattata come se non fosse contenuta alcuna proprietà al suo interno.

Un messaggio può passare attraverso la rete del gestore code con una cartella che contiene tale errore di sintassi, ma non viene mai analizzato e rilevato, mentre una o più cartelle nel messaggio sono:

- Valido
- Analizzato correttamente
- Utilizzato nell'elaborazione del messaggio

Pertanto, il rilevamento non è garantito.

Se una delle applicazioni utilizza “MQSETMP - Imposta proprietà messaggio” a pagina 799, o MQINQMP per accedere a una proprietà, e in questo modo si determina l'analisi completa di una cartella MQRFH2 , rilevando un errore tale che l'analisi non può essere completata, ciò è indicato da un codice di ritorno appropriato per la chiamata API. Nessuna proprietà nella cartella è resa disponibile per l'applicazione.

Se viene effettuato un tentativo di analisi completa di una cartella MQRFH2 e il programma di analisi rileva attributi di elementi non riconosciuti o un tipo di dati non riconosciuto, l'analisi continua e viene completata correttamente senza l'emissione di avvertenze; ciò non costituisce un errore di analisi.

Conversione code page

Questa sezione descrive i nomi e i CCSID della serie di codici, la lingua nazionale, la conversione z/OS , la conversione IBM i e il supporto per la conversione Unicode.

Ogni sezione della lingua nazionale elenca le seguenti informazioni:

- I CCSID nativi supportati
- Le conversioni di codepage non supportate

I seguenti termini vengono utilizzati nelle informazioni:

AIX
Indica IBM MQ for AIX.

Linux
Indica IBM MQ per Linux per Intel e IBM MQ per Linux per zSeries.

OS/400
Indica IBM MQ for IBM i.

Windows
Indica IBM MQ for Windows.

z/OS
Indica IBM MQ for z/OS.

Il valore predefinito per la conversione dei dati è per la conversione da eseguire sul sistema di destinazione (ricevente).

Se il prodotto di origine supporta la conversione, è possibile impostare un canale e scambiare i dati impostando l'attributo del canale CONVERT su YES nell'origine.

Nota:

1. La conversione per le informazioni IBM MQ MQI client avviene nel server, quindi il server deve supportare la conversione dal CCSID del client al CCSID del server.
2. La conversione potrebbe includere il supporto aggiunto da CSD/PTF alla versione più recente di IBM MQ. Controllare il contenuto del livello di servizio più recente per verificare se è necessario installare un CSD/PTF per abilitare questa conversione.
3. Il CCSID del gestore code IBM MQ deve essere misto o SBCS.
4. Alcuni CCSID, ad esempio 850 su AIX, che non sono supportati dal sistema operativo possono ancora essere utilizzati dall'applicazione e possono anche essere impostati come CCSID del gestore code IBM MQ . Ciò è consentito solo per scopi di compatibilità con le versioni precedenti e la conversione avrà esito negativo se le relative tabelle di conversione non sono installate.

Consultare Tabella 641 a pagina 966 per un riferimento incrociato tra alcuni dei numeri CCSID e alcuni nomi di codeset di settore.

Riferimenti correlati

“Lingue nazionali” a pagina 966
Queste informazioni contengono le lingue supportate da IBM MQ.

Nomi di codeset e CCSID

I nomi dei codeset e i CCSID corrispondenti per ciascun nome di codeset.

z/OS IBM MQ for z/OS fornisce più conversioni di quelle elencate nelle tabelle specifiche della lingua. Per un elenco completo delle conversioni, consultare Table 674 a pagina 992.

Tabella 641. Nomi di codeset e CCSID

Nomi codeset	**CCSID**
ISO 8859-1	819
ISO 8859-2	912
ISO 8859-3	913
ISO 8859-5	915
ISO 8859-6	1089
ISO 8859-7	813
ISO 8859-8	916
ISO 8859-9	920
ISO 8859-13	921
ISO 8859-15 (euro)	923
big5	950
eucJP	954 5050 33722
eucKR	970
eucTW	964
eucCN	1383
pck	943
GBK	1386
koi8-r	878

Lingue nazionali

Queste informazioni contengono le lingue supportate da IBM MQ.

Le lingue supportate da IBM MQ sono:

- Inglese americano - consultare l'argomento “Inglese (Stati Uniti)” a pagina 967
- Tedesco - consultare l'argomento “Tedesco” a pagina 968
- Danese e norvegese - consultare l'argomento “Danese e norvegese” a pagina 968
- Finlandese e svedese - vedi argomento “Finlandese e svedese” a pagina 969
- Italiano - vedere l'argomento “Italiano” a pagina 970
- Spagnolo - consultare l'argomento “Spagnolo” a pagina 971

- Inglese britannico / Gaelico - consultare l'argomento “Inglese britannico /Gaelico” a pagina 971
- Francese - consultare l'argomento “Francese” a pagina 972
- Multilingue - consultare l'argomento “Multilinguaggio” a pagina 973
- Portoghese - consultare l'argomento “Portoghese” a pagina 973
- Islandese - consultare l'argomento “Islandese” a pagina 974
- Lingue dell'Europa orientale - vedi argomento “Lingue dell'Europa orientale” a pagina 975
- Cirillico - consultare l'argomento “cirillico” a pagina 976
- Estone - consultare l'argomento “Estone” a pagina 977
- Lettone e lituano - vedere l'argomento “Lettone e lituano” a pagina 978
- Ukranian - consultare l'argomento “Ucraino” a pagina 979
- Greco - consultare l'argomento “Greco” a pagina 980
- Turco - consultare l'argomento “Turco” a pagina 980
- Ebraico - consultare l'argomento “Ebraico” a pagina 981
- Farsi - vedere l'argomento “Farsi” a pagina 983
- Urdu - consultare l'argomento “Urdu” a pagina 983
- Tailandese - consultare l'argomento “Tailandese” a pagina 984
- Lao - vedi argomento “Lao” a pagina 984
- Vietnamita - consultare l'argomento “Vietnamita” a pagina 985
- Giapponese latino SBCS - consultare l'argomento “Giapponese latino SBCS” a pagina 985
- Giapponese Katakana SBCS - consultare l'argomento “SBCS Katakana giapponese” a pagina 986
- Giapponese Kanji / Latino misto - consultare l'argomento “Giapponese Kanji / Latino misto” a pagina 987
- Giapponese Kanji / Katakana Misto - vedere l'argomento “Giapponese Kanji / Katakana misto” a pagina 988
- Coreano - consultare l'argomento “Coreano” a pagina 989
- Cinese semplificato - consultare l'argomento “Cinese semplificato” a pagina 990
- Cinese tradizionale - vedi argomento “Cinese tradizionale” a pagina 991

Inglese (Stati Uniti)

Dettagli dei CCSID e della conversione CCSID per l'inglese americano.

Tabella 642. CCSID nativi per l'inglese americano sulle piattaforme supportate

Piattaforma	**CCSID nativi**
IBM i IBM i z/OS z/OS	37, 924, 1140
AIX AIX	819, 923, 5348
Windows Windows	437, 850, 1252, 5348, 858
Linux Linux	819, 923
Client Apple	1275

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

280
Non converte in codepage 923, 858

924
Non converte in codepage 437, 858, 1051, 1140, 1252, 1275, 5348

1140
Non converte in codepage 924, 1051, 1275

Tedesco

Dettagli dei CCSID e della conversione CCSID per il tedesco.

Tabella 643. CCSID nativi per il tedesco su piattaforme supportate

Piattaforma	CCSID nativi
IBM i IBM i z/OS z/OS	273, 924, 1141
AIX AIX	819, 923, 5348
Windows Windows	437, 850, 858, 1252, 5348
Linux Linux	819, 923
Client Apple	1275

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

273
Non converte in codepage 858, 923, 924, 1275

924
Non converte in codepage 273, 437, 858, 1051, 1141, 1252, 1275, 5348

1141
Non converte in codepage 924, 1051, 1275

Danese e norvegese

Dettagli dei CCSID e della conversione CCSID per il danese e il norvegese.

Tabella 644. CCSID nativo per danese e norvegese sulle piattaforme supportate

Piattaforma	CCSID nativi
IBM i IBM i z/OS z/OS	277, 924, 1142

Tabella 644. CCSID nativo per danese e norvegese sulle piattaforme supportate (Continua)	
Piattaforma	**CCSID nativi**
AIX	819, 923, 5348
Windows	850, 858, 865, 1252, 5348
Linux	819, 923
Client Apple	1275

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

277
Non converte in codepage 858, 923, 924, 1275

924
Non converte in codepage 277, 858, 865, 1051, 1142, 1252, 1275, 5348

1142
Non converte in codepage 924, 865, 1051, 1275

AIX

AIX

Codepage:

819
Non esegue la conversione nella codepage 865

Windows

Windows

Codepage:

865
Non converte in codepage 1051, 1275

Finlandese e svedese

Dettagli dei CCSID e della conversione CCSID per il finlandese e lo svedese.

Tabella 645. CCSID nativi per il finlandese e lo svedese sulle piattaforme supportate	
Piattaforma	**CCSID nativi**
IBM i z/OS	278, 924, 1143
AIX	819, 923, 5348
Windows	437, 850, 858, 865, 1252, 5348

Tabella 645. CCSID nativi per il finlandese e lo svedese sulle piattaforme supportate (Continua)	
Piattaforma	**CCSID nativi**
Linux Linux	819, 923
Client Apple	1275

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

278
Non converte in codepage 858, 923, 924, 1275

924
Non converte in codepage 278, 437, 858, 865, 1051, 1143, 1252, 1275, 5348

1143
Non converte in codepage 865, 924, 1051, 1275

AIX

AIX

Codepage:

819
Non esegue la conversione nella codepage 865

850
Non esegue la conversione nella codepage 865

Windows

Windows

Codepage:

865
Non converte in codepage 1051, 1275

Italiano

Dettagli dei CCSID e della conversione CCSID per l'italiano.

Tabella 646. CCSID nativi per l'italiano su piattaforme supportate	
Piattaforma	**CCSID nativi**
IBM i IBM i z/OS z/OS	280, 924, 1144
AIX AIX	819, 923, 5348
Windows Windows	437, 850, 858, 1252, 5348
Linux Linux	819, 923

Tabella 646. CCSID nativi per l'italiano su piattaforme supportate (Continua)	
Piattaforma	**CCSID nativi**
Client Apple	1275

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

280
Non converte in codepage 858, 923, 924, 1275

924
Non converte in codepage 280, 437, 858, 1051, 1144, 1252, 1275, 5348

1144
Non converte in codepage 924, 1051, 1275

Spagnolo

Dettagli dei CCSID e della conversione CCSID per lo spagnolo.

Tabella 647. CCSID nativi per lo spagnolo sulle piattaforme supportate	
Piattaforma	**CCSID nativi**
IBM i IBM i z/OS z/OS	284, 924, 1145
AIX AIX	819, 923, 5348
Windows Windows	437, 850, 858, 1252, 5348
Linux Linux	819, 923
Client Apple	1275

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

284
Non converte in codepage 858, 923, 924, 1275

924
Non converte in codepage 284, 437, 858, 1051, 1145, 1252, 1275, 5348

1145
Non converte in codepage 924, 1051, 1275

Inglese britannico /Gaelico

Dettagli dei CCSID e della conversione CCSID per l'inglese britannico / gaelico.

Tabella 648. CCSID nativi per inglese britannico / gaelico su piattaforme supportate	
Piattaforma	**CCSID nativi**
IBM i IBM i z/OS z/OS	285, 924, 1146
AIX AIX	819, 923, 5348
Windows Windows	437, 850, 858, 1252, 5348
Linux Linux	819, 923
Client Apple	1275

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

285
Non converte in codepage 858, 923, 924, 1275

924
Non converte in codepage 285, 437, 858, 1051, 1146, 1252, 1275, 5348

1146
Non converte in codepage 924, 1051, 1275

Francese

Dettagli dei CCSID e della conversione CCSID per il francese.

Tabella 649. CCSID nativi per il francese su piattaforme supportate	
Piattaforma	**CCSID nativi**
IBM i IBM i z/OS z/OS	297, 924, 1147
AIX AIX	819, 923, 5348
Windows Windows	437, 850, 858, 1252, 5348
Linux Linux	819, 923
Client Apple	1275

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

297
Non converte in codepage 858, 923, 924, 1275, 5348

924
Non converte in codepage 297, 437, 858, 1051, 1147, 1252, 1275, 5348

1147
Non converte in codepage 924, 1051, 1275

Multilinguaggio

Dettagli dei CCSID e della conversione CCSID per Multilingual.

Tabella 650. CCSID nativi per la conversione multilingue su piattaforme supportate

Piattaforma	CCSID nativi
IBM i z/OS	500, 924, 1148
AIX	819, 923, 5348
Windows	437, 850, 858, 1252, 5348
Linux	819, 923
Client Apple	1275

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

500
Non converte in codepage 858, 923

924
Non converte in code page 437, 858, 1051, 1148, 1252, 1275, 5348

1148
Non converte in codepage 924, 1051, 1275

Portoghese

Dettagli dei CCSID e della conversione CCSID per il portoghese.

Tabella 651. CCSID nativi per il portoghese sulle piattaforme supportate

Piattaforma	CCSID nativi
IBM i	37, 500, 924, 1140
z/OS	500, 924, 1140
AIX	819, 923, 5348
Windows	850, 858, 860, 1252, 5348

Tabella 651. CCSID nativi per il portoghese sulle piattaforme supportate (Continua)	
Piattaforma	**CCSID nativi**
Linux Linux	819, 923
Client Apple	1275

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

280
Non converte in code page 858, 923, 1275

500
Non converte in code page 858, 923, 1275

924
Non converte in codepage 858, 860, 1051, 1140, 1252, 1275, 5348

1140
Non converte in codepage 860, 924, 1051, 1275

Windows

Windows

Codepage:

860
Non converte in codepage 1051, 1275

Islandese

Dettagli dei CCSID e della conversione CCSID per l'islandese.

Tabella 652. CCSID nativi per l'islandese sulle piattaforme supportate	
Piattaforma	**CCSID nativi**
IBM i IBM i z/OS z/OS	871, 924, 1149
AIX AIX	819, 923, 5348
Windows Windows	850, 858, 861, 1252, 5348
Linux Linux	819, 923
Client Apple	1275

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

871
Non converte in codepage 858, 923, 924, 1275, 5348

924
Non si converte in codepage 858, 861, 871, 1051, 1149, 1252, 1275, 5348

1149
Non converte in codepage 924, 1051, 1275

Windows

Windows

Codepage:

861
Non converte in codepage 1051, 1275

Lingue dell'Europa orientale

Dettagli dei CCSID e della conversione CCSID per le lingue dell'Europa orientale. Le lingue tipiche che utilizzano questi CCSID includono albanese, croato, ceco, ungherese, polacco, rumeno, serbo, slovacco e sloveno.

Tabella 653. CCSID nativi per le lingue dell'Europa orientale sulle piattaforme supportate

Piattaforma	**CCSID nativi**
IBM i IBM i z/OS z/OS	870, 1153
Windows Windows	852, 1250, 5346, 9044
AIX AIX Linux Linux	912
Client Apple dell'Europa orientale	1282
Client Apple rumeno	1285
Client Apple croato	1284

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme, con le seguenti eccezioni.

z/OS

z/OS

Codepage:

870
Non converte in codepage 1284, 1285

1153
Non converte in codepage 1250, 1284, 1285

IBM i

IBM i

Codepage:

870
Non converte in codepage 1284, 1285, 5346, 9044

1153
Non converte in code page 1282, 1284, 1285, 5346, 9044

, Linux

Linux

Codepage:

912
Non converte in codepage 1284, 1285

Windows

Windows

Codepage:

852
Non converte in codepage 1284, 1285

1250
Non converte in codepage 1284, 1285

9044
Non converte in codepage 912, 1282, 1284, 1285

cirillico

Dettagli dei CCSID e della conversione CCSID per Cirillico. Le lingue tipiche che utilizzano questi CCSID includono il bielorusso, il bulgaro, il macedone, il russo e il serbo.

Tabella 654. CCSID nativi per Cyrillic su piattaforme supportate

Piattaforma	**CCSID nativi**
z/OS z/OS	1025
IBM i IBM i	880, 1025
Windows Windows	855, 866, 1131, 1251, 5347
AIX AIX Linux Linux	915
Client Apple	1283

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

880
Non converte in codepage 855, 866, 878, 1131, 5347

1025
Non converte in codepage 878, 5347

Windows

Windows

Codepage:

855
Non converte nella codepage 1131

866
Non converte nella codepage 1131

1131
Non converte in codepage 855, 866, 880, 1283

Estone

Dettagli dei CCSID e della conversione CCSID per l'estone.

Tabella 655. CCSID nativi per l'estone sulle piattaforme supportate

Piattaforma	CCSID nativi
IBM i IBM i z/OS z/OS	1122, 1157
Windows Windows	902, 922, 1257, 5353, 9449
AIX AIX Linux Linux	902, 922

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi di altre piattaforme, con le seguenti eccezioni.

z/OS

z/OS

Codepage:

1122
Non converte in codepage 902, 1157, 9449

1157
Non converte in codepage 922, 1122, 1257, 9449

IBM i

IBM i

Codepage:

1122
Non converte in codepage 902, 5353, 9449

1157
Non converte in codepage 922, 5353, 9449

Linux

Linux

Codepage:

902
Non converte in codepage 922, 1122, 9449

922
Non converte in codepage 902, 1157, 9449

Windows

Windows

Codepage:

5353
Non converte nella codepage 9449

9449
Non converte in codepage 902, 922, 1122, 1157, 1257, 5353

902
Non converte in codepage 922, 1122, 9449

Lettone e lituano

Dettagli dei CCSID e della conversione CCSID per lettone e lituano.

Tabella 656. CCSID nativi per lettone e lituano sulle piattaforme supportate

Piattaforma	CCSID nativi
IBM i IBM i z/OS z/OS	1112, 1156
Windows Windows	901, 921, 1257, 5353, 9449
AIX AIX Linux Linux	901, 921

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi di altre piattaforme, con le seguenti eccezioni.

z/OS

z/OS

Codepage:

1112
Non converte in codepage 901, 1156, 9449

1156
Non converte in codepage 901, 1156, 9449

IBM i

IBM i

Codepage:

1112
Non converte nella codepage 5353

1153
Non converte in codepage 921, 5353, 9449

Linux

Linux

Codepage:

902
Non converte in codepage 921, 1112, 1257, 9449

921
Non converte in codepage 901, 1156, 9449

Windows

Windows

Codepage:

901
Non converte in codepage 921, 1112, 1257, 9449

5355
Non converte nella codepage 9449

9449
Non converte in codepage 901, 921, 1112, 1156, 1257

Ucraino

Dettagli dei CCSID e della conversione CCSID per l'ucraino.

Tabella 657. CCSID nativi per Ukranian su piattaforme supportate

Piattaforma	CCSID nativi
IBM i IBM i z/OS z/OS	1123
Windows Windows	1124, 1125, 1251, 5347
AIX AIX Linux Linux	1124

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi di altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

1123
Non converte nella codepage 5347

Windows

Windows

Codepage:

1125
Non converte nella codepage 1123

Greco

Dettagli dei CCSID e della conversione CCSID per il greco.

Tabella 658. CCSID nativi per il greco sulle piattaforme supportate

Piattaforma	CCSID nativi
IBM i IBM i z/OS z/OS	875
Windows Windows	869, 1253, 5349
AIX AIX Linux Linux NCR	813
Client Apple	1280
Client DOS	737

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi, i CCSID nativi delle altre piattaforme con le seguenti eccezioni.

IBM i

IBM i

Codepage:

875
Non converte nella codepage 5349

Windows

Windows

Codepage:

1253
Non converte nella codepage 737

5349
Non converte nella codepage 737

Turco

Dettagli dei CCSID e della conversione CCSID per il turco.

Tabella 659. CCSID nativi per il turco sulle piattaforme supportate	
Piattaforma	**CCSID nativi**
IBM i IBM i z/OS z/OS	1026
Windows Windows	857, 1254, 5350
AIX AIX Linux Linux	920
Client Apple	1281

Tutte le piattaforme non client supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

1026
Non converte in codepage 5350

Ebraico

Dettagli dei CCSID e della conversione CCSID per l'ebraico.

Tabella 660. CCSID nativi per l'ebraico su piattaforme supportate	
Piattaforma	**CCSID nativi**
z/OS z/OS	424, 803, 4899, 12712
IBM i IBM i	424
AIX AIX	916, 9048
Windows Windows	1255, 5351
Linux Linux	916

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi di altre piattaforme, con le seguenti eccezioni.

z/OS

z/OS

Codepage:

424
Non converte in codepage 867, 4899, 9048, 12712

803
Non converte in codepage 867, 4899, 5351, 9048, 12712

4899
Non converte nelle codepage 424, 803, 856, 862, 916, 1255

12712
Non converte in codepage 424, 803, 856, 916, 1255

IBM i

IBM i

Codepage:

424
Non converte in code page 803, 867, 4899, 5351, 9048, 12712

La codepage 424 converte anche in e da CCSID 4952, che è una variante di 856.

AIX

AIX

Codepage:

916
Non converte in codepage 867, 4899, 9048, 12712

9048
Non converte nelle codepage 424, 803, 856, 862, 916, 1255

Windows

Windows

Codepage:

1255
Non converte in codepage 867, 4899, 9048, 12712

5351
Non converte nella codepage 803

Arabo

Dettagli dei CCSID e della conversione CCSID per l'arabo

Tabella 661. CCSID nativi per l'arabo sulle piattaforme supportate

Piattaforma	**CCSID nativi**
IBM i IBM i z/OS z/OS	420
AIX AIX	1046, 1089
	1089 (vedere nota)
Windows Windows	720, 864, 1256, 5352
Linux Linux	1089

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi di altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

420
Non converte nella codepage 5352

Linux, Tru64

Linux

Codepage:

1089
Non converte nella codepage 720

Windows

Windows

Codepage:

720
Non converte in code page 1089, 5352

5352
Non converte nella codepage 720

Farsi

Dettagli dei CCSID e della conversione CCSID per farsi.

Tabella 662. CCSID nativi per farsi su piattaforme supportate

Piattaforma	CCSID nativi
IBM i IBM i z/OS z/OS	1097
AIX AIX Linux Linux Windows Windows	1098 (vedere nota)

Nota: Il CCSID nativo per queste piattaforme non è stato standardizzato e potrebbe essere modificato.

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme.

Urdu

Dettagli dei CCSID e della conversione CCSID per Urdu.

Tabella 663. CCSID nativi per Urdu su piattaforme supportate

Piattaforma	CCSID nativi
IBM i IBM i z/OS z/OS	918
Windows Windows	868

Tabella 663. CCSID nativi per Urdu su piattaforme supportate (Continua)	
Piattaforma	**CCSID nativi**
AIX AIX Linux Linux	1006

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi di altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

918
Non converte nella codepage 1006

Tailandese

Dettagli dei CCSID e della conversione CCSID per il tailandese.

Tabella 664. CCSID nativi per il tailandese sulle piattaforme supportate	
Piattaforma	**CCSID nativi**
IBM i IBM i z/OS z/OS	838
AIX AIX Linux Linux Windows Windows	874 (vedere nota)

Nota: Il CCSID nativo per queste piattaforme non è stato standardizzato e potrebbe essere modificato.

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme.

Lao

Dettagli dei CCSID e della conversione CCSID per il Laos.

Tabella 665. CCSID nativi per il Laos su piattaforme supportate	
Piattaforma	**CCSID nativi**
IBM i IBM i z/OS z/OS	1132
AIX AIX Linux Linux Windows Windows	1133

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi delle altre piattaforme.

Vietnamita

Dettagli dei CCSID e della conversione CCSID per il vietnamita.

Tabella 666. CCSID nativi per il vietnamita sulle piattaforme supportate	
Piattaforma	**CCSID nativi**
IBM i z/OS	1130
Windows	1258, 5354
AIX Linux	1129

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi di altre piattaforme, con le seguenti eccezioni.

IBM i

IBM i

Codepage:

1130
Non converte in codepage 1129, 5354

Giapponese latino SBCS

Dettagli dei CCSID e della conversione CCSID per il giapponese latino SBCS.

Tabella 667. CCSID nativi per SBCS latino giapponese su piattaforme supportate	
Piattaforma	**CCSID nativi**
IBM i z/OS	1027
AIX	932, 5050, 33722 (vedere nota 1)
Windows	932, 943 (vedi nota 2)
Linux	943, 5050

Nota:

1. AIX 5050 e 33722 sono CCSID relativi alla codepage di base 954 su AIX. Il CCSID notificato dal sistema operativo è 33722.
2. Windows Windows NT utilizza la codepage 932, ma questa è rappresentata al meglio dal CCSID 943. Tuttavia, non tutte le piattaforme di IBM MQ supportano questo CCSID.

   Su IBM MQ for Windows CCSID 932 viene utilizzato per rappresentare la codepage 932, ma è possibile apportare una modifica al file `../conv/table/ccsid.tbl` che modifica il CCSID utilizzato in 943.

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi di altre piattaforme, con le seguenti eccezioni.

z/OS

z/OS

Codepage:

1027
Non si converte in codepage 932, 942, 943, 954, 5050, 33722

IBM i

IBM i

Codepage:

1027
Non converte nella codepage 932

AIX

AIX

Codepage:

932
Non converte in codepage 1027

5050
Non converte in codepage 1027

33722
Non converte in codepage 1027

Linux

Linux

Codepage:

943
Non converte in codepage 1027

5050
Non converte in codepage 1027

SBCS Katakana giapponese

Dettagli dei CCSID e della conversione CCSID per il giapponese Katakana SBCS.

Tabella 668. CCSID nativi per SBCS Katakana giapponese su piattaforme supportate

Piattaforma	CCSID nativi
IBM i IBM i z/OS z/OS	290
AIX AIX	932, 5050, 33722 (vedere nota 1)
Windows Windows	932, 943 (vedi nota 2)
Linux Linux	943, 5050

Nota:

1. AIX 5050 e 33722 sono CCSID relativi alla codepage di base 954 su AIX. Il CCSID notificato dal sistema operativo è 33722.
2. Windows Windows NT utilizza la codepage 932, ma questa è rappresentata al meglio dal CCSID 943. Tuttavia, non tutte le piattaforme di IBM MQ supportano questo CCSID.

   Su IBM MQ for Windows CCSID 932 viene utilizzato per rappresentare la codepage 932, ma è possibile apportare una modifica al file `../conv/table/ccsid.tbl` che modifica il CCSID utilizzato in 943.
3. Oltre alle conversioni precedenti, IBM MQ supporta la conversione da CCSID 897 a CCSID 37, 273, 277, 278, 280, 284, 285, 290, 297, 437, 500, 819, 850, 1027 e 1252 sulle piattaforme seguenti:
   - AIX AIX
   - Linux Linux

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi di altre piattaforme, con le seguenti eccezioni.

z/OS

z/OS

Codepage:

290
Non converte in code page 932, 943, 954, 5050, 33722

IBM i

IBM i

Codepage:

290
Non converte nella codepage 932

AIX

AIX

Codepage:

932
Non converte in codepage 290, 897

5050
Non converte in codepage 290, 897

33722
Non converte in codepage 290, 897

Linux

Linux

Codepage:

943
Non converte in codepage 290, 897

5050
Non converte in codepage 290, 897

Giapponese Kanji / Latino misto

Dettagli dei CCSID e della conversione CCSID per giapponese Kanji / latino misto.

Tabella 669. CCSID nativi per Kanji giapponese / Latin misti su piattaforme supportate	
Piattaforma	**CCSID nativi**
IBM i IBM i z/OS z/OS	1399, 5035 (vedi nota 1)
AIX AIX	932, 5050, 33722 (vedere nota 2)
Windows Windows	932, 943 (vedi nota 4)
Linux Linux	943, 5050

Nota:

1. IBM i z/OS 5035 è un CCSID correlato alla codepage 939
2. AIX 5050 e 33722 sono CCSID relativi alla codepage di base 954 su AIX. Il CCSID notificato dal sistema operativo è 33722.
3. Windows Windows NT utilizza la codepage 932, ma questa è rappresentata al meglio dal CCSID 943. Tuttavia, non tutte le piattaforme di IBM MQ supportano questo CCSID.

   Su IBM MQ for Windows CCSID 932 viene utilizzato per rappresentare la codepage 932, ma è possibile apportare una modifica al file `../conv/table/ccsid.tbl` che modifica il CCSID utilizzato in 943.

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi di altre piattaforme, con le seguenti eccezioni.

z/OS

z/OS

Codepage:

1399
Non converte in code page 954, 5035, 5050, 33722

5035
Non converte in codepage 954, 1399, 5050, 33722

IBM i

IBM i

Codepage:

1399
Non converte nella codepage 5039

5035
Non converte nella codepage 5039

Giapponese Kanji / Katakana misto

Dettagli dei CCSID e della conversione CCSID per giapponese Kanji / Katakana Mixed.

Tabella 670. CCSID nativi per Kanji / Katakana giapponese misti su piattaforme supportate	
Piattaforma	**CCSID nativi**
z/OS z/OS	1390, 5026 (consultare la nota “1” a pagina 989)

Tabella 670. CCSID nativi per Kanji / Katakana giapponese misti su piattaforme supportate (Continua)	
Piattaforma	**CCSID nativi**
IBM i	5026 (consultare la nota “1” a pagina 989)
AIX	932, 5050, 33722 (consultare la nota “2” a pagina 989)
Windows	932, 943 (vedere Nota “3” a pagina 989)
Linux	943, 5050

Nota:

1. IBM i z/OS La modalità a byte singolo dei CCSID 1390 e 5026 in EBCDIC contiene caratteri minuscoli in ubicazioni differenti rispetto al layout tipico o invariante per il latino di base. Pertanto è necessario assicurarsi che i dati non vadano persi quando i dati del messaggio vengono convertiti in altri CCSID. Inoltre, l'utilizzo di questi CCSID come CCSID predefinito del gestore code può causare problemi durante la comunicazione con altri gestori code. Ad esempio, i nomi canale che utilizzano caratteri minuscoli potrebbero non essere interpretati correttamente sul sistema remoto. 5026 è un CCSID correlato alla codepage 930. CCSID 5026 è il CCSID notificato su IBM i quando viene selezionata la funzione giapponese Katakana (DBCS).
2. AIX 5050 e 33722 sono CCSID relativi alla codepage di base 954 su AIX. Il CCSID notificato dal sistema operativo è 33722.
3. Windows Windows NT utilizza la codepage 932, ma questa è rappresentata al meglio dal CCSID 943. Tuttavia, non tutte le piattaforme di IBM MQ supportano questo CCSID.

   Su Windows, CCSID 932 viene usato per rappresentare la codepage 932, ma è possibile apportare una modifica al file `../conv/table/ccsid.tbl` che modifica il CCSID utilizzato in 943.

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi di altre piattaforme, con le seguenti eccezioni.

z/OS

z/OS

Codepage:

1390
Non converte in codepage 954, 5026, 5050, 33722

Non accetta caratteri minuscoli.

5026
Non converte nelle codepage 954, 1390, 5050, 33722

IBM i

IBM i

Codepage:

5026
Non converte in codepage 1390, 5039

Coreano

Dettagli dei CCSID e della conversione CCSID per il coreano.

Tabella 671. CCSID nativi per il coreano su piattaforme supportate	
Piattaforma	**CCSID nativi**
IBM i IBM i z/OS z/OS	933, 1364
AIX AIX Linux Linux	970
Windows Windows	949, 1363

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi di altre piattaforme, con le seguenti eccezioni.

z/OS

z/OS

Codepage:

933
Non converte in codepage 970

1364
Non converte in codepage 970

Cinese semplificato

Dettagli dei CCSID e della conversione CCSID per il cinese semplificato.

Tabella 672. CCSID nativi per il cinese semplificato su piattaforme supportate	
Piattaforma	**CCSID nativi**
z/OS z/OS	935, 1388
IBM i IBM i	935, 1388
AIX AIX	1383, 1386
Windows Windows	1381, 1386 (vedi nota 2)
Linux Linux	1383

Nota:

1. Windows Windows utilizza la codepage 936, ma ciò è meglio rappresentato dal CCSID di 1386. Tuttavia, non tutte le piattaforme di IBM MQ supportano questo CCSID.

   Su IBM MQ for Windows CCSID 1381 viene usato per rappresentare la codepage 936, ma è possibile apportare una modifica al file `../conv/table/ccsid.tbl` che modifica il CCSID utilizzato in 1386.
2. IBM MQ supporta lo standard cinese GB18030 .

   z/OS Linux Windows Su z/OS, Windows e Linux, il supporto di conversione viene fornito tra Unicode (UTF-8 e UTF-16) e CCSID 1388 (EBCDIC con estensioni GB18030), Unicode (UTF-8 e UTF-16) e CCSID 5488 (GB18030) e tra CCSID 1388 e CCSID 5488.

Nota: Il CCSID deve essere impostato su 5488 per poter utilizzare i caratteri GB18030 . Tuttavia, non è possibile impostare il CCSID su un gestore code creato con IBM MQ Explorer o IBM MQ Console. Invece, è necessario creare il gestore code utilizzando la CLI con un CCSID di 5488 oppure utilizzare la riga comandi CLI per modificare il CCSID dopo aver creato il gestore code.

IBM i Su IBM i, il supporto viene fornito dal sistema operativo per la conversione tra Unicode (UTF-8 e UTF-16) e CCSID 1388 (EBCDIC con estensioni GB18030).

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi di altre piattaforme, con le seguenti eccezioni.

z/OS

z/OS

Codepage:

935
Non converte nella codepage 1383

1388
Non converte nella codepage 1383

Cinese tradizionale

Dettagli dei CCSID e della conversione CCSID per il cinese tradizionale.

Tabella 673. CCSID nativi per il cinese tradizionale su piattaforme supportate

Piattaforma	CCSID nativi
IBM i IBM i z/OS z/OS	937
Windows Windows	950
AIX AIX Linux Linux	950, 964

Tutte le piattaforme supportano la conversione tra i CCSID nativi e i CCSID nativi di altre piattaforme, con le seguenti eccezioni.

z/OS

z/OS

Codepage:

937
Non converte in codepage 964

1388
Non converte nella codepage 1383

Linux

Linux

Codepage:

964
Non converte nella codepage 938

z/OS conversion support

A list of supported CCSID conversions.

Table 674. IBM MQ for z/OS CCSID conversion support

CCSID	Converts to and from CCSIDS
37	256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664, 28709
256	37, 273, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 819, 833, 836, 838, 850, 852, 857, 860-866, 869-871, 875, 880, 905, 1025-1027, 1112, 1122, 1200, 1208, 1251-1252, 1275, 4386, 4929, 4932, 4934, 4946, 4948, 4953, 4960, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 13121, 13488, 16804, 17248, 17584, 28709
259	437, 808, 850-852, 855-858, 860-865, 867, 869, 872, 874, 899, 901-902, 915, 1098, 1161-1162, 1200, 1208, 1250-1258, 4946, 4948, 4951-4953, 4960, 4970, 5346, 5348, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584
273	37, 256, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1250, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5346, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
274	500, 1047
275	37, 437, 500, 819, 850, 1047, 1200, 1208, 1252, 4946, 5348, 8229, 13488, 17584, 28709
277	37, 256, 273, 278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
278	37, 256, 273, 277, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
280	37, 256, 273, 277-278, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
281	1047
282	500, 1047, 1200, 1208, 13488, 17584
284	37, 256, 273, 277-278, 280, 285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
285	37, 256, 273, 277-278, 280, 284, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
290	37, 256, 273, 277-278, 280, 284-285, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25473, 25617, 25619, 25664, 28709
293	1200, 1208, 13488, 17584
297	37, 256, 273, 277-278, 280, 284-285, 290, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1100, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
300	301, 941, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
301	300, 941, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
367	37, 256, 273, 277-278, 280, 284, 290, 297, 500, 819, 833, 836, 850, 871, 875, 1009, 1026-1027, 1041, 1088, 1115, 1126, 1200, 1208, 4386, 4929, 4932, 4946, 4971, 5123, 5211, 8229, 8482, 9025, 13121, 13488, 17584, 25617, 25664, 28709
420	37, 256, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 8612, 9044, 9049, 9056, 9238, 13488, 16804, 17248, 17584, 28709
423	37, 256, 273, 277-278, 280, 284-285, 297, 437, 500, 737, 775, 813, 819, 838, 850-852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
424	37, 256, 420, 437, 500, 737, 775, 803, 819, 836, 850, 852, 856-857, 860-865, 916, 1112, 1122, 1200, 1208, 1252, 1255, 4932, 4946, 4948, 4952-4953, 4960, 5012, 5351, 8229, 8612, 9044, 9049, 9056, 13488, 16804, 17248, 17584, 28709
437	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-863, 865-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1025-1027, 1040-1043, 1047, 1051, 1097, 1098, 1114-1115, 1126, 1140-1149, 1200, 1208, 1252, 1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210-5211, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
500	37, 256, 273-275, 277-278, 280, 282, 284-285, 290, 297, 367, 420, 423-424, 437, 737, 775, 813, 819, 833, 836, 838, 850-852, 855-858, 860-866, 869-871, 874-875, 880, 891, 895, 897, 903-905, 912, 914-916, 920-924, 1004, 1009-1021, 1023, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097, 1100-1107, 1112, 1114-1115, 1122, 1124-1126, 1129-1133, 1137, 1140-1149, 1200, 1208, 1250-1258, 1275, 1280-1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5142, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 9238, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664, 28709
720	37, 420, 864, 1200, 1208, 1256, 4960, 8229, 8612, 9056, 13488, 16804, 17248, 17584, 28709
737	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 833, 836, 838, 850, 869-871, 875, 880, 905, 1025-1027, 1097, 1200, 1208, 1252-1253, 1280, 4386, 4909, 4929, 4932, 4934, 4946, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 9061, 13121, 13488, 16804, 17584, 28709
775	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 833, 836, 838, 850, 870-871, 875, 880, 905, 1025-1027, 1097, 1112, 1122, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
803	424, 819, 850, 856, 862, 916, 1200, 1208, 1252, 1255, 4946, 4952, 5012, 13488, 17584
806	1200, 1208, 13488, 17584
808	259, 858-859, 872, 923-924, 1140, 1148, 1153-1154, 1200, 1208, 5347, 5348, 13488, 17584
813	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1253, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
819	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 803, 813, 833, 836, 838, 850, 852, 855, 857-858, 860-861, 863-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1041-1043, 1047, 1051, 1088-1089, 1097, 1098, 1112, 1114, 1122-1123, 1126, 1130, 1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
833	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25617, 25619, 25664, 28709
834	926, 951, 1200, 1208, 1362, 4930, 9026, 13488, 17584
835	927, 947, 1200, 1208, 4931, 9027, 13488, 17584, 21427
836	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 424, 437, 500, 737, 775, 819, 833, 850, 852, 855, 857, 870-871, 875, 903, 1009, 1025-1027, 1040-1043, 1088, 1112, 1114-1115, 1122, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 5210-5211, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25479, 25617, 25619, 25664, 28709
837	928, 1200, 1208, 1380, 1385, 4933, 13488, 17584
838	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
848	924, 1148, 1158, 1200, 1208, 5347, 13488, 17584
849	924, 1148, 1154, 1200, 1208, 5347, 13488, 17584

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
850	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 852, 855-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1040-1043, 1047, 1051, 1088-1089, 1097, 1098, 1100, 1112, 1114, 1122, 1126, 1130, 1132, 1140-1149, 1200, 1208, 1250-1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
851	259, 423, 500, 875, 1200, 1208, 4971, 13488, 17584
852	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
855	37, 259, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 819, 833, 836, 850, 852, 857, 866, 870-871, 878, 880, 912, 915, 1025-1027, 1040-1043, 1088, 1200, 1208, 1250-1252, 1283, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 5123, 5346, 5347, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
856	259, 273, 424, 500, 803, 850, 862, 916, 1200, 1208, 1255, 4946, 4952, 5012, 5351, 13488, 17584
857	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
858	37, 259, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 860-861, 865, 871-872, 901-902, 923-924, 1047, 1051, 1140-1149, 1153-1157, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
859	808, 872, 901-902, 1153-1157, 1160-1162, 1164, 1200, 1208, 13488, 17584
860	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857-858, 861, 863, 865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 923-924, 1025-1027, 1041-1043, 1097, 1140, 1145-1146, 1148, 1200, 1208, 1252, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
861	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857-858, 860, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 923-924, 1025-1027, 1041-1043, 1097, 1148, 1149, 1200, 1208, 1252, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
862	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 803, 833, 838, 850, 856, 870-871, 875, 880, 905, 916, 1025-1027, 1097, 1200, 1208, 1252, 1255, 4386, 4929, 4934, 4946, 4952, 4971, 5012, 5123, 5351, 8229, 8482, 8612, 9025, 9030, 12712, 13121, 13488, 16804, 17584, 28709
863	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857, 860-861, 865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1041-1043, 1051, 1097, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 28709
864	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17248, 17584, 28709
865	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 819, 833, 838, 850, 858, 860, 863, 870-871, 875, 880, 905, 923-924, 1025-1027, 1097, 1142-1143, 1148, 1200, 1208, 1252, 4386, 4929, 4934, 4946, 4971, 5123, 5348, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
866	37, 256, 437, 500, 819, 850, 855, 870, 878, 880, 915, 1025, 1200, 1208, 1251-1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709
867	259, 1153-1155, 1160, 1200, 1208, 4899, 5351, 9048, 12712, 13488, 17584
868	918, 1006, 1200, 1208, 13488, 17584
869	37, 256, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 870-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1254, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
870	37, 256, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-866, 869, 871, 874-875, 880, 897, 903, 912, 915-916, 920, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5346, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
871	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869, 870, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1112, 1122, 1140-1149, 1200, 1208, 1252, 1275, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5348, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
872	259, 808, 858-859, 923-924, 1140-1149, 1153-1155, 1200, 1208, 5347, 5348, 13488, 17584
874	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
875	37, 256, 273, 277-278, 280, 284-285, 297, 367, 423, 437, 500, 737, 775, 813, 819, 836, 838, 850-852, 857, 860-865, 869-871, 874, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4932, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
878	855, 866, 880, 915, 1025, 1131, 1200, 1208, 1251, 1283, 4951, 5347, 13488, 17584
880	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 855, 857, 860-866, 869-871, 874-875, 878, 897, 903, 912, 915-916, 920, 1009, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1251-1252, 1283, 4909, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5347, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
891	500, 833, 1088, 1200, 1208, 4929, 9025, 13121, 13488, 17584, 25664
895	290, 500, 1027, 1041, 1200, 1208, 4386, 5123, 8482, 13488, 17584, 25617
896	290, 1027, 1041, 1200, 1208, 4386, 4992, 5123, 8482, 13488, 17584, 25617
897	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4386, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
899	259
901	259, 858-859, 902, 923-924, 1140, 1148, 1156-1157, 1200, 1208, 5348, 5353, 13488, 17584
902	259, 858-859, 901, 923-924, 1140, 1148, 1156-1157, 1200, 1208, 5348, 5353, 13488, 17584

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
903	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 836, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 912, 916, 920, 1025-1027, 1041-1043, 1115, 1200, 1208, 1252, 4909, 4932, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5211, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
904	37, 500, 1114, 1200, 1208, 5210, 8229, 13488, 17584, 25480, 28709
905	37, 256, 437, 500, 737, 775, 819, 850, 852, 857, 860-865, 920, 1026, 1112, 1122, 1200, 1208, 1252, 1254, 1281, 4946, 4948, 4953, 4960, 8229, 9044, 9049, 9056, 13488, 17248, 17584, 28709
912	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 916, 920, 1025-1027, 1041-1043, 1047, 1200, 1208, 1250, 1252, 1282, 4909, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
914	37, 437, 500, 819, 850, 1200, 1208, 1252, 1257, 4946, 8229, 13488, 17584, 28709
915	37, 259, 437, 500, 819, 850, 855, 866, 870, 878, 880, 1025, 1131, 1200, 1208, 1251-1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709
916	37, 273, 277-278, 280, 284-285, 297, 423-424, 437, 500, 803, 813, 819, 838, 850, 852, 856-857, 860-863, 869-871, 874-875, 880, 897, 903, 912, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 1255, 4909, 4934, 4946, 4948, 4952-4953, 4970-4971, 5012, 5123, 5351, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
918	864, 868, 1006, 1200, 1208, 4960, 9056, 13488, 17248, 17584
920	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 1025-1026, 1200, 1208, 1252, 1254, 1281, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5350, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 28709
921	37, 437, 500, 819, 850, 922, 1112, 1122, 1200, 1208, 1252, 1257, 4946, 5353, 8229, 13488, 17584, 28709
922	37, 437, 500, 819, 850, 921, 1112, 1122, 1200, 1208, 1252, 1257, 4946, 5353, 8229, 13488, 17584, 28709
923	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860-861, 865, 871-872, 901-902, 924, 1047, 1051, 1140-1149, 1153-1158, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
924	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 848-850, 858, 860-861, 865, 871-872, 901-902, 923, 1047, 1051, 1140-1149, 1153-1157, 1160-1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
926	834, 951, 9026
927	835, 947, 1200, 1208, 4931, 9027, 13488, 17584, 21427
928	837, 1200, 1208, 1380, 13488, 17584

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
930	931-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
931	930, 932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
932	930-931, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
933	934, 944, 949, 1200, 1208, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25510, 25520, 25525, 29616, 29621, 33717, 37813
934	933, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25510, 25525, 29621, 33717, 37813
935	936, 946, 1200, 1208, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584, 25512
936	935, 946, 1381, 5031, 5477, 5484, 9127, 13223, 25512
937	938, 948, 950, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
938	937, 950, 1370, 5033, 5046, 9142, 25514
939	930-932, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
941	300-301, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
942	930-932, 939, 943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
943	930-932, 939, 942, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
944	933, 949, 1200, 1208, 5029, 5045, 5460, 9125, 13221, 13488, 17317, 17584, 25520, 25525, 29616, 29621, 33717, 37813
946	935-936, 1200, 1208, 5031, 5484, 9127, 13223, 13488, 17584, 25512
947	835, 927, 1200, 1208, 4931, 9027, 13488, 17584, 21427
948	937, 950, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25524, 29620
949	933-934, 944, 1200, 1208, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25510, 25520, 25525, 29616, 29621, 33717, 37813
950	937-938, 948, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
951	834, 926, 1200, 1208, 1362, 4930, 9026, 13488, 17584

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
1004	500, 819, 850, 1200, 1208, 4946, 13488, 17584
1006	868, 918, 1200, 1208, 13488, 17584
1008	420, 864, 1200, 1208, 4960, 5104, 8612, 9056, 13488, 16804, 17248, 17584
1009	37, 273, 277-278, 280, 284, 290, 297, 367, 423, 500, 833, 836, 870-871, 875, 880, 1025-1026, 1200, 1208, 4386, 4929, 4932, 4971, 8229, 8482, 9025, 13121, 13488, 17584, 28709
1010	500, 1200, 1208, 13488, 17584
1011	500, 1200, 1208, 13488, 17584
1012	500, 1200, 1208, 13488, 17584
1013	500, 1140, 1200, 1208, 13488, 17584
1014	500, 1200, 1208, 13488, 17584
1015	500, 1200, 1208, 13488, 17584
1016	500, 1200, 1208, 13488, 17584
1017	500, 1200, 1208, 13488, 17584
1018	500, 1200, 1208, 13488, 17584
1019	500, 1200, 1208, 13488, 17584
1020	500
1021	500
1023	500
1025	37, 256, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-866, 869-871, 874-875, 878, 880, 897, 903, 912, 915-916, 920, 1009, 1026-1027, 1040-1043, 1051, 1088, 1112, 1122, 1131, 1200, 1208, 1251-1252, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5347, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1026	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1009, 1025, 1027, 1040-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5350, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1027	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-865, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1026, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
1040	37, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 833, 836, 850, 852, 855, 857, 870-871, 1025-1027, 1041-1043, 1088, 1200, 1208, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
1041	37, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1027, 1040, 1042-1043, 1088, 1200, 1208, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1042	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040, 1041, 1043, 1088, 1200, 1208, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1043	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040, 1041, 1042, 1088, 1114, 1200, 1208, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 17584, 25473, 25479, 25617, 25619, 25664, 28709
1046	420, 500, 864, 1089, 1127, 1200, 1208, 1256, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1047	37, 273-275, 277-278, 280, 281, 282, 284-285, 290, 297, 437, 500, 819, 850, 852, 858, 870-871, 875, 912, 923-924, 1026-1027, 1140-1149, 1200, 1208, 1252, 1254, 4946, 4948, 5123, 8229, 8482, 13488, 17584, 28709
1051	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871, 923-924, 1025, 1097, 1140-1149, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1088	37, 273, 277-278, 280, 284-285, 290, 297, 367, 500, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 891, 1025-1027, 1040-1043, 1126, 1200, 1208, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
1089	420, 500, 819, 850, 864, 1046, 1127, 1200, 1208, 1256, 4946, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1097	37, 437, 500, 737, 775, 819, 850, 852, 857, 860-865, 1051, 1098, 1112, 1122, 1200, 1208, 1252, 4946, 4948, 4953, 4960, 8229, 9044, 9049, 9056, 13488, 17248, 17584, 28709
1098	259, 420, 437, 819, 850, 1097, 1200, 1208, 1252, 4946, 8612, 13488, 16804, 17584
1100	37, 273, 277-278, 280, 284-285, 297, 500, 850, 4946, 8229, 28709
1101	500
1102	500

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
1103	500
1104	500
1105	500
1106	500
1107	500
1112	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 775, 819, 833, 836, 838, 850, 870-871, 875, 880, 905, 921-922, 1025-1027, 1097, 1122, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 5353, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
1114	37, 437, 500, 819, 836, 850, 904, 1043, 1115, 1200, 1208, 4932, 4946, 5210-5211, 8229, 13488, 17584, 25480, 25619, 28709
1115	37, 367, 437, 500, 836, 903, 1114, 1200, 1208, 4932, 5210-5211, 8229, 13488, 17584, 25479, 28709
1122	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 775, 819, 833, 836, 838, 850, 870-871, 875, 880, 905, 921-922, 1025-1027, 1097, 1112, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 5353, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804, 17584, 28709
1123	819, 1124-1125, 1148, 1200, 1208, 1251-1252, 1283, 5347, 13488, 17584
1124	37, 500, 1123, 1125, 1200, 1208, 1251, 1283, 5347, 8229, 13488, 17584, 28709
1125	500, 1123, 1124, 1200, 1208, 1251, 1283, 5347, 13488, 17584
1126	37, 367, 437, 500, 819, 833, 850, 1088, 1200, 1208, 1252, 4929, 4946, 8229, 9025, 13121, 13488, 17584, 25664, 28709
1127	420, 864, 1046, 1089, 1256, 4960, 5142, 8612, 9056, 9238, 16804, 17248
1129	500, 1130, 1200, 1208, 1258, 5354, 13488, 17584
1130	37, 500, 819, 850, 1129, 1200, 1208, 1252, 1258, 4946, 5354, 8229, 13488, 17584, 28709
1131	37, 500, 878, 915, 1025, 1200, 1208, 1251, 1283, 5347, 8229, 13488, 17584, 28709
1132	37, 500, 819, 850, 1133, 1200, 1208, 1252, 4946, 8229, 13488, 17584, 28709
1133	500, 1132, 1200, 1208, 13488, 17584
1137	37, 500, 819, 1200, 1208, 8229, 13488, 17584, 28709
1139	290, 1027, 4386, 5123, 8482
1140	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860, 863, 871-872, 901-902, 923-924, 1013, 1047, 1051, 1141-1149, 1153-1157, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
1141	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140, 1142-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1142	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 865, 871-872, 923-924, 1047, 1051, 1140-1141, 1143-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1143	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 865, 871-872, 923-924, 1047, 1051, 1140-1142, 1144-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1144	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140-1143, 1145-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1145	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 860, 863, 871-872, 923-924, 1047, 1051, 1140-1144, 1146-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1146	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 860, 863, 871-872, 923-924, 1047, 1051, 1140-1145, 1147-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1147	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140-1146, 1148-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1148	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 848-850, 858, 860-861, 863, 865, 871-872, 901-902, 923-924, 1047, 1051, 1123, 1140-1147, 1149, 1153-1164, 1200, 1208, 1252, 1275, 4899, 4946, 5348, 5349, 8229, 12712, 13488, 17584, 28709
1149	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 861, 863, 871-872, 923-924, 1047, 1051, 1140-1148, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1153	808, 858-859, 867, 872, 923-924, 1140-1149, 1154-1157, 1160-1162, 1200, 1208, 5348, 9044, 13488, 17584
1154	808, 849, 858-859, 867, 872, 923-924, 1140-1149, 1153, 1155-1157, 1160-1162, 1200, 1208, 5347, 5348, 13488, 17584
1155	858-859, 867, 872, 923-924, 1140-1149, 1153-1154, 1156-1157, 1160-1162, 1200, 1208, 5348, 5350, 9049, 13488, 17584
1156	858-859, 901-902, 923-924, 1140-1149, 1153-1155, 1157, 1160, 1200, 1208, 5348, 5353, 12712, 13488, 17584
1157	858-859, 901-902, 923-924, 1140-1149, 1153-1156, 1160, 1200, 1208, 5348, 5353, 12712, 13488, 17584
1158	848, 923, 1148, 1200, 1208, 5347, 5348, 13488, 17584

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
1159	1148, 1200, 1208, 13488, 17584
1160	858-859, 867, 923-924, 1140-1149, 1153-1157, 1161-1162, 1200, 1208, 5348, 13488, 17584
1161	259, 858-859, 923-924, 1140-1149, 1153-1155, 1160, 5348, 17584
1162	259, 858-859, 923-924, 1140-1149, 1153-1155, 1160, 5348, 17584
1163	924, 1148, 1164, 5354, 17584
1164	858-859, 923-924, 1140, 1148, 1163, 1200, 1208, 5348, 5354, 13488, 17584
1166	1200,1208,13488,17584
1200	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1374-1379, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 17584, 21427, 28709
1208	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1374-1379, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5026, 5035, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 17584, 21427, 28709
1250	37, 259, 273, 500, 819, 850, 852, 855, 870, 912, 1200, 1208, 1252, 1282, 4946, 4948, 4951, 5346, 8229, 9044, 13488, 17584, 28709
1251	37, 256, 259, 500, 819, 850, 855, 866, 878, 880, 915, 1025, 1123-1125, 1131, 1200, 1208, 1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
1252	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1025-1027, 1041, 1047, 1051, 1097-1098, 1112, 1122-1123, 1126, 1130, 1132, 1140-1149, 1200, 1208, 1250-1251, 1254-1255, 1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 28709
1253	37, 259, 423, 500, 737, 813, 819, 850, 869, 875, 1200, 1208, 1280, 4909, 4946, 4971, 5349, 8229, 9061, 13488, 17584, 28709
1254	37, 259, 500, 819, 850, 857, 869, 905, 920, 1026, 1047, 1200, 1208, 1252, 1281, 4946, 4953, 5350, 8229, 9049, 9061, 13488, 17584, 28709
1255	37, 259, 424, 500, 803, 819, 850, 856, 862, 916, 1200, 1208, 1252, 1281, 4946, 4952, 5012, 5351, 8229, 13488, 17584, 28709
1256	259, 420, 500, 720, 850, 864, 1046, 1089, 1127, 1200, 1208, 4946, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1257	37, 259, 437, 500, 775, 819, 850, 914, 921-922, 1112, 1122, 1200, 1208, 1252, 4946, 5353, 8229, 13488, 17584, 28709
1258	37, 259, 500, 819, 1129-1130, 1200, 1208, 5354, 8229, 13488, 17584, 28709
1275	37, 256, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871, 923-924, 1051, 1140-1149, 1200, 1208, 1252, 4946, 5348, 8229, 13488, 17584, 28709
1276	1200, 1208, 13488, 17584
1277	1200, 1208, 13488, 17584
1280	37, 423, 437, 500, 737, 813, 819, 850, 869, 875, 1200, 1208, 1252-1253, 4909, 4946, 4971, 5349, 8229, 9061, 13488, 17584, 28709
1281	37, 437, 500, 819, 850, 857, 905, 920, 1026, 1200, 1208, 1252, 1254-1255, 4946, 4953, 5350, 8229, 9049, 13488, 17584, 28709
1282	500, 852, 870, 912, 1200, 1208, 1250, 4948, 5346, 9044, 13488, 17584
1283	37, 437, 500, 819, 850, 855, 866, 878, 880, 915, 1025, 1123-1125, 1131, 1200, 1208, 1251-1252, 4946, 4951, 5347, 8229, 13488, 17584, 28709
1284	1200, 1208, 13488, 17584
1285	1200, 1208, 13488, 17584
1351	300-301, 941, 1200, 1208, 4396, 8492, 13488, 16684, 17584
1362	834, 951, 1200, 1208, 4930, 9026, 13488, 17584
1363	933, 949, 1200, 1208, 1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25525, 29621, 33717, 37813
1364	933, 949, 1200, 1208, 1363, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25525, 29621, 33717, 37813

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
1370	937-938, 948, 950, 1200, 1208, 1371, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
1371	1200, 1208, 1370, 13488, 17584
1374	1200, 1208
1375	1200, 1208
1376	1200, 1208
1377	1200, 1208
1378	1200, 1208
1379	1200, 1208
1380	837, 928, 1200, 1208, 1385, 4933, 13488, 17584
1381	935-936, 1200, 1208, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584, 25512
1385	837, 1200, 1208, 1380, 4933, 13488, 17584
1386	935, 1200, 1208, 1381, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584
1388	935, 1200, 1208, 1381, 1386, 5031, 5477, 5482, 5484, 5488, 9127, 13223, 13488, 17584
1390	930-932, 939, 942-943, 1200, 1208, 1399, 5026, 5028, 5035, 5038-5039, 5055, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
1399	930-932, 939, 942-943, 1200, 1208, 1390, 5026, 5028, 5035, 5038-5039, 5050, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
4386	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1139, 1252, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 17248, 25473, 25617, 25619, 25664, 28709
4396	300-301, 941, 1351, 8492, 16684
4899	867, 1148, 1200, 1208, 5351, 9048, 12712, 13488, 17584
4909	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1253, 1280, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
4929	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1252, 4386, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13121, 17248, 25617, 25619, 25664, 28709
4930	834, 951, 1200, 1208, 1362, 9026, 13488, 17584
4931	835, 927, 947, 9027, 21427

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
4932	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 424, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 903, 1009, 1025-1027, 1040-1043, 1088, 1112, 1114-1115, 1122, 1252, 4386, 4929, 4946, 4948, 4951, 4953, 4971, 5123, 5210-5211, 8229, 8482, 9025, 9044, 9049, 13121, 25479, 25617, 25619, 25664, 28709
4933	837, 1200, 1208, 1380, 1385, 13488, 17584
4934	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1252, 4909, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 17248, 25473, 25479, 25617, 25619, 28709
4946	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 850, 852, 855-858, 860-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1040-1043, 1047, 1051, 1088-1089, 1097-1098, 1100, 1112, 1114, 1122, 1126, 1130, 1132, 1140-1149, 1250-1257, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4948, 4951-4953, 4960, 4970-4971, 5012, 5123, 5210, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 16804, 17248, 25473, 25479, 25617, 25619, 25664, 28709
4948	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
4951	37, 259, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 819, 833, 836, 850, 852, 855, 857, 866, 870-871, 878, 880, 912, 915, 1025-1027, 1040-1043, 1088, 1200, 1208, 1250-1252, 1283, 4386, 4929, 4932, 4946, 4948, 4953, 5123, 5346, 5347, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 28709
4952	259, 273, 424, 500, 803, 850, 856, 862, 916, 1200, 1208, 1255, 4946, 5012, 5351, 13488, 17584
4953	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 16804, 25473, 25479, 25617, 25619, 25664, 28709
4960	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17248, 17584, 28709

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
4970	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1252, 4909, 4934, 4946, 4948, 4953, 4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 9066, 25473, 25479, 25617, 25619, 28709
4971	37, 256, 273, 277-278, 280, 284-285, 297, 367, 423, 437, 500, 737, 775, 813, 819, 836, 838, 850-852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4932, 4934, 4946, 4948, 4953, 4960, 4970, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
4992	290, 896, 1027, 1041, 4386, 5123, 8482, 25617
5012	37, 273, 277-278, 280, 284-285, 297, 423-424, 437, 500, 803, 813, 819, 838, 850, 852, 856-857, 860-863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 1255, 4909, 4934, 4946, 4948, 4952-4953, 4970-4971, 5123, 5351, 8229, 9030, 9044, 9049, 9061, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
5026	930-932, 939, 942-943, 1390, 1399, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 1208, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5028	930-932, 939, 942-943, 1390, 1399, 5026, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5029	933-934, 944, 949, 1363-1364, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5031	935-936, 946, 1381, 1386, 1388, 5477, 5482, 5484, 9127, 13223, 25512
5033	937-938, 948, 950, 1370, 5046, 9142, 25514, 25524, 29620
5035	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5038-5039, 9122, 9124, 9131, 9135, 1208, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5038	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5039	930-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
5045	933-934, 944, 949, 1363-1364, 5029, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5046	937-938, 948, 950, 1370, 5033, 9142, 25514, 25524, 29620
5104	420, 864, 1008, 1200, 1208, 4960, 8612, 9056, 13488, 16804, 17248, 17584
5123	290, 367, 423, 437, 819, 1027, 1041, 1047, 1140-1149, 1156, 1157, 1160, 1200, 1208, 1252, 4948, 5348, 8482, 13488

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
5142	420, 500, 864, 1046, 1089, 1127, 1200, 1208, 1256, 4960, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
5210	37, 437, 500, 819, 836, 850, 904, 1043, 1114-1115, 1200, 1208, 4932, 4946, 5211, 8229, 13488, 17584, 25480, 25619, 28709
5211	37, 367, 437, 500, 836, 903, 1114-1115, 4932, 5210, 8229, 25479, 28709
5346	37, 259, 273, 500, 819, 850, 852, 855, 870, 912, 1200, 1208, 1250, 1252, 1282, 4946, 4948, 4951, 8229, 9044, 13488, 17584, 28709
5347	808, 848-849, 855, 866, 872, 878, 880, 915, 1025, 1123-1125, 1131, 1154, 1158, 1200, 1208, 1251, 1283, 4951, 13488, 17584
5348	37, 259, 273, 275, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860-861, 863, 865, 871-872, 901-902, 923-924, 1051, 1140-1149, 1153-1158, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 8229, 13488, 17584, 28709
5349	813, 869, 875, 1148, 1200, 1208, 1253, 1280, 4909, 4971, 9061, 13488, 17584
5350	857, 920, 1026, 1155, 1200, 1208, 1254, 1281, 4953, 9049, 13488, 17584
5351	424, 856, 862, 867, 916, 1200, 1208, 1255, 4899, 4952, 5012, 9048, 12712, 13488, 17584
5352	420, 864, 1046, 1089, 1200, 1208, 1256, 4960, 5142, 8612, 9056, 9238, 13488, 16804, 17248, 17584
5353	901-902, 921-922, 1112, 1122, 1156-1157, 1200, 1208, 1257, 13488, 17584
5354	1129-1130, 1163, 1164, 1200, 1208, 1258, 13488, 17584
5460	933-934, 944, 949, 1363-1364, 5029, 5045, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5477	935-936, 1381, 1386, 1388, 5031, 5482, 5484, 9127, 13223, 25512
5482	935, 1381, 1386, 1388, 5031, 5477, 5484, 9127, 13223
5484	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 9127, 13223, 25512
5488	1388
8229	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 16804, 17248, 25473, 25479, 25480, 25617, 25619, 25664, 28709

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
8482	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 8229, 9025, 9044, 9049, 9056, 13121, 13488, 17248, 17584, 25473, 25617, 25619, 25664, 28709
8492	300-301, 941, 1351, 4396, 16684
8612	37, 256, 420, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 9044, 9049, 9056, 9238, 13488, 16804, 17248, 17584, 28709
9025	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9044, 9049, 9056, 13121, 17248, 25617, 25619, 25664, 28709
9026	834, 926, 951, 1362, 4930
9027	835, 927, 947, 1200, 1208, 4931, 13488, 17584, 21427
9030	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4960, 4970-4971, 5012, 5123, 8229, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584, 25473, 25479, 25617, 25619, 28709
9044	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1153, 1200, 1208, 1250, 1252, 1282, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5346, 8229, 8482, 8612, 9025, 9030, 9049, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
9048	867, 1200, 1208, 4899, 5351, 12712, 13488, 17584
9049	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1155, 1200, 1208, 1252, 1254, 1281, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5350, 8229, 8482, 8612, 9025, 9030, 9044, 9061, 9066, 13121, 13488, 16804, 17584, 25473, 25479, 25617, 25619, 25664, 28709
9056	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9238, 13121, 13488, 16804, 17248, 17584, 28709

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
9061	37, 256, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1254, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5349, 8229, 9030, 9044, 9049, 9066, 13488, 17584, 25473, 25479, 25617, 25619, 28709
9066	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 9030, 9044, 9049, 9061, 13488, 17584, 25473, 25479, 25617, 25619, 28709
9122	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9124	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9125	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
9127	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 13223, 25512
9131	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9135	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9142	937-938, 948, 950, 1370, 5033, 5046, 25514, 25524, 29620
9238	420, 500, 864, 1046, 1089, 1127, 1200, 1208, 1256, 4960, 5142, 5352, 8612, 9056, 13488, 16804, 17248, 17584
9555	933, 949, 1363-1364, 5029, 5045, 5460, 9125, 13221, 13651, 17317, 25525, 29621, 33717, 37813
12712	862, 867, 1148, 1156-1157, 1200, 1208, 4899, 5351, 9048, 13488, 17584
13121	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025, 9044, 9049, 9056, 13488, 17248, 17584, 25617, 25619, 25664, 28709
13218	930-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
13219	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
13221	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
13223	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 25512
13231	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 17314, 25508, 25518, 29614, 33698-33700, 37796
13488	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 16684, 16804, 17248, 17584, 21427, 28709
13651	933, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 17317, 25525, 29621, 33717, 37813
16684	300-301, 941, 1200, 1208, 1351, 4396, 8492, 13488, 17584
16804	37, 256, 420, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 8612, 9044, 9049, 9056, 9238, 13488, 17248, 17584, 28709
17248	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 5352, 8229, 8482, 8612, 9025, 9030, 9056, 9238, 13121, 13488, 16804, 17584, 28709
17314	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 25508, 25518, 29614, 33698-33700, 37796
17317	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 25510, 25520, 25525, 29616, 29621, 33717, 37813
17584	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927-928, 930, 932-933, 935, 937, 939, 941-944, 946-951, 1004, 1006, 1008-1019, 1025-1027, 1040-1043, 1046-1047, 1051, 1088-1089, 1097-1098, 1112, 1114-1115, 1122-1126, 1129-1133, 1137, 1140-1149, 1153-1160, 1164, 1166, 1200, 1208, 1250-1258, 1275-1277, 1280-1285, 1351, 1362-1364, 1370-1371, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 12712, 13121, 13218, 13488, 16684, 16804, 17248, 21427, 28709
21427	835, 927, 947, 1200, 1208, 4931, 9027, 13488, 17584

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
25473	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1252, 4386, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 8229, 8482, 9030, 9044, 9049, 9061, 9066, 25479, 25617, 25619, 28709
25479	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 836, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1115, 1252, 4909, 4932, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5211, 8229, 9030, 9044, 9049, 9061, 9066, 25473, 25617, 25619, 28709
25480	37, 500, 904, 1114, 5210, 8229, 28709
25508	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25518, 29614, 33698-33700, 37796
25510	933-934, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25525, 29621, 33717, 37813
25512	935-936, 946, 1381, 5031, 5477, 5484, 9127, 13223
25514	937-938, 950, 1370, 5033, 5046, 9142
25518	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 29614, 33698-33700, 37796
25520	933, 944, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25525, 29616, 29621, 33717, 37813
25524	937, 948, 950, 1370, 5033, 5046, 9142, 29620
25525	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 29616, 29621, 33717, 37813
25617	37, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1027, 1040-1043, 1088, 1252, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 4992, 5012, 5123, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 25473, 25479, 25619, 25664, 28709
25619	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040-1043, 1088, 1114, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4970-4971, 5012, 5123, 5210, 8229, 8482, 9025, 9030, 9044, 9049, 9061, 9066, 13121, 25473, 25479, 25617, 25664, 28709
25664	37, 273, 277-278, 280, 284-285, 290, 297, 367, 500, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 891, 1025-1027, 1040-1043, 1088, 1126, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 25617, 25619, 28709

Table 674. IBM MQ for z/OS CCSID conversion support (continued)	
CCSID	**Converts to and from CCSIDS**
28709	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1051, 1088, 1097, 1100, 1112, 1114-1115, 1122, 1124, 1126, 1130-1132, 1137, 1140-1149, 1200, 1208, 1250-1255, 1257-1258, 1275, 1280-1281, 1283, 4386, 4909, 4929, 4932, 4934, 4946, 4948, 4951, 4953, 4960, 4970-4971, 5012, 5123, 5210-5211, 5346, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664
29614	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 33698-33700, 37796
29616	933, 944, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25520, 25525, 29621, 33717, 37813
29620	937, 948, 950, 1370, 5033, 5046, 9142, 25524
29621	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 33717, 37813
33698	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33699-33700, 37796
33699	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698, 33700, 37796
33700	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33699, 37796
33717	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 37813
37796	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700
37813	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717

IBM i Supporto di conversione IBM i

Un elenco completo dei CCSID e delle conversioni supportate da IBM iè disponibile nella pubblicazione IBM i appropriata.

Le codepage supportate sono elencate in Associazioni CCSID supportate.

Supporto conversione Unicode

Alcune piattaforme supportano la conversione dei dati utente in o dalla codifica Unicode. Le due forme di codifica Unicode supportate sono UTF-16 (CCSID 1200, 13488 e 17584) e UTF-8 (CCSID 1208). È necessario utilizzare i CCSID 1200 o 1208, poiché rappresentano la versione Unicode più recente supportata.

Sono supportate le coppie di surrogati UTF-16 (una coppia di caratteri UTF-16 a 2 byte nell'intervallo da X'D800'a X'DFFF' che rappresentano un punto di codice Unicode superiore a U + FFFF). Se un CCSID di destinazione non contiene una corrispondenza per un punto di codice rappresentato da una coppia di surrogati UTF-16 , la coppia di caratteri viene convertita in un singolo carattere di sostituzione.

La combinazione di sequenze di caratteri è supportata da IBM MQ. Ciò significa che, in alcuni casi, un carattere precomposto nel CCSID di origine verrà convertito in una sequenza di caratteri di combinazione nel CCSID di destinazione o viceversa.

Nota: IBM MQ non supporta i CCSID del gestore code UTF-16 , pertanto i dati di intestazione del messaggio non possono essere codificati in UTF-16.

Supporto IBM MQ AIX per Unicode

AIX

Su IBM MQ for AIX la conversione in e da CCSID Unicode supportati (preferibilmente 1200 o 1208) è supportata per i CCSID non Unicode nel seguente elenco:

037
273, 278, 280, 284, 285, 297
423, 437
500
813, 819, 850, 852, 856, 857, 858, 860, 861, 865, 867, 869, 875, 878, 880
901, 902, 912, 915, 916, 920, 923, 924, 932, 933, 935, 937, 938, 939, 942, 943, 948, 949, 950, 954, 964, 970
1026, 1046, 1089
1129, 1130, 1131, 1132, 1133, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1153, 1156, 1157
1200, 1208, 1250, 1251, 1253, 1254, 1258, 1280, 1281, 1282, 1283, 1284, 1285
1363, 1364, 1381, 1383, 1386, 1388
4899
5026, 5035, 5050, 5346, 5347, 5348, 5349, 5350, 5351, 5352, 5353, 5354, 5488
9044, 9048, 9449
12712
13488
17584
33722

Supporto IBM MQ for Windows e Linux per Unicode

Linux Windows

Su IBM MQ for Windows e IBM MQ per la conversione da e verso Linux i CCSID Unicode supportati (preferibilmente 1200 o 1208) sono supportati per i CCSID non Unicode nel seguente elenco:

037,
277, 278, 280, 284, 285, 290, 297
300, 301
420, 424, 437
500
813, 819, 833, 835, 836, 837, 838, 850, 852, 855, 856, 857, 858, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 874, 875, 878, 880, 891, 897
901, 902, 904, 912, 913"5" a pagina 1017, 915, 916, 918, 920, 921, 922, 923, 924, 927, 928, 930, 931"1" a pagina 1017, 932"2" a pagina 1017, 933, 935, 937, 938"3" a pagina 1017, 939, 941, 942, 943, 947, 948, 949, 950, 951, 954"4" a pagina 1017, 964, 970
1006, 1025, 1026, 1027, 1040, 1041, 1042, 1043, 1046, 1047, 1051, 1088, 1089, 1097, 1098

1112, 1114, 1115, 1122, 1123, 1124, 1129, 1130, 1132, 1133, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1153, 1156, 1157
1200, 1208, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1275, 1280, 1281, 1282, 1283
1363, 1364, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 1383, 1386, 1388
4899
5050, 5346, 5347, 5348, 5349, 5350, 5351, 5352, 5353, 5354, 5488"5" a pagina 1017
9044, 9048, 9449
12712
13488
17584
33722"4" a pagina 1017

Note:

1. 931 utilizza 939 per la conversione.
2. 932 utilizza 942 per la conversione.
3. 938 utilizza 948 per la conversione.
4. 954 e 33722 utilizzano 5050 per la conversione.
5. Solo su Windows e Linux .

Supporto IBM i per Unicode

Per i dettagli sul supporto UNICODE, fare riferimento alla pubblicazione IBM i appropriata relativa al sistema operativo.

Supporto IBM MQ for z/OS per Unicode

Su IBM MQ for z/OS la conversione in e da CCSID Unicode supportati (preferibilmente 1200 o 1208) è supportata per i CCSID non Unicode nel seguente elenco:

280
256, 259, 273, 275, 277, 278, 280, 282, 284, 285, 290, 293, 297
300, 301, 367
420, 423, 424, 437
500
720, 737, 775
803, 806, 808, 813, 819, 833, 834, 835, 836, 837, 838, 848, 849, 850, 851, 852, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 874, 875, 878, 880, 891, 895, 896, 897
901, 902, 903, 904, 905, 912, 914, 915, 916, 918, 920, 921, 922, 923, 924, 927, 928, 930, 932, 933, 935, 937, 939, 941, 942, 943, 944, 946, 947, 948, 949, 950, 951
1004, 1006, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1025, 1026, 1027, 1040, 1041, 1042, 1043, 1046, 1047, 1051, 1088, 1089, 1097, 1098
1112, 1114, 1115, 1122, 1123, 1124, 1125, 1126, 1129, 1130, 1131, 1132, 1133, 1137, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1164
1200, 1208, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1275, 1276, 1277, 1280, 1281, 1282, 1283, 1284, 1285
1351, 1362, 1363, 1364, 1370, 1371, 1380, 1381, 1385, 1386, 1388, 1390, 1399
4899, 4909, 4930, 4933, 4948, 4951, 4952, 4960, 4971
5012 5039 5104 5123 5142 5210 53465347 5348 5349 5350 5351 5352 5353 5354 5488

8482 8612
9027 9030 9044 9048 9049 9056 9061 9066 9238 9449
1166
12712
13121, 13218, 13488, 1374, 1375, 1376, 1377, 1378, 1379
16684, 16804
17248, 17584
21427
28709

Standard di codificazione su piattaforme a 64 bit

Utilizzare queste informazioni per informazioni sugli standard di codifica sulle piattaforme a 64 bit e sui tipi di dati preferiti.

Tipi di dati preferiti

Questi tipi non cambiano mai dimensione e sono disponibili su piattaforme IBM MQ a 32 bit e a 64 bit:

Tabella 675. Nomi e lunghezze dei tipi di dati

Nome	Lunghezza
MQLONG	4 byte
MQULONG	4 byte
MQINT32	4 byte
MQUINT32	4 byte
MQINT64	8 byte
MQUINT64	8 byte

ALW Tipi di dati standard su AIX, Linux, and Windows

Informazioni sui tipi di dati standard su applicazioni AIX and Linuxa 32 bit e AIX, Linux, and Windows a 64 bit.

Applicazioni AIX and Linux a 32 bit

Linux AIX

Tabella 676. Nomi e lunghezze dei tipi di dati per applicazioni AIX and Linux a 32 bit

Nome	Lunghezza
char	1 byte
breve	2 byte
int	4 byte
lungo	4 byte
mobile	4 byte
doppio	8 byte
doppio lungo	8 byte
puntatore	4 byte
ptrdiff_t	4 byte

Tabella 676. Nomi e lunghezze dei tipi di dati per applicazioni AIX and Linux a 32 bit (Continua)

Nome	Lunghezza
dimensione_t	4 byte
ora_t	4 byte
clock_t	4 byte
wchar_t	4 byte
	AIX Notare che su AIX un wchar_t è di 2 byte.

Applicazioni AIX and Linux a 64 bit

Linux AIX

Tabella 677. Nomi e lunghezze del tipo di dati per applicazioni AIX and Linux a 64 bit

Nome	Lunghezza
char	1 byte
breve	2 byte
int	4 byte
lungo	8 byte
mobile	4 byte
doppio	8 byte
doppio lungo	8 byte
puntatore	8 byte
ptrdiff_t	8 byte
dimensione_t	8 byte
ora_t	8 byte
clock_t	4 byte
wchar_t	4 byte
	AIX Notare che su AIX un wchar_t è di 2 byte.

Applicazioni Windows a 64 bit

Windows

Tabella 678. Nomi e lunghezze dei tipi di dati per applicazioni Windows a 64 bit

Nome	Lunghezza
char	1 byte
breve	2 byte
int	4 byte
lungo	4 byte
mobile	4 byte
doppio	8 byte

Tabella 678. Nomi e lunghezze dei tipi di dati per applicazioni Windows a 64 bit (Continua)

Nome	Lunghezza
doppio lungo	8 byte
puntatore	8 byte Notare che tutti i puntatori sono di 8 byte.
ptrdiff_t	8 byte
dimensione_t	8 byte
ora_t	8 byte
clock_t	4 byte
wchar_t	2 byte
Parola	2 byte
DWORD	4 byte
applic.	8 byte
FILE	4 byte

Considerazioni sulla codifica su Windows

Windows

HANDLE hf;

Utilizzo

```
hf = CreateFile((LPCTSTR) FileName,
                Access,
                ShareMode,
                xihSecAttsNTRestrict,
                Create,
                AttrAndFlags,
                NULL);
```

Non utilizzare

```
HFILE hf;
hf = (HFILE) CreateFile((LPCTSTR) FileName,
                        Access,
                        ShareMode,
                        xihSecAttsNTRestrict,
                        Create,
                        AttrAndFlags,
                        NULL);
```

poiché ciò produce un errore.

size_t len fget

Utilizzo

```
size_t len
while (fgets(string1, (int) len, fp) != NULL)
len = strlen(buffer);
```

Non utilizzare

```
int len;
```

```
while (fgets(string1, len, fp) != NULL)
len = strlen(buffer);
```

printf

Utilizzo

```
printf("My struc pointer: %p", pMyStruc);
```

Non utilizzare

```
printf("My struc pointer: %x", pMyStruc);
```

Se è necessario un output esadecimale, è necessario stampare separatamente i 4 byte superiori e inferiori.

char * ptr

Utilizzo

```
char * ptr1;
char * ptr2;
size_t bufLen;

bufLen = ptr2 - ptr1;
```

Non utilizzare

```
char *ptr1;
char *ptr2;
UINT32 bufLen;

bufLen = ptr2 - ptr1;
```

alignBytes

Utilizzo

```
alignBytes = (unsigned short) ((size_t) address % 16);
```

Non utilizzare

```
void *address;
unsigned short alignBytes;

alignBytes = (unsigned short) ((UINT32) address % 16);
```

len

Utilizzo

```
len = (UINT32) ((char *) address2 - (char *) address1);
```

Non utilizzare

```
void *address1;
void *address2;
UINT32 len;

len = (UINT32) ((char *) address2 - (char *) address1);
```

sscanf

Utilizzo

```
MQLONG SBCSprt;
sscanf(line, "%d", &SBCSprt);
```

Non utilizzare

```
MQLONG SBCSprt;
sscanf(line, "%1d", &SBCSprt);
```

`%ld` tenta di inserire un tipo a 8 byte in un tipo a 4 byte; utilizzare solo `%l` se si sta gestendo un tipo di dati `long` effettivo. MQLONG, UINT32 e INT32 vengono definiti come quattro byte, lo stesso di un `int` su tutte le piattaforme IBM MQ :

IBM i ILE/RPG (IBM i Application Programming Reference)

Programmazione dell'applicazione per IBM i.

Utilizzare queste informazioni per sviluppare applicazioni per IBM i.

- “Descrizioni dei tipi di dati su IBM i” a pagina 1023
- “Chiamate di funzioni su IBM i” a pagina 1286
- “Attributi degli oggetti su IBM i” a pagina 1406
- “Applicazioni” a pagina 1454
- “Codici di ritorno per IBM i (ILE RPG)” a pagina 1467
- “Regole per la convalida delle opzioni MQI per IBM i (ILE RPG)” a pagina 1468
- “Codifiche macchina su IBM i” a pagina 1470
- “Opzioni di report e indicatori di messaggi su IBM i” a pagina 1473

Obsolescenza della modalità di compatibilità per le applicazioni RPG e COBOL su IBM i

IBM i

Da IBM MQ for IBM i 9.0, il prodotto non fornisce più il supporto per le applicazioni RPG o COBOL che utilizzano il collegamento dinamico noto come modalità di compatibilità. Questa modalità operativa era necessaria per le applicazioni scritte prima di MQSeries 5.1e le successive versioni del prodotto fornivano un ambiente di runtime compatibile per tali applicazioni, anche se i copybook necessari per la loro compilazione sono stati rimossi in IBM WebSphere MQ 6.0. Il collegamento dinamico (modalità di compatibilità) è stato fornito dai programmi seguenti nella libreria QMQM, che sono stati rimossi in IBM MQ for IBM i 9.0:

- AMQVSTUB
- AMQZSTUB
- QMQM
- MQCLOSE
- MQCONN
- MQDISC
- MQGET
- MQINQ
- MQOPEN

- MQPUT
- MQPUT1
- MQSET

Da IBM MQ for IBM i 9.0, le applicazioni che utilizzano questa modalità operativa di compatibilità devono essere ricompilate per utilizzare le chiamate MQ associate statiche fornite dai programmi di servizio LIBMQM e LIBMQM_R. Programmi di esempio, come AMQ3PUT4 e AMQ3GET4, mostrano come utilizzare questo modello di programmazione. Per ulteriori informazioni sull'utilizzo di queste chiamate MQ , consultare IBM i Application Programming Reference (ILE/RPG).

Note:

- È necessario ricodificare le applicazioni, utilizzando attualmente l'interfaccia CALL 'QMQM', per utilizzare il programma di servizio LIBMQM.

  Gli oggetti programma e i programmi di servizio presenti nell'elenco precedente, ad esempio, QMQM, MQCONN, MQPUT, AMQVSTUB e AMQZSTUB, vengono rimossi in IBM MQ for IBM i 9.0e le applicazioni codificate per l'utilizzo della modalità di compatibilità cessano di funzionare.
- Se le applicazioni sono collegate al programma di servizio LIBMQM all'indirizzo IBM MQ for IBM i 8.0, non è necessario ricompilare o ricollegare tali applicazioni in IBM MQ for IBM i 9.0 o versioni successive.
- Non è possibile installare più di una versione di IBM MQ for IBM i sulla stessa partizione.

Per appurare se il programma RPG o COBOL utilizza la modalità di compatibilità, utilizzare il comando **DSPPGMREF** (Visualizzazione riferimenti programma) per visualizzare i programmi esterni richiamati dal programma applicativo. Se vi sono riferimenti ai programmi elencati in questa sezione, il programma non verrà eseguito in IBM MQ for IBM i 9.0 o versioni successive. Il seguente esempio di output **DSPPGMREF** mostra tre oggetti programma obsoleti, MQCONN, MQOPEN, MQCLOSE:

```
Program . . . . . :  MYAPPPGM
Library . . . . . :    MYLIB
Text 'description'. . . . . :     ILE/COBOL SAMPLE PUT TO QUEUE (MQPUT)
Number of objects referenced  . . . . . :        5
Object  . . . . . :    MQCONN
Library . . . . . :      *LIBL
Object type . . . . . :        *PGM
Object  . . . . . :    MQOPEN
Library . . . . . :      *LIBL
Object type . . . . . :        *PGM
Object  . . . . . :    MQCLOSE
Library . . . . . :      *LIBL
Object type . . . . . :        *PGM
```

Tali programmi devono essere ricompilati utilizzando il metodo Bound Procedural Call descritto in Preparazione dei programmi COBOL in IBM i.

Se si tenta di eseguire un programma applicativo IBM MQ for IBM i 9.0 o successivo che utilizza la modalità di compatibilità, il primo errore più comunemente visualizzato è un MCH3401 che tenta di richiamare il programma MQCONN o QMQM.

Attività correlate

Sviluppo di applicazioni

IBM i Descrizioni dei tipi di dati su IBM i

Questa raccolta di argomenti fornisce le descrizioni dei tipi di dati utilizzati nella programmazione IBM i .

Convenzioni utilizzate nella descrizione dei tipi di dati

Per ogni tipo di dati elementare, queste informazioni forniscono una descrizione del suo uso, in una forma indipendente dal linguaggio di programmazione. Ciò è seguito da dichiarazioni tipiche nella versione ILE del linguaggio di programmazione RPG. Le definizioni dei tipi di dati elementari sono incluse qui per fornire coerenza. RPG utilizza le specifiche 'D' in cui i campi di lavoro possono essere dichiarati utilizzando gli attributi necessari. È possibile, tuttavia, eseguire questa operazione nelle specifiche di calcolo in cui viene utilizzato il campo.

Per utilizzare i tipi di dati elementari, creare:

- Un membro /COPY contenente tutti i tipi di dati oppure
- Una struttura dati esterna (PF) contenente tutti i tipi di dati. È quindi necessario specificare i campi di lavoro con gli attributi 'LIKE' il campo del tipo di dati appropriato.

I vantaggi della seconda opzione sono che le definizioni possono essere utilizzate come 'XX_ENCODE_CASE_CAPS_LOCK_ON field reference file ' per altri oggetti IBM i . Se una definizione del tipo di dati IBM MQ viene modificata, è relativamente semplice ricreare questi oggetti.

Tipi di dati elementari

Tutti gli altri tipi di dati descritti in questa sezione equivalgono direttamente a questi tipi di dati elementari o agli aggregati di questi tipi di dati elementari (array o strutture).

Tabella 679. Tipi di dati elementari

Tipo dati	**Rappresentazione**
MQBOOL	Numero intero con segno a 10 cifre
MQBYTE	Campo alfanumerico a 1 byte
MQBYTE16	Campo alfanumerico a 16 byte
MQBYTE24	campo alfanumerico a 24 byte
MQBYTE32	Campo alfanumerico a 32 - byte
MQBYTE64	campo alfanumerico a 64 byte
MQCAR	Campo alfanumerico a 1 byte
MQCHAR4	Campo alfanumerico a 4 - byte
MQCHAR8	Campo alfanumerico a 8 byte
MQCHAR12	Campo alfanumerico a 12 byte
MQCHAR16	Campo alfanumerico a 16 byte
MQCHAR20	Campo alfanumerico a 20 byte
MQCHAR28	Campo alfanumerico a 28 byte
MQCHAR32	Campo alfanumerico a 32 - byte
MQCHAR48	campo alfanumerico a 48 byte
MQCHAR64	campo alfanumerico a 64 byte
MQCHAR128	Campo alfanumerico a 128 byte
MQCHAR256	Campo alfanumerico a 256 byte
MQFLOAT32	Numero a virgola mobile a 4 byte
MQFLOAT64	Numero a virgola mobile a 8 byte
MQHCONFIG	Handle di configurazione
MQHCONN	Numero intero con segno a 10 cifre
MQHMSG	Handle del messaggio che fornisce l'accesso a un messaggio
MQHOBG	Numero intero con segno a 10 cifre
MQINT8	Numero intero con segno a 8 bit

Tabella 679. Tipi di dati elementari (Continua)	
Tipo dati	**Rappresentazione**
MQINT16	Numero intero con segno a 16 bit
MQINT32	Numero intero con segno a 32 bit
MQINT64	Numero intero firmato a 64 bit
MQLONG	Numero intero con segno a 32 bit
IDMQP	Identificativo del processo
MQPTR	Puntatore
IDMQT	Identificativo thread
MQUINT8	Numero intero senza segno a 8 bit
MQUINT16	Numero intero senza segno a 16 bit
MQUINT32	Numero intero senza segno a 32 bit
MQUINT64	Numero intero senza segno a 64 bit
MQULONG	Numero intero senza segno a 32 bit
PMQACH	Puntatore a una struttura dati di tipo MQACH
PMQAIR	Puntatore a una struttura dati di tipo MQAIR
PMQAXC	Puntatore a una struttura dati di tipo MQAXC
PMAXP	Puntatore a una struttura dati di tipo MAXP
PMQBMO	Puntatore a una struttura dati di tipo MQBMHO
PMQBO	Puntatore a una struttura dati di tipo MQBO
PMQBOOL	Puntatore ai dati di tipo MQBOOL
PMQBYTE	Puntatore ai dati di tipo MQBYTE
PMQBYTEn	Puntatore ai dati di tipo MQBYTEn
PMQCBC	Puntatore ad una struttura di dati di tipo MQCBC
CBD PMQ	Puntatore ad una struttura dati di tipo MQCBD
PMQCAR	Puntatore a una struttura di dati di tipo MQCHAR
PMQCARV	Puntatore ad una struttura dati di tipo MQCHARV
PMQCARn	Puntatore ai dati di tipo MQCHARn
PMQCIH	Puntatore a una struttura dati di tipo MQCIH
PMQCMO	Puntatore a una struttura dati di tipo MQCMHO
NMQPMQ	Puntatore a una struttura di dati di tipo MQCNO
PMQCSP	Puntatore a una struttura dati di tipo MQCSP
MMQCTLO	Puntatore a una struttura di dati di tipo MQCTLO
PMQDH	Puntatore a una struttura dati di tipo MQDH
PMQDHO	Puntatore a una struttura dati di tipo MQDHO
PMQDLH	Puntatore a una struttura dati di tipo MQDLH

Tabella 679. Tipi di dati elementari (Continua)	
Tipo dati	**Rappresentazione**
PMQDMHO	Puntatore a una struttura dati di tipo MQDMHO
PMQDMPO	Puntatore a una struttura dati di tipo MQDMPO
PMQEPH	Puntatore a una struttura di dati di tipo MQEPH
PMQFLOAT32	Puntatore ai dati di tipo MQFLOAT32
PMQFLOAT64	Puntatore ai dati di tipo MQFLOAT64
PMQFUNC	Puntatore a una funzione
PMQGMO	Puntatore a una struttura dati di tipo MQGMO
PMQHCONFIG	Puntatore ai dati di tipo MQHCONFIG
PMQHCONN	Puntatore ai dati di tipo MQHCONN
PMQHMSG	Puntatore ai dati di tipo MQHMSG
PMQHOBG	Puntatore ai dati di tipo MQHOBJ
PMQIIH	Puntatore a una struttura dati di tipo MQIIH
PMQIMPO	Puntatore a una struttura di dati di tipo MQIMPO
PMQINT8	Puntatore ai dati di tipo MQINT8
PMQINT16	Puntatore ai dati di tipo MQINT16
PMQINT32	Puntatore ai dati di tipo MQINT32
PMQINT64	Puntatore ai dati di tipo MQINT64
PMQLONG	Puntatore ai dati di tipo MQLONG
MMQP	Puntatore a una struttura dati di tipo MQMD
MQMDE	Puntatore a una struttura dati di tipo MQMDE
PMQMD1	Puntatore ad una struttura dati di tipo MQMD1
PMQMD2	Puntatore a una struttura di dati di tipo MQMD2
PMQMHBO	Puntatore a una struttura di dati di tipo MQMHBO
PMQOD	Puntatore a una struttura di dati di tipo MQOD
PMQOR	Puntatore a una struttura di dati di tipo MQOR
PMQPD	Puntatore a una struttura di dati di tipo MQPD
IDPMQP	Puntatore a un identificativo processo MQPID
PMQPMO	Puntatore a una struttura di dati di tipo MQPMO
PMQPTR	Puntatore ai dati di tipo MQPTR
PMQRFH	Puntatore a una struttura di dati di tipo MQRFH
PMQRFH2	Puntatore a una struttura dati di tipo MQRFH2
PMQRMH	Puntatore a una struttura dati di tipo MQRMH
RMQP	Puntatore ad una struttura dati di tipo MQRR
PMQSCO	Puntatore a una struttura di dati di tipo MQSCO

Tabella 679. Tipi di dati elementari (Continua)	
Tipo dati	**Rappresentazione**
SMQP	Puntatore a una struttura di dati di tipo MQSD
PMQSMPO	Puntatore a una struttura di dati di tipo MQSMPO
SRO PMQ	Puntatore a una struttura dati di tipo MQSRO
PMQSTS	Puntatore a una struttura di dati di tipo MQSTS
IDMQT	Puntatore a un identificativo thread MQTID
TMQP	Puntatore a una struttura di dati di tipo MQTM
PMQTMC2	Puntatore ad una struttura dati di tipo MQTMC2
PMQUINT8	Puntatore ai dati di tipo MQUINT8
PMQUINT16	Puntatore ai dati di tipo MQUINT16
PMQUINT32	Puntatore ai dati di tipo MQUINT32
PMQUINT64	Puntatore ai dati di tipo MQUINT64
PMQULONG	Puntatore ai dati di tipo MQULONG
IDVOPMQ	Puntatore
PMQWIH	Puntatore a una struttura dati di tipo MQWIH
QQMQP	Puntatore a una struttura dati di tipo MQXQH

IBM i *MQBOOL su IBM i*

Il tipo di dati MQBOOL rappresenta un valore booleano. Il valore 0 rappresenta false. Qualsiasi altro valore rappresenta true.

Un MQBOOL deve essere allineato come per il tipo di dati MQLONG.

IBM i *MQBYTE su IBM i*

Il tipo di dati MQBYTE rappresenta un singolo byte di dati.

Nessuna interpretazione particolare viene posizionata sul byte - viene considerata come una stringa di bit e non come un carattere o un numero binario. Non è richiesto alcun allineamento speciale.

Un array di MQBYTE viene a volte utilizzato per rappresentare un'area di memoria principale con una natura non nota al gestore code. Ad esempio, l'area potrebbe contenere i dati del messaggio dell'applicazione o una struttura. L'allineamento dei limiti di questa area deve essere compatibile con la natura dei dati contenuti al suo interno.

IBM i *MQBYTEn (stringa di n byte) su IBM i*

Ogni tipo di dati MQBYTEn rappresenta una stringa di *n* byte.

Dove *n* può assumere uno dei seguenti valori:

- 16, 24, 32 o 64.

Ogni byte è descritto dal tipo di dati MQBYTE. Non è richiesto alcun allineamento speciale.

Se i dati nella stringa sono più corti della lunghezza definita della stringa, i dati devono essere riempiti con valori null per riempire la stringa.

Quando il gestore code restituisce stringhe di byte all'applicazione (ad esempio, sulla chiamata MQGET), il gestore code riempie sempre con valori null la lunghezza definita della stringa.

Sono disponibili costanti che definiscono la lunghezza dei campi stringa di byte.

IBM i *MQCHAR (carattere) su IBM i*

Il tipo di dati MQCHAR rappresenta un singolo carattere.

Il CCSID (coded character set identifier) del carattere è quello del gestore code (consultare l'attributo **CodedCharSetId** nell'argomento CodedCharSetId). Non è richiesto alcun allineamento speciale.

Nota: I dati del messaggio dell'applicazione specificati nelle chiamate MQGET, MQPUT e MQPUT1 sono descritti dal tipo di dati MQBYTE e non dal tipo di dati MQCHAR.

IBM i *MQCHARn (Stringa di n caratteri) su IBM i*

Ciascun tipo di dati MQCHARn rappresenta una stringa di *n* caratteri.

Dove *n* può assumere uno dei seguenti valori:

- 4, 8, 12, 16, 20, 28, 32, 48, 64, 128 o 256

Ogni carattere è descritto dal tipo di dati MQCHAR. Non è richiesto alcun allineamento speciale.

Se i dati nella stringa sono più brevi della lunghezza definita della stringa, i dati devono essere riempiti con spazi vuoti per riempire la stringa. In alcuni casi è possibile utilizzare un carattere null per terminare prematuramente la stringa, invece di riempirla di spazi vuoti; il carattere null e i caratteri successivi vengono trattati come spazi vuoti, fino alla lunghezza definita della stringa. Le posizioni in cui è possibile utilizzare un valore null vengono identificate nelle descrizioni della chiamata e del tipo di dati.

Quando il gestore code restituisce delle stringhe di caratteri all'applicazione (ad esempio, sulla chiamata MQGET), il gestore code riempisce sempre con spazi vuoti la lunghezza definita della stringa; il gestore code non utilizza il carattere null per delimitare la stringa.

Sono disponibili costanti che definiscono la lunghezza dei campi stringa di caratteri.

IBM i *MQFLOAT32 su IBM i*

Il tipo di dati MQFLOAT32 è un numero a virgola mobile a 32 bit rappresentato utilizzando il formato a virgola mobile IEEE standard.

Un MQFLOAT32 deve essere allineato su un limite di 4 byte.

IBM i *MQFLOAT64 su IBM i*

Il tipo di dati MQFLOAT64 è un numero a virgola mobile a 64 bit rappresentato utilizzando il formato a virgola mobile IEEE standard.

Un MQFLOAT64 deve essere allineato su un limite di 8 byte.

MQHCONFIG - handle di configurazione

Il tipo di dati MQHCONFIG rappresenta un handle di configurazione, ossia il componente che viene configurato per un determinato servizio installabile. Un handle di configurazione deve essere allineato sul suo limite naturale.

Nota: Le applicazioni devono testare le variabili di questo tipo solo per l'uguaglianza.

IBM i *MQHCONN (handle di connessione) su IBM i*

Il tipo di dati MQHCONN rappresenta un handle di connessione, ossia la connessione a uno specifico gestore code.

Un handle di connessione deve essere allineato sul suo limite naturale.

Nota: Le applicazioni devono testare le variabili di questo tipo solo per l'uguaglianza.

IBM i *MQHMSG (Gestione messaggi) su IBM i*

Il tipo di dati **MQHMSG** rappresenta un handle del messaggio che fornisce l'accesso a un messaggio.

Un gestore messaggi deve essere allineato su un limite di 8 byte.

Nota: Le applicazioni devono testare le variabili di questo tipo solo per l'uguaglianza.

IBM i *MQHOBJ (handle oggetto) su IBM i*

Il tipo di dati MQHOBJ rappresenta un handle di oggetto che fornisce l'accesso a un oggetto.

Una gestione oggetto deve essere allineata sul suo limite naturale.

Nota: Le applicazioni devono testare le variabili di questo tipo solo per l'uguaglianza.

IBM i *MQINT8 (numero intero con segno a 8 bit) su IBM i*

Il tipo di dati MQINT8 è un numero intero con segno a 8 bit che può assumere qualsiasi valore compreso tra -128 e +127, a meno che non sia altrimenti limitato dal contesto.

IBM i *MQINT16 (numero intero con segno a 16 bit) su IBM i*

Il tipo di dati MQINT16 è un numero intero con segno a 16 bit che può assumere qualsiasi valore compreso tra -32 768 e +32 767, a meno che non sia altrimenti limitato dal contesto.

Un MQINT16 deve essere allineato su un limite di 2 byte.

IBM i *MQINT32 (numero intero a 32 bit) su IBM i*

Il tipo di dati MQINT32 è un numero intero con segno a 32 bit.

Equivale a MQLONG.

IBM i *MQINT64 (numero intero a 64-bit) su IBM i*

Il tipo di dati MQINT64 è un numero intero con segno a 64 bit che può assumere qualsiasi valore compreso tra -9 223 372 036 854 775 808 e + 9 223 372 036 854 775 807, a meno che non sia altrimenti limitato dal contesto.

Per COBOL, l'intervallo valido è limitato da -999 999 999 999 999 999 a +999 999 999 999 999 999 999. MQINT64 deve essere allineato su un limite di 8 byte.

IBM i *MQLONG (numero intero lungo) su IBM i*

Il tipo di dati MQLONG è un numero intero binario con segno a 32 bit che può assumere qualsiasi valore compreso tra -2 147 483 648 e + 2 147 483 647, a meno che non sia altrimenti limitato dal contesto, allineato sul suo limite naturale.

MQPID - identificativo processo

L'identificativo del processo IBM MQ .

Questo è lo stesso identificativo utilizzato nella traccia IBM MQ e nei dump FFST , ma potrebbe essere diverso dall'identificativo del processo del sistema operativo.

Puntatore MQPTR

Il tipo di dati MQPTR è l'indirizzo dei dati di qualsiasi tipo. Un puntatore deve essere allineato sul suo limite naturale; questo è un limite di 16-byte su IBM i.

Alcuni linguaggi di programmazione supportano i puntatori digitati; l'MQI li utilizza anche in alcuni casi.

MQTID - identificativo thread

L'identificativo del thread MQ .

Questo è lo stesso identificativo utilizzato nella traccia di MQ e nei dump FFST , ma potrebbe essere diverso dall'identificativo del thread del sistema operativo.

IBM i *MQUINT8 (numero intero senza segno a 8 bit) su IBM i*

Il tipo di dati MQUINT8 è un numero intero senza segno a 8 bit che può assumere qualsiasi valore compreso tra 0 e +255, a meno che non sia altrimenti limitato dal contesto.

MQUINT16 - numero intero senza segno a 16 bit

Il tipo di dati MQUINT16 è un numero intero senza segno a 16 bit che può assumere qualsiasi valore compreso tra 0 e +65 535, a meno che non sia altrimenti limitato dal contesto.

Un MQUINT16 deve essere allineato su un limite di 2 byte.

IBM i *MQUINT32 (numero intero senza segno a 32 bit) su IBM i*

Il tipo di dati MQUINT32 è un numero intero senza segno a 32 bit. È equivalente a MQULONG.

MQUINT64 - numero intero senza segno a 64 bit

Il tipo di dati MQUINT64 è un numero intero senza segno a 64 bit che può assumere qualsiasi valore compreso tra 0 e +18 446 744 073 709 551 615 a meno che non sia altrimenti limitato dal contesto.

Per COBOL, l'intervallo valido è compreso tra 0 e 999 999 999 999 999 999. Un MQUINT64 deve essere allineato su un limite di 8 byte.

MQULONG - Numero intero senza segno a 32 bit

Il tipo di dati MQULONG è un numero intero binario senza segno a 32 bit che può assumere qualsiasi valore compreso tra 0 e + 4 294 967 294, a meno che non sia altrimenti limitato dal contesto.

Un MQULONG deve essere allineato su un limite di 4 byte.

PMQACH - puntatore a una struttura di dati di tipo MQACH

Un puntatore a una struttura dati di tipo MQACH.

PMQAIR - puntatore a una struttura di dati di tipo MQAIR

Un puntatore ad una struttura dati di tipo MQAIR.

PMQAXC - puntatore a una struttura dati di tipo MQAXC

Un puntatore a una struttura dati di tipo MQAXC.

PMQAXP - puntatore a una struttura di dati di tipo MQAXP

Un puntatore a una struttura dati di tipo MQAXP.

PMQBMHO - puntatore a una struttura di dati di tipo MQBMHO

Un puntatore ad una struttura dati di tipo MQBMHO.

PMQBO - puntatore a una struttura di dati di tipo MQBO

Un puntatore a una struttura dati di tipo MQBO.

PMQBOOL - puntatore ai dati di tipo MQBOOL
Un puntatore ai dati di tipo MQBOOL.

Un puntatore ai dati di tipo MQBOOL.

PMQBYTE - puntatore a un tipo di dati MQBYTE
Un puntatore a un tipo di dati MQBYTE.

PMQBYTEn - puntatore a una struttura dati di tipo MQBYTEn
Un puntatore a una struttura dati di tipo MQBYTEn, dove n può essere 8, 12, 16, 24, 32, 40, 48 o 128.

PMQCBC - puntatore a una struttura di dati di tipo MQCBC
Un puntatore ad una struttura dati di tipo MQCBC.

PMQCBD - puntatore a una struttura di dati di tipo MQCBD
Un puntatore ad una struttura dati di tipo MQCBD.

PMQCHAR - puntatore ai dati di tipo MQCHAR
Un puntatore ai dati di tipo MQCHAR.

PMQCHARV - puntatore ad una struttura dati di tipo MQCHARV
Un puntatore ad una struttura dati di tipo MQCHARV.

PMQCHARn - puntatore a un tipo di dati MQCHARn
Un puntatore a un tipo di dati MQCHARn, dove n può essere 4, 8, 12, 20, 28, 32, 64, 128, 256, 264.

PMQCIH - puntatore a una struttura dati di tipo MQCIH
Un puntatore a una struttura dati di tipo MQCIH.

PMQCMHO - puntatore a una struttura di dati di tipo MQCMHO
Un puntatore a una struttura dati di tipo MQCMHO.

PMQCNO - puntatore a una struttura dati di tipo MQCNO
Un puntatore a una struttura dati di tipo MQCNO.

PMQCSP - puntatore a una struttura dati di tipo MQCSP
Un puntatore a una struttura dati di tipo MQCSP.

PMQCTLO - puntatore a una struttura dati di tipo MQCTLO
Un puntatore a una struttura dati di tipo MQCTLO.

PMQDH - puntatore a una struttura di dati di tipo MQDH
Un puntatore a una struttura dati di tipo MQDH.

PMQDHO - puntatore a una struttura di dati di tipo MQDHO
Un puntatore a una struttura dati di tipo MQDHO.

PMQDLH - puntatore ad una struttura dati di tipo MQDLH
Un puntatore a una struttura dati di tipo MQDLH.

PMQDMHO - puntatore a una struttura dati di tipo MQDMHO
Un puntatore a una struttura dati di tipo MQDMHO.

PMQDMPO - puntatore a una struttura dati di tipo MQDMPO
Un puntatore a una struttura dati di tipo MQDMPO.

Un puntatore a una struttura dati di tipo MQDMPO.

PMQEPH - puntatore a una struttura di dati di tipo MQEPH
Un indicatore a una struttura dati di tipo MQEPH.

PMQFLOAT32 - Puntatore ai dati di tipo MQFLOAT32
Un puntatore ai dati di tipo MQFLOAT32.

PMQFLOAT64 - puntatore ai dati di tipo MQFLOAT64
Un puntatore ai dati di tipo MQFLOAT64.

PMQFUNC - puntatore a una funzione
Un puntatore a una funzione.

PMQGMO - puntatore ad una struttura dati di tipo MQGMO
Un puntatore a una struttura di dati di tipo MQGMO.

PMQHCONFIG - puntatore a un tipo di dati MQHCONFIG
Un puntatore a un tipo di dati MQHCONFIG.

PMQHCONN - puntatore a un tipo di dati MQHCONN
Un puntatore ad un tipo di dati MQHCONN.

PMQHMSG - puntatore a un tipo di dati MQHMSG
Un puntatore ad un tipo di dati MQHMSG.

PMQHOBJ - puntatore ai dati di tipo MQHOBJ
Un puntatore ai dati di tipo MQSMPO.

PMQIIH - puntatore a una struttura dati di tipo MQIIH
Un puntatore a una struttura dati di tipo MQIIH.

PMQIMPO - puntatore a una struttura di dati di tipo MQIMPO
Un puntatore a una struttura dati di tipo MQIMPO.

PMQINT8 - puntatore ai dati di tipo MQINT8
Un puntatore ai dati di tipo MQINT8.

PMQINT16 - puntatore ai dati di tipo MQINT16
Un puntatore ai dati di tipo MQINT16.

IBM i *PMQINT32 (Puntatore ai dati di tipo MQINT32) su IBM i*
Il tipo di dati PMQINT32 è un puntatore ai dati di tipo MQINT32. Equivale a PMQLONG.

IBM i *PMQINT64 (Puntatore ai dati di tipo MQINT64) su IBM i*
Il tipo di dati PMQINT64 è un puntatore ai dati di tipo MQINT64.

PMQLONG - puntatore ai dati di tipo MQLONG
Un puntatore ai dati di tipo MQLONG.

PMQMD - puntatore alla struttura di tipo MQMD
Un puntatore alla struttura di tipo MQMD.

PMQMDE - puntatore a una struttura dati di tipo MQMDE
Un puntatore a una struttura dati di tipo MQMDE.

PMQMDI - puntatore a una struttura di dati di tipo MQMDI
Un puntatore a una struttura dati di tipo MQMDI.

PMQMD2 - puntatore a una struttura di dati di tipo MQMD2
Un puntatore a una struttura dati di tipo MQMD2

PMQMHBO - puntatore a una struttura dati di tipo MQMHBO
Un puntatore a una struttura di dati di tipo MQMHBO.

PMQOD - puntatore a una struttura di dati di tipo MQOD
Un puntatore ad una struttura dati di tipo MQOD.

PMQOR - puntatore a una struttura dati di tipo MQOR
Un puntatore a una struttura di dati di tipo MQOR.

PMQPD - puntatore a una struttura dati di tipo MQPD
Un puntatore a una struttura dati di tipo MQPD.

PMQPID - puntatore a un identificativo processo
Un puntatore a un identificativo di processo.

PMQPMO - puntatore a una struttura dati di tipo MQPMO
Un puntatore ad una struttura dati di tipo MQPMO.

PMQPTR - puntatore ai dati di tipo MQPTR
Un puntatore ai dati di tipo MQPTR.

PMQRFH - puntatore a una struttura dati di tipo MQRFH
Un puntatore a una struttura dati di tipo MQRFH.

PMQRFH2 - puntatore a una struttura dati di tipo MQRFH2
Un puntatore ad una struttura dati di tipo MQRFH2.

.

PMQRMH - puntatore a una struttura dati di tipo MQRMH
Un puntatore a una struttura dati di tipo MQRMH.

PMQRR - puntatore a una struttura dati di tipo MQRR
Un puntatore a una struttura dati di tipo MQRR.

PMQSCO - puntatore a una struttura dati di tipo MQSCO
Un puntatore a una struttura dati di tipo MQSCO.

.

PMQSD - puntatore a una struttura dati di tipo MQSD
Un puntatore a una struttura dati di tipo MQSD.

PMQSMPO - puntatore a una struttura dati di tipo MQSMPO
Un puntatore ad una struttura dati di tipo MQSMPO.

PMQSRO - puntatore a una struttura dati di tipo MQSRO
Un puntatore a una struttura dati di tipo MQSRO.

PMQSTS - puntatore a una struttura di dati di tipo MQSTS
Un puntatore ad una struttura dati di tipo MQSTS.

PMQTID - puntatore a una struttura dati di tipo MQTID
Un puntatore a una struttura dati di tipo MQTID.

PMQTM - puntatore a una struttura di dati di tipo MQTM
Un puntatore a una struttura dati di tipo MQTM.

PMQTMC2 - puntatore a una struttura dati di tipo MQTMC2
Un puntatore a una struttura dati di tipo MQTMC2.

PMQUINT8 - puntatore ai dati di tipo MQUINT8
Un puntatore ai dati di tipo MQUINT8.

PMQUINT16 - puntatore ai dati di tipo MQUINT16
Un puntatore ai dati di tipo MQUINT16.

IBM i *PMQUINT32 (Puntatore ai dati di tipo MQUINT32) su IBM i*
Il tipo di dati PMQUINT32 è un puntatore ai dati di tipo MQUINT32. Equivale a PMQULONG.

IBM i *PMQUINT64 (Puntatore ai dati di tipo MQUINT64) su IBM i*
Il tipo di dati PMQUINT64 è un puntatore ai dati di tipo MQUINT64.

PMQULONG - puntatore ai dati di tipo MQULONG
Un puntatore ai dati di tipo MQULONG.

puntatore PMQVOID
Un puntatore.

PMQWIH - puntatore a una struttura dati di tipo MQWIH
Un puntatore a una struttura dati di tipo MQWIH.

PMQXQH - puntatore a una struttura dati di tipo MQXQH
Un puntatore a una struttura dati di tipo MQXQH.

Considerazioni sulla lingua

Questo argomento contiene informazioni che consentono di utilizzare l'MQI dal linguaggio di programmazione RPG.

Alcune di queste considerazioni sul linguaggio sono:

- "file COPY" a pagina 1036
- "Chiamate" a pagina 1038
- "Parametri di chiamata" a pagina 1038
- "Strutture" a pagina 1039
- "Costanti con nome" a pagina 1039
- "Procedure MQI" a pagina 1039
- "Considerazioni sui thread" a pagina 1039
- "Controllo del commit" a pagina 1040
- "Codifica delle chiamate associate" a pagina 1040
- "Convenzioni notazionali" a pagina 1041

file COPY

Vari file COPY vengono forniti per assistere nella scrittura di programmi applicativi RPG che utilizzano l'accodamento messaggi. Esistono tre serie di file COPY:

- I file COPY con nomi che terminano con la lettera *G* vengono utilizzati con programmi che utilizzano collegamenti statici. Questi file vengono inizializzati con le eccezioni riportate in "Strutture" a pagina 1039.
- I file COPY con nomi che terminano con la lettera *H* sono da utilizzare con programmi che utilizzano il collegamento statico, ma **non** sono inizializzati.
- I file COPY con nomi che terminano con la lettera *R* vengono utilizzati con programmi che utilizzano il collegamento dinamico. Questi file vengono inizializzati con le eccezioni riportate in "Strutture" a pagina 1039.

I file COPY risiedono in QRPGLESRC nella libreria QMQM.

Per ogni serie di file COPY ci sono due file contenenti costanti denominate e un file per ognuna delle strutture. I file COPY sono riepilogati in Tabella 680 a pagina 1036.

Tabella 680. File RPG COPY

Nome file (collegamento statico, inizializzato, CMQ* G)	**Nome file (collegamento statico, non inizializzato, CMQ* H)**	**Nome file (collegamento dinamico, inizializzato, CMQ* R)**	**Indice**
CMQBOG	CMQBOH	-	Struttura delle opzioni di inizio

Tabella 680. File RPG COPY (Continua)

Nome file (collegamento statico, inizializzato, CMQ* G)	**Nome file (collegamento statico, non inizializzato, CMQ* H)**	**Nome file (collegamento dinamico, inizializzato, CMQ* R)**	**Indice**
CMQCDG	CMQCDH	CMQCDR	Struttura definizione canale
CMQCFBFG	CMQCFBFH	-	Parametro filtro bit PCF
CMQCFG	-	-	Costanti per PCF ed eventi
CMQCFBSG	CMQCFBSH	-	Stringa di byte PCF
CMQCFGRG	CMQCFGRH	-	Parametro gruppo PCF
CMQCFIFG	CMQCFIFH	-	Parametro filtro numero intero PCF
CMQCFHG	CMQCFH	-	Intestazione PCF
CMQCFILG	CMQCFILH	-	Struttura dei parametri dell'elenco di numeri interi PCF
CMQCFING	CMQCFIN	-	Struttura parametro numero intero PCF
CMQCFSFG	CMQCFSFH	-	Parametro filtro stringa PCF
CMQCFSLG	CMQCFSLA	-	Struttura dei parametri dell'elenco di stringhe PCF
CMQCFSTG	CMQCFSTH	-	Struttura del parametro stringa PCF
CMQCFXLG	CMQCFXLH	-	Nome breve PCF per CFIL64
CMQCFXNG	CMQCFXNH	-	Nome breve PCF per CFIN64
CMQCIHG	CMQCIHH	-	Struttura dell'intestazione delle informazioni CICS
CMQCNOG	CMQCNOH	-	Struttura delle opzioni di collegamento
CMQCSPG	CMQCSPH	-	Parametri di sicurezza
CMQCXPG	CMQCXPH	CMQCXPR	Struttura del parametro di uscita canale
CMQDHG	CMQDHH	CMQDHR	Struttura intestazione distribuzione
CMQDLG	CMQDLH	CMQDLHR	Struttura intestazione lettera non recapitabile
CMQDXPG	CMQDXPH	CMQDPR	Struttura del parametro di uscita conversione dati
CMQEPHG	HCPMEPH	-	Struttura dell'intestazione PCF incorporata
CMQG	-	RMQC	Costanti denominate per MQI principale
CMQGMOG	MMQGMOH	CMQGMOR	Richiama la struttura delle opzioni del messaggio
CMQIIHG	CMQIIHH	CMQIIR	Struttura dell'intestazione delle informazioni IMS
CMQMDEG	CMQMDEH	CMQMDER	Struttura di estensione del descrittore del messaggio

Tabella 680. File RPG COPY (Continua)			
Nome file (collegamento statico, inizializzato, CMQ* G)	**Nome file (collegamento statico, non inizializzato, CMQ* H)**	**Nome file (collegamento dinamico, inizializzato, CMQ* R)**	**Indice**
CMQMDG	MCMMDH	MDRCMM	Struttura descrittore del messaggio
CMQMD1G	CMQMD1H	CMQMD1R	Struttura descrittore messaggi versione 1
CMQMD2G	CMQMD2H	-	Struttura descrittore messaggio versione 2
CMQODG	CMQODH	CMQODR	Struttura descrittore oggetto
CMQORG	CMQORH	CMQORR	Struttura record oggetto
CMQPMOG	MCMPMOH	CMQPMOR	Struttura delle opzioni del messaggio di inserimento
CMQPSG	-	-	Costanti per pubblicazione / sottoscrizione
CMQRFHG	CMQRFH	-	Regole e struttura intestazione di formattazione
CMQRFH2G	CMQRFH2H	-	Struttura dell'intestazione 2 di formattazione e regole
CMQRMHG	CMQRMHH	CMQRMHR	Struttura intestazione messaggio di riferimento
CMQRRG	CMQRRH	CMQRRR	Struttura del record di risposta
CMQTMCG	CMQTMCH	CMQTMCR	Struttura del messaggio trigger (formato carattere)
CMQTMC2G	CMQTMC2H	CMQTMC2R	Struttura del messaggio trigger (formato carattere) versione 2
MQTMG	MMQTMH	CMQTMR	Struttura del messaggio trigger
CMQWIHG	CMQWIHH	-	Struttura intestazione informazioni di lavoro
CMQXG	-	CMQXR	Costanti denominate per l'uscita di conversione dati
CMQXQHG	CMQXQH	CMQXQR	Struttura intestazione coda di trasmissione

Chiamate

Le chiamate vengono descritte utilizzando i singoli nomi.

Parametri di chiamata

Alcuni parametri passati a MQI possono avere più di una funzione simultanea. Ciò è dovuto al fatto che il valore intero passato viene spesso testato sull'impostazione dei singoli bit all'interno del campo e non sul suo valore totale. Ciò consente di 'aggiungere ' diverse funzioni insieme e passarle come un unico parametro.

Strutture

Tutte le strutture IBM MQ sono definite con valori iniziali per i campi, con le seguenti eccezioni:

- Qualsiasi struttura con un suffisso di H.
- MMQTMC
- MQTMC2

Questi valori sono definiti nella relativa tabella per ciascuna struttura.

Le dichiarazioni di struttura non contengono istruzioni DS . Ciò consente all'applicazione di dichiarare una singola struttura di dati o una struttura di dati a più ricorrenze, codificando l'istruzione DS e quindi utilizzando l'istruzione /COPY per copiare nel resto della dichiarazione:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7
D* Declare an MQMD data structure with 5 occurrences
DMYMD        DS                          5
D/COPY CMQMDR
```

Costanti con nome

Esistono molti valori interi e di caratteri che forniscono l'interscambio di dati tra il programma applicativo e il gestore code. Per facilitare un approccio più leggibile e coerente all'utilizzo di questi valori, le costanti denominate vengono definite per tali valori. È possibile utilizzare queste costanti denominate e non i valori che rappresentano, poiché ciò migliora la leggibilità del codice sorgente del programma.

Quando il file COPY CMQG è incluso in un programma per la definizione delle costanti, il compilatore RPG emetterà molti messaggi con severità zero per le costanti che non vengono utilizzate dal programma; questi messaggi sono benigni e possono essere tranquillamente ignorati.

Procedure MQI

Quando si utilizzano le chiamate collegate ILE, è necessario collegarsi alle procedure MQI quando si crea il programma. Queste procedure vengono esportate dai programmi di servizio riportati di seguito, in base alle necessità:

QMQM/LIBMQM
Questo programma di servizio contiene i collegamenti a thread singolo per versione 5.1 e superiore. Consultare la seguente sezione per considerazioni speciali sulla scrittura di applicazioni con thread.

QMQM/LIBMQM_R
Questo programma di servizio contiene i collegamenti multi - thread per la versione 5.1 e successive. Consultare la seguente sezione per considerazioni speciali sulla scrittura di applicazioni con thread.

QMQM/LIBMQIC
Questo programma di servizio è per il collegamento di applicazioni client senza thread.

QMQM/LIBMQIC_R
Questo programma di servizio è per il collegamento di applicazioni client con thread.

Utilizzare il comando CRTPGM per creare i propri programmi. Ad esempio, il seguente comando crea un programma a thread singolo che utilizza le chiamate collegate ILE:

```
CRTPGM PGM(MYPROGRAM) BNDSRVPGM(QMQM/LIBMQM)
```

Considerazioni sui thread

Il compilatore RPG utilizzato per IBM i fa parte di WebSphere Development Toolset and WebSphere Development Studio for IBM i ed è noto come compilatore ILE RPG IV.

In generale, i programmi RPG non devono utilizzare i programmi di servizio a più sottoprocessi. Le eccezioni sono programmi RPG creati utilizzando ILE RPG IV Compiler e contenenti la parola chiave

THREAD(*SERIALIZE) nella specifica di controllo. Tuttavia, anche se questi programmi sono thread - safe, è necessario considerare attentamente la progettazione dell'applicazione in generale, poiché THREAD(*SERIALIZE) forza la serializzazione delle procedure RPG a livello di modulo, e ciò potrebbe avere un impatto negativo sulle prestazioni complessive.

Dove i programmi RPG vengono utilizzati come uscite di conversione dati, devono essere resi thread - safe e devono essere ricompilati utilizzando il compilatore ILE RPG versione 4.4 o superiore, con THREAD(*SERIALIZE) specificato nella specifica di controllo.

Per ulteriori informazioni sui thread, consultare *IBM i IBM MQ Development Studio: ILE RPG Referencee* il manuale *IBM i IBM MQ Development Studio: ILE RPG Programmer's Guide*.

Controllo del commit

Le funzioni di sincronizzazione MQI MQCMIT e MQBACK sono disponibili per i programmi ILE RPG in esecuzione in modalità normale; queste chiamate consentono al programma di eseguire il commit e il backout delle modifiche alle risorse MQ .

Codifica delle chiamate associate

Le procedure MQI ILE sono riportate in Tabella 681 a pagina 1040.

Tabella 681. Chiamate collegate ILE RPG supportate da ciascun programma di servizio

Nome della chiamata	LIBMQM e LIBMQM_R	LIBMQIC e LIBMQIC_R
MQBACK	Y	Y
MQBEGIN	Y	Y
MQCMIT	Y	Y
MQCLOSE	Y	Y
MQCONN	Y	Y
MQCONNX	Y	Y
MQDISC	Y	Y
MQGET	Y	Y
MQINQ	Y	Y
MQOPEN	Y	Y
MQPUT	Y	Y
MQPUT1	Y	Y
MQSET	Y	Y
MQXCNC	Y	Y

Per utilizzare queste procedure è necessario:

1. Definire le procedure esterne nelle specifiche 'D'. Questi sono tutti disponibili all'interno del membro del file COPY CMQG contenente le costanti denominate.
2. Utilizzare il codice operazione CALLP per richiamare la procedura insieme ai relativi parametri.

Ad esempio la chiamata MQOPEN richiede l'inclusione del codice seguente:

```
D*----------------------------------------------------------------
D**  MQOPEN Call -- Open Object  (From COPY file CMQG)           **
D*----------------------------------------------------------------
D*
```

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQOPEN           PR                  EXTPROC('MQOPEN')
D* Connection handle
D HCONN                          10I 0 VALUE
D* Object descriptor
D OBJDSC                        224A
D* Options that control the action of MQOPEN
D OPTS                           10I 0 VALUE
D* Object handle
D HOBJ                           10I 0
D* Completion code
D CMPCOD                         10I 0
D* Reason code qualifying CMPCOD
D REASON                         10I 0
D*
```

Per richiamare la procedura, dopo aver inizializzato i vari parametri, è necessario il seguente codice:

```
...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+....8
    C                   CALLP     MQOPEN(HCONN : MQOD : OPTS : HOBJ :
    C                             CMPCOD : REASON)
```

Qui, la struttura MQOD viene definita utilizzando il membro COPY CMQODG che la suddivide nei suoi componenti.

Convenzioni notazionali

Gli ultimi argomenti in questa sezione mostrano come:

- È necessario richiamare le chiamate
- I parametri devono essere dichiarati
- È necessario dichiarare diversi tipi di dati

In alcuni casi, i parametri sono matrici o stringhe di caratteri con una dimensione non fissa. Per questi, viene utilizzata una "n" minuscola per rappresentare una costante numerica. Quando la dichiarazione per tale parametro è codificata, la "n" deve essere sostituita dal valore numerico richiesto.

IBM i MQAIR (Record delle informazioni di autenticazione) su IBM i

La struttura MQAIR rappresenta il record delle informazioni di autenticazione.

Panoramica

Scopo: la struttura MQAIR consente a una applicazione in esecuzione come client IBM MQ di specificare le informazioni su un programma di autenticazione da utilizzare per la connessione client. La struttura è un parametro di input sulla chiamata MQCONNX.

Serie di caratteri e codifica: i dati in MQAIR devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e la codifica del gestore code locale fornita da ENNAT.

- “Campi” a pagina 1041
- “Valori iniziali” a pagina 1043
- “Dichiarazione RPG” a pagina 1044

Campi

La struttura MQAIR contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

AICN (numero intero con segno a 10 cifre)

Si tratta del nome host o dell'indirizzo di rete di un host su cui è in esecuzione il server LDAP. Questo può essere seguito da un numero di porta facoltativo, racchiuso tra parentesi.

Se il valore è più breve della lunghezza del campo, terminare il valore con un carattere null o riempirlo con spazi vuoti fino alla lunghezza del campo. Se il valore non è valido, la chiamata ha esito negativo con codice di errore RC2387.

Il numero di porta predefinito è 389.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da LNAICN. Il valore iniziale di questo campo è costituito da spazi.

AITYP (numero intero con segno a 10 cifre)

Questo è il tipo di informazioni di autenticazione contenute nel record.

Il valore deve essere:

AITLDP
Revoca del certificato utilizzando il server LDAP.

Se il valore non è valido, la chiamata ha esito negativo con codice di errore RC2386.

Questo è un campo di immissione. Il valore iniziale di questo campo è AITLDP.

AIPW (numero intero con segno a 10 cifre)

Questa è la parola d'ordine necessaria per accedere al server CRL LDAP.

Se il valore è più breve della lunghezza del campo, terminare il valore con un carattere null o riempirlo con spazi vuoti fino alla lunghezza del campo. Se il server LDAP non richiede una password o si omette il nome utente LDAP, *AIPW* deve essere null o vuoto. Se si omette il nome utente LDAP e *AIPW* non è null o vuoto, la chiamata ha esito negativo con codice di errore RC2390.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da LNLDPW. Il valore iniziale di questo campo di caratteri vuoti.

AILUL (numero intero con segno a 10 cifre)

Questa è la lunghezza in byte del nome utente LDAP indicato dal campo *AILUP* o *AILUO* . Il valore deve essere compreso nell'intervallo tra zero e LNDISN. Se il valore non è valido, la chiamata ha esito negativo con codice di errore RC2389.

Se il server LDAP interessato non richiede un nome utente, impostare questo campo su zero.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

AILUO (numero intero con segno a 10 cifre)

Questo è l'offset in byte del nome utente LDAP dall'inizio della struttura MQAIR.

L'offset può essere positivo o negativo. Il campo viene ignorato se *LDAPUserNameLength* è zero.

È possibile utilizzare *LDAPUserNamePtr* o *LDAPUserNameOffset* per specificare il nome utente LDAP, ma non entrambi; consultare la descrizione del campo *LDAPUserNamePtr* per i dettagli.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

AILUP (numero intero con segno a 10 cifre)

Questo è il nome utente LDAP.

Consiste nel DN (Distinguished Name) dell'utente che sta tentando di accedere al server CRL LDAP. Se il valore è più breve della lunghezza specificata da *AILUL*, terminare il valore con un carattere null o riempirlo con spazi vuoti fino alla lunghezza *AILUL*. Il campo viene ignorato se *AILUL* è zero.

È possibile specificare il nome utente LDAP in uno dei modi seguenti:

- Utilizzando il campo puntatore *AILUP*

  In questo caso, l'applicazione può dichiarare una stringa separata dalla struttura MQAIR e impostare *AILUP* sull'indirizzo della stringa.

Considerare l'utilizzo di *AILUP* per i linguaggi di programmazione che supportano il tipo di dati del puntatore in un modo che sia portabile in ambienti differenti (ad esempio, il linguaggio di programmazione C).

- Utilizzando il campo offset *AILUO*

  In questo caso, l'applicazione deve dichiarare una struttura composta contenente la struttura MQSCO seguita dall'array di record MQAIR seguito dalle stringhe del nome utente LDAP e impostare *AILUO* sull'offset della stringa del nome appropriata dall'inizio della struttura MQAIR. Verificare che questo valore sia corretto e che abbia un valore che possa essere utilizzato all'interno di un MQLONG (il linguaggio di programmazione più restrittivo è COBOL, per cui l'intervallo valido è compreso tra -999 999 999 e +999 999 999).

  Considerare l'utilizzo di *AILUO* per i linguaggi di programmazione che non supportano il tipo di dati puntatore o che implementano il tipo di dati puntatore in un modo che potrebbe non essere portabile in ambienti differenti (ad esempio, il linguaggio di programmazione COBOL).

Qualunque sia la tecnica scelta, utilizzare solo uno tra *AILUP* e *AILUO* ; la chiamata ha esito negativo con codice motivo RC2388.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e, in caso contrario, una stringa di byte completamente null.

Nota: Su piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata.

AISID (numero intero con segno a 10 cifre)

Il valore deve essere:

IDISV
Identificativo per il record delle informazioni di autenticazione.

Questo è sempre un campo di input. Il valore iniziale di questo campo è AISIDV.

AIVER (numero intero con segno a 10 cifre)

Il valore deve essere:

AIVER1
Record delle informazioni di autenticazione Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

RIRVERC
Versione corrente del record delle informazioni di autenticazione.

Questo è sempre un campo di input. Il valore iniziale di questo campo è AIVER1.

Valori iniziali

Tabella 682. Campi in MQAIR per MQAIR

Nome campo	Nome della costante	Valore della costante
AISID	IDISV	'AIR¬'
AIVER	AVERC	1
AITYP	AITLDP	1
AICN	Nessuna	Stringa null o spazi vuoti
AILUP	Nessuna	Puntatore null o byte null
AILUO	Nessuna	0

Tabella 682. Campi in MQAIR per MQAIR (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
AILUL	Nessuna	0
AIPW	Nessuna	Stringa null o spazi vuoti

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQAIR Structure
D*
D* Structure identifier
D  AISID                  1      4    INZ('AIR ')
D* Structure version number
D  AIVER                  5      8I 0 INZ(1)
D* Type of authentication information
D  AITYP                  9     12I 0 INZ(1)
D* Connection name of CRL LDAP server
D  AICN                  13    276    INZ
D* Address of LDAP user name
D  AILUP                277    292*   INZ(*NULL)
D* Offset of LDAP user name from start of MQAIR structure
D  AILUO                293    296I 0 INZ(0)
D* Length of LDAP user name
D  AILUL                297    300I 0 INZ(0)
D* Password to access LDAP server
D  AIPW                 301    332    INZ
```

IBM i MQBMHO (Buffer per le opzioni di gestione dei messaggi) su IBM i

Struttura che definisce il buffer per le opzioni di gestione del messaggio.

Panoramica

Scopo: La struttura MQBMHO consente alle applicazioni di specificare le opzioni che controllano il modo in cui i gestori dei messaggi vengono prodotti dai buffer. La struttura è un parametro di input sulla chiamata MQBUFMH.

Serie di caratteri e codifica: i dati in MQBMHO devono trovarsi nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (ENNAT).

- "Campi" a pagina 1044
- "Valori iniziali" a pagina 1045
- "Dichiarazione RPG" a pagina 1045

Campi

La struttura MQBMHO contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

BMSID (numero intero con segno a 10 cifre)

Buffer nella struttura dell'handle del messaggio - campo StrucId .

Questo è l'identificativo della struttura. Il valore deve essere:

BMSIDV
Identificativo per il buffer nella struttura di gestione del messaggio.

Questo è sempre un campo di input. Il valore iniziale di questo campo è BMSIDV.

BMVER (numero intero con segno a 10 cifre)

Buffer nella struttura dell'handle del messaggio - campo Versione.

Questo è il numero di versione della struttura. Il valore deve essere:

BMVER1
Numero di versione per il buffer nella struttura dell'handle del messaggio.

La seguente costante specifica il numero di versione della versione corrente:

BMVERC
La versione corrente del buffer nella struttura dell'handle del messaggio.

Questo è sempre un campo di input. Il valore iniziale di questo campo è BMVER1.

BMOPT (numero intero con segno a 10 cifre)

Buffer nella struttura dell'handle del messaggio - Campo Opzioni.

Il valore può essere:

BMDLPR
Le proprietà aggiunte all'handle del messaggio vengono eliminate dal buffer. Se la chiamata ha esito negativo, non viene eliminata alcuna proprietà.

Opzioni predefinite: se non è necessaria l'opzione descritta, utilizzare la seguente opzione:

BMNONE
Nessuna opzione specificata.

Questo è sempre un campo di input. Il valore iniziale di questo campo è BMDLPR.

Valori iniziali

Tabella 683. Campi in MQBMHO

Nome campo	Nome della costante	Valore della costante
BMSID	BMSIDV	'BMHO'
BMVER	BMVER1	1
BMOPT	BMNONE	0

Dichiarazione RPG

```
D* MQBMHO Structure
D*
D*
D* Structure identifier
D  BMSID                  1      4    INZ('BMHO')
D*
D* Structure version number
D  BMVER                  5      8I 0 INZ(1)
D*
D* Options that control the action of MQBUFMH
D  BMOPT                  9     12I 0 INZ(1)
```

IBM i MQBO (Opzioni di inizio) su IBM i

La struttura MQBO consente all'applicazione di specificare le opzioni relative alla creazione di un'unità di lavoro.

Panoramica

Scopo: La struttura è un parametro di input / output nella chiamata MQBEGIN.

Serie di caratteri e codifica: i dati in MQBO devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e la codifica del gestore code locale fornita da ENNAT.

- "Campi" a pagina 1046
- "Valori iniziali" a pagina 1046
- "Dichiarazione RPG" a pagina 1047

Campi

La struttura MQBO contiene i campi riportati di seguito; i campi sono descritti in **ordine alfabetico**:

BOOPT (numero intero con segno a 10 cifre)

Opzioni che controllano l'azione di MQBEGIN.

Il valore deve essere:

BONONE
Nessuna opzione specificata.

Questo è sempre un campo di input. Il valore iniziale di questo campo è BONONE.

BOSID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

BOSIDV
Identificativo per la struttura delle opzioni iniziali.

Questo è sempre un campo di input. Il valore iniziale di questo campo è BOSIDV.

BOVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere:

BOVER1
Numero di versione per la struttura delle opzioni iniziali.

La seguente costante specifica il numero di versione della versione corrente:

BOVERC
Versione corrente della struttura delle opzioni di inizio.

Questo è sempre un campo di input. Il valore iniziale di questo campo è BOVER1.

Valori iniziali

Tabella 684. Campi in MQBO

Nome campo	Nome della costante	Valore della costante
BOSID	BOSIDV	'BO¬¬'
BOVER	BOVER1	1
BOOPT	BONONE	0

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQBO Structure
D*
D* Structure identifier
D  BOSID                 1      4    INZ('BO  ')
D* Structure version number
D  BOVER                 5      8I 0 INZ(1)
D* Options that control the action of MQBEGIN
D  BOOPT                 9     12I 0 INZ(0)
```

IBM i MQCBC (Callback context) su IBM i

Struttura che descrive la routine di callback.

Panoramica

Finalità

La struttura di MQCBC viene utilizzata per specificare le informazioni di contesto che vengono trasmesse a una funzione di callback.

La struttura è un parametro di input / output sulla chiamata a una routine del consumatore di messaggi.

Versione

La versione corrente di MQCBC è CBCV2.

Serie di caratteri e codifica

I dati in MQCBC si trovano nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da ENNAT. Tuttavia, se l'applicazione è in esecuzione come client IBM MQ , la struttura si trova nella serie di caratteri e nella codifica del client.

- "Campi" a pagina 1047
- "Valori iniziali" a pagina 1053
- "Dichiarazione RPG" a pagina 1053

Campi

La struttura MQCBC contiene i seguenti campi; i campi sono descritti in ordine alfabetico:

CBCBUFFLEN (numero intero con segno a 10 cifre)

Il buffer può essere maggiore sia del valore di lunghezza MaxMsgdefinito per il consumer che del valore ReturnedLength in MQGMO.

Struttura contesto callback - campo BufferLength .

Questa è la lunghezza in byte del buffer dei messaggi che è stato passato a questa funzione.

La lunghezza effettiva del messaggio viene fornita nel campo DataLength .

L'applicazione può utilizzare l'intero buffer per i propri scopi per la durata della funzione di callback.

Questo è un campo di input per la funzione del destinatario del messaggio; non è rilevante per una funzione del gestore eccezioni.

CBCCALLBA (numero intero con segno a 10 cifre)

Struttura contesto callback - Campo CallbackArea .

Questo è un campo disponibile per la funzione di callback da utilizzare.

Il gestore code non prende alcuna decisione in base al contenuto di questo campo e viene trasmesso invariato dal campo CBDCALLBA nella struttura MQCBD, che è un parametro sulla chiamata MQCB utilizzata per definire la funzione di callback.

Le modifiche a *CBCCALLBA* vengono conservate nelle chiamate della funzione di callback per un *CBCHOBJ*. Questo campo non è condiviso con funzioni di callback per altri handle.

Questo è un campo di immissione / emissione per la funzione di callback. Il valore iniziale di questo campo è un puntatore null o byte null.

CBCCALLT (numero intero con segno a 10 cifre)

Struttura contesto callback - Campo CallType .

Campo contenente informazioni sul motivo per cui questa funzione è stata richiamata. Sono definiti i seguenti tipi di chiamata.

Tipi di chiamata di consegna del messaggio: questi tipi di chiamata contengono informazioni su un messaggio. I parametri **CBCLEN** e **CBCBUFFLEN** sono validi per questi tipi di chiamata.

CBCTMR
La funzione del destinatario del messaggio è stata richiamata con un messaggio che è stato eliminato in maniera distruttiva dall'handle dell'oggetto.

Se il valore di *CBCCC* è CCWARN, il valore del campo *Reason* è RC2079 o uno dei codici che indicano un problema di conversione dati.

CBCTMN
La funzione del destinatario del messaggio è stata richiamata con un messaggio che non è stato ancora eliminato in maniera distruttiva dall'handle dell'oggetto. Il messaggio può essere eliminato in modo distruttivo dall'handle dell'oggetto utilizzando *MsgToken*.

Il messaggio potrebbe non essere stato rimosso perché:

- Le opzioni MQGMO hanno richiesto un'operazione di esplorazione, GMBR*
- Il messaggio è più grande del buffer disponibile e le opzioni MQGMO non specificano gmatm

Se il valore di *CBCCC* è CCWARN, il valore del campo *Reason* è RC2080 o uno dei codici che indicano un problema di conversione dati.

Tipi di chiamata di controllo callback: questi tipi di chiamata contengono informazioni sul controllo del callback e non contengono dettagli su un messaggio. Questi tipi di chiamata sono richiesti utilizzando CBDOPT nella struttura MQCBD.

I parametri **CBCLEN** e **CBCBUFFLEN** non sono validi per questi tipi di chiamata.

CBCTRC
Lo scopo di questo tipo di chiamata è consentire alla funzione di callback di eseguire alcune impostazioni iniziali.

La funzione di callback viene richiamata immediatamente dopo la registrazione del callback, ovvero al ritorno da una chiamata MQCB utilizzando un valore per il campo *Operation* di CBREG.

Questo tipo di chiamata viene utilizzato sia per i consumer di messaggi che per i gestori eventi.

Se richiesto, questo è il primo richiamo della funzione di callback.

Il valore del campo *CBCREA* è RCNONE.

CBCTSC
Lo scopo di questo tipo di chiamata è quello di consentire alla funzione di callback di eseguire alcune operazioni di impostazione quando viene avviata, ad esempio, reintegrando le risorse che erano state ripulite quando era stata precedentemente arrestata.

La funzione di callback viene richiamata quando la connessione viene avviata utilizzando CTLSR o CTLSW.

Se una funzione di callback è registrata all'interno di un'altra funzione di callback, questo tipo di chiamata viene richiamato quando viene restituito il callback.

Questo tipo di chiamata viene utilizzato solo per gli utenti del messaggio.

Il valore del campo *CBCREA* è RCNONE.

CBCTTC
Lo scopo di questo tipo di chiamata è quello di consentire alla funzione di callback di eseguire una ripulitura quando viene arrestata per un certo periodo, ad esempio, ripulendo le risorse aggiuntive che sono state acquisite durante l'utilizzo dei messaggi.

La funzione di callback viene richiamata quando viene emessa una chiamata MQCTL utilizzando un valore per il campo *Operation* di CTLSP.

Questo tipo di chiamata viene utilizzato solo per gli utenti del messaggio.

Il valore del campo *CBCREA* è impostato per indicare il motivo dell'arresto.

CBCTDC
Lo scopo di questo tipo di chiamata è di consentire alla funzione di callback di eseguire la ripulitura finale alla fine del processo di consumo. La funzione callback viene richiamata quando:

- La funzione di callback viene annullata utilizzando una chiamata MQCB con BCUNR.
- La coda è chiusa, causando l'annullamento implicito della registrazione. In questa istanza la funzione callback viene passata HOUNUH come handle dell'oggetto.
- La chiamata MQDISC viene completata - causando una chiusura implicita e, quindi, un annullamento della registrazione. In questo caso, la connessione non viene disconnessa immediatamente e non viene ancora eseguito il commit di tutte le transazioni in corso.

Se una di queste azioni viene eseguita all'interno della funzione di callback, l'azione viene richiamata una volta restituita la callback.

Questo tipo di chiamata viene utilizzato sia per i consumer di messaggi che per i gestori eventi.

Se richiesto, questo è l'ultimo richiamo della funzione di callback.

Il valore del campo *CBCREA* è impostato per indicare il motivo dell'arresto.

CBCTEC
Funzione del gestore eventi

La funzione gestore eventi è stata richiamata senza un messaggio quando:

- Una chiamata MQCTL viene emessa con un valore per il campo *Operation* di CTLSP oppure
- Il gestore code o la connessione vengono arrestati o disattivati.

Questa chiamata può essere utilizzata per eseguire l'azione appropriata per tutte le funzioni di callback.

- **Funzione consumatore messaggi**

  La funzione utente del messaggio è stata richiamata senza un messaggio quando è stato rilevato un errore (*CBCCC* = CCFAIL) specifico per l'handle dell'oggetto; ad esempio *CBCREA* code = RC2016 .

Il valore del campo *CBCREA* è impostato per indicare il motivo della chiamata.

Questo è un campo di immissione. CBCTMR e CMCTMN sono applicabili solo alle funzioni del consumatore di messaggi.

CBCCC (numero intero con segno a 10 cifre)

Struttura contesto callback - campo CompCode .

Questo è il codice di completamento. Indica se si sono verificati problemi durante l'utilizzo del messaggio; è uno dei seguenti:

CCOK
Completamento riuscito

AVVCCN
Avvertenza (completamento parziale)

CCNON RIUSCITO
Chiamata non riuscita

Questo è un campo di immissione. Il valore iniziale di questo campo è CCOK.

CBCCONNAREA (numero intero con segno a 10 cifre)

Struttura contesto callback - Campo ConnectionArea .

Questo è un campo disponibile per la funzione di callback da utilizzare.

Il gestore code non prende alcuna decisione in base al contenuto di questo campo e viene trasmesso non modificato dal campo ConnectionArea nella struttura MQCTLO, che è un parametro sulla chiamata MQCTL utilizzata per controllare la funzione di callback.

Tutte le modifiche apportate a questo campo dalle funzioni di callback vengono conservate nei richiami della funzione di callback. Questa area può essere utilizzata per passare informazioni che devono essere condivise da tutte le funzioni di callback. A differenza di *CallbackArea*, questa area è comune a tutte le richiamate per un handle di connessione.

Questo è un campo di immissione e di emissione. Il valore iniziale di questo campo è un puntatore null o byte null.

CBCLEN (numero intero con segno a 10 cifre)

Questa è la lunghezza in byte dei dati dell'applicazione nel messaggio. Se il valore è zero, significa che il messaggio non contiene dati dell'applicazione.

Il campo CBCLEN contiene la lunghezza del messaggio ma non necessariamente la lunghezza dei dati del messaggio passati al consumer. È possibile che il messaggio sia stato troncato. Utilizzare il campo GMRL in MQGMO per determinare la quantità di dati passati al consumer.

Se il codice di errore indica che il messaggio è stato troncato, è possibile utilizzare il campo CBCLEN per determinare la dimensione effettiva del messaggio. Ciò consente di determinare la dimensione del buffer richiesto per contenere i dati del messaggio ed emettere una chiamata MQCB per aggiornare CBDMML in MQCBD con un valore appropriato.

Se viene specificata l'opzione GMCONV, il messaggio convertito potrebbe essere maggiore del valore restituito per DataLength. In questi casi, l'applicazione probabilmente deve emettere una chiamata MQCB per aggiornare CBDMML in MQCBD in modo che sia maggiore del valore restituito dal gestore code per DataLength.

Per evitare problemi di troncamento dei messaggi, specificare MaxMsgLength come CBDFM. Ciò fa sì che il gestore code assegni un buffer per la lunghezza completa del messaggio dopo la conversione dei dati. Tenere presente, tuttavia, che anche se questa opzione viene specificata, è comunque possibile che non sia disponibile memoria sufficiente per elaborare correttamente la richiesta. Le applicazioni devono sempre controllare il codice motivo restituito. Ad esempio, se non è possibile assegnare memoria sufficiente per convertire il messaggio, i messaggi vengono restituiti all'applicazione non convertita.

Questo è un campo di input per la funzione del consumer del messaggio; non è rilevante per una funzione del gestore eventi.

CBCFLG (numero intero con segno a 10 cifre)

Indicatori contenenti informazioni su questo consumer.

È definita la seguente opzione:

CBCFBE

Questo indicatore può essere restituito se una precedente chiamata MQCLOSE utilizzando l'opzione COQSC ha avuto esito negativo con un codice motivo RC2458.

Questo codice indica che viene restituito l'ultimo messaggio di lettura anticipata e che il buffer è ora vuoto. Se l'applicazione emette un'altra chiamata MQCLOSE utilizzando l'opzione COQSC, l'operazione ha esito positivo.

Notare che non è garantito che a un'applicazione venga fornito un messaggio con questo indicatore impostato, poiché potrebbero essere ancora presenti messaggi nel buffer di lettura anticipata che non corrispondono ai criteri di selezione correnti. In questa istanza, la funzione consumer viene richiamata con il codice motivo RC2019 .

Se il buffer di lettura anticipata è vuoto, il consumer viene richiamato con l'indicatore CBCFBE e il codice di errore RC2518.

Questo è un campo di input per la funzione del consumer del messaggio; non è rilevante per una funzione del gestore eventi.

CBCHOBJ (numero intero con segno a 10 cifre)

Struttura contesto callback - campo CBCHOBJ.

Per una chiamata a un utente del messaggio, questo è l'handle per l'oggetto relativo al destinatario del messaggio.

Per un gestore eventi, questo valore è HONONE

L'applicazione può utilizzare questo handle e il token del messaggio nel blocco Get Message Options per richiamare il messaggio se un messaggio non è stato rimosso dalla coda.

Questo è sempre un campo di input. Il valore iniziale di questo campo è HOUNUH

CBCRCD (numero intero con segno a 10 cifre)

CBCRCD indica il tempo di attesa del gestore code prima di tentare la riconnessione. Il campo può essere modificato da un gestore eventi per modificare il ritardo o arrestare completamente la riconnessione.

Utilizzare il campo **CBCRCD** solo se il valore del campo **Reason** nel contesto di callback è RC2545.

Alla voce del gestore eventi, il valore di **CBCRCD** è il numero di millisecondi che il gestore code attenderà prima di effettuare un tentativo di riconnessione. Tabella 685 a pagina 1051 elenca i valori che è possibile impostare per modificare il comportamento del gestore code al ritorno dal gestore eventi.

*Tabella 685. **CBCRCD** values*

Valore	Descrizione
-1	Non effettuare ulteriori tentativi di riconnessione. Viene restituito un errore all'applicazione.
0	Provare a riconnettersi immediatamente.
>0	Attendere questo numero di millisecondi prima di ritentare la connessione.

CBCREA (numero intero con segno a 10 cifre)

Struttura del contesto di callback - campo Motivo.

Questo è il codice di errore che qualifica *CBCCC*

Questo è un campo di immissione. Il valore iniziale di questo campo è RCNONE.

CBCSTATE (numero intero con segno a 10 cifre)

Un'indicazione dello stato del consumer corrente. Questo campo è di maggior valore per un'applicazione quando un codice di errore diverso da zero viene passato alla funzione consumer.

È possibile utilizzare questo campo per semplificare la programmazione dell'applicazione poiché non è necessario codificare il comportamento per ciascun codice di errore.

Questo è un campo di immissione. Il valore iniziale di questo campo è CSNONE

Tabella 686. Valori CBCSTATE e azioni risultanti

Stato	**Azione gestore code**	**Valore della costante**
CSNONE Questo codice di errore rappresenta una normale chiamata senza ulteriori informazioni sul motivo	Nessuna; questa è l'operazione normale.	0
CSSUST Questi codici di errore rappresentano condizioni temporanee.	La routine di callback viene richiamata per segnalare la condizione e quindi sospesa. Dopo un periodo di tempo, il sistema potrebbe tentare di nuovo l'operazione, il che potrebbe portare alla riattivazione della stessa condizione.	1
CSSUSU Questi codici di errore rappresentano le condizioni in cui il callback deve agire per risolvere la condizione.	Il consumer è sospeso e la routine di callback viene richiamata per notificare la condizione. La routine di callback deve risolvere la condizione se possibile e RESUME o chiudere la connessione.	2
CSSUS Questi codici di errore rappresentano errori che impediscono ulteriori callback del messaggio.	Il gestore code sospende automaticamente la funzione di callback. Se la funzione di richiamata viene ripresa, è probabile che riceva di nuovo lo stesso codice di errore.	3
CSSTOP Questi codici di errore rappresentano la fine del consumo del messaggio.	Consegnato al gestore delle eccezioni e ai callback che specificano CBDTC. Non è possibile utilizzare ulteriori messaggi.	4

CBCSID (numero intero con segno a 10 cifre)

Struttura contesto callback - campo StrucId .

Questo è l'identificatore della struttura; il valore deve essere:

CBCSI
Identificativo per la struttura del contesto di callback.

Questo è sempre un campo di input. Il valore iniziale di questo campo è CBCSI.

CBCVER (numero intero con segno a 10 cifre)

Struttura del contesto di callback - Campo Versione.

Questo è il numero di versione della struttura; il valore deve essere:

CBCV1
Struttura del contesto di callback Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

CBCV
La versione corrente della struttura del contesto di callback.

Questo è sempre un campo di input. Il valore iniziale di questo campo è CBCV1.

Valori iniziali

Tabella 687. Campi in MQCBC		
Nome campo	**Nome della costante**	**Valore della costante**
CBCSID	CBCSI	'CBC¬'
CBCVER	CBCV1	1
CBCCALLT	Nessuna	0
CBCHOBJ	HOUNUH	-1
CBCCALLBA	Nessuna	Puntatore null o byte null
CBCCONNAREA	Nessuna	Puntatore null o byte null
CBCCC	CCOK	0
CBCREA	RCNONE	0
CBCSTATE	CSNONE	0
CBCLEN	Nessuna	0
CBCBUFFLEN	Nessuna	0
CBCFLG	Nessuna	0
CBCRCD	Nessuna	0

Nota:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D* MQCBC Structure
D*
D*
D* Structure identifier
D  CBCSID                 1      4    INZ('CBC ')
D*
D* Structure version number
D  CBCVER                 5      8I 0 INZ(1)
D*
D* Why Function was called
D  CBCCALLT               9     12I 0 INZ(0)
D*
D* Object Handle
D  CBCHOBJ               13     16I 0 INZ(-1)
D*
D* Callback data passed to the function
D  CBCCALLBA             17     32*   INZ(*NULL)
D*
D* MQCTL Data area passed to the function
D  CBCCONNAREA           33     48*   INZ(*NULL)
D*
D* Completion Code
D  CBCCC                 49     52I 0 INZ(0)
D*
D* Reason Code
D  CBCREA                53     56I 0 INZ(0)
D*
D* Consumer State
D  CBCSTATE              57     60I 0 INZ(0)
D*
D* Message Data Length
D  CBCLEN                61     64I 0 INZ(0)
```

```
     D*
     D* Buffer Length
     D  CBCBUFFLEN            65     68I 0 INZ(0)
     D*
     ** Flags containing information about
     D* this consumer
     D  CBCFLG                69     72I 0 INZ(0)
     D* Ver:1 **
     D* Number of milliseconds before reconnect attempt
     D  CBCRCD                73     76I 0 INZ(0)
     D* Ver:2 **
     D*
```

IBM i MQCBD (Descrittore di callback) su IBM i

Struttura che specifica la funzione di callback.

Panoramica

Scopo: La struttura di MQCBD viene utilizzata per specificare una funzione di callback e le opzioni che ne controllano l'utilizzo da parte del gestore code.

La struttura è un parametro di immissione nella chiamata MQCB.

Versione: la versione corrente di MQCBD è CBDV1.

Serie di caratteri e codifica: i dati in MQCBD devono essere nella serie di caratteri e nella codifica del gestore code locale; questi sono forniti dall'attributo del gestore code **CodedCharSetId** e da ENNAT. Tuttavia, se l'applicazione è in esecuzione come IBM MQ MQI client, la struttura deve essere nella serie di caratteri e nella codifica del client.

- "Campi" a pagina 1054
- "Valori iniziali" a pagina 1058
- "Dichiarazione RPG" a pagina 1058

Campi

La struttura MQCBD contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

CBDCALLBA (numero intero con segno a 10 cifre)

Questo è un campo disponibile per la funzione di callback da utilizzare.

Il gestore code non prende alcuna decisione in base al contenuto di questo campo e viene trasmesso invariato dal campo CBCCALLBA nella struttura MQCBD, che è un parametro nella dichiarazione di funzione di callback.

Il valore viene utilizzato solo su un *Operation* che ha un valore CBREG, senza callback attualmente definito, non sostituisce una definizione precedente.

Questo è un campo di input e output per la funzione di callback. Il valore iniziale di questo campo è un puntatore null o byte null.

CBDCALLBF (numero intero con segno a 10 cifre)

La funzione callback viene richiamata come una chiamata di funzione.

Utilizzare questo campo per specificare un puntatore alla funzione callback.

È necessario specificare *CallbackFunction* o *CallbackName*. Se si specificano entrambi, viene restituito il codice di errore RC2486 .

Se non è impostato né *CallbackName* né *CallbackFunction* , la chiamata ha esito negativo con il codice di errore RC2486.

Questa opzione non è supportata nei seguenti ambienti:

- CICS su z/OS

- Linguaggi di programmazione e compilatori che non supportano i riferimenti function - pointer

In tali situazioni, la chiamata ha esito negativo con il codice di errore RC2486.

Questo è un campo di immissione. Il valore iniziale di questo campo è un puntatore null o byte null.

CBDCALLBN (numero intero con segno a 10 cifre)

La funzione callback viene richiamata come un programma collegato dinamicamente.

È necessario specificare *CallbackFunction* o *CallbackName*. Se si specificano entrambi, viene restituito il codice di errore RC2486 .

Se *CallbackName* o *CallbackFunction* non è true, la chiamata ha esito negativo con il codice di errore RC2486.

Il modulo viene caricato quando viene registrata la prima routine di callback da utilizzare e scaricato quando viene annullata la registrazione dell'ultima routine di callback da utilizzare.

Tranne dove indicato nel testo seguente, il nome è allineato a sinistra all'interno del campo, senza spazi vuoti incorporati; il nome stesso viene riempito con spazi vuoti fino alla lunghezza del campo. Nelle descrizioni che seguono, le parentesi quadre ([]) indicano informazioni facoltative:

IBMi

Il nome callback può essere uno dei formati seguenti:

- Programma "/" libreria
- Libreria "/" ServiceProgram "("FunctionName")"

Ad esempio, `MyLibrary/MyProgram(MyFunction)`.

Il nome della libreria può essere *LIBL. I nomi della libreria e del programma sono limitati ad un massimo di 10 caratteri.

AIX and Linux

Il nome callback è il nome di un modulo o di una libreria caricabili dinamicamente, con il suffisso del nome di una funzione che risiede in tale libreria. Il nome della funzione deve essere racchiuso tra parentesi. Il nome della libreria può essere facoltativamente preceduto da un percorso di directory:

```
[path]library(function)
```

Se il percorso non viene specificato, viene utilizzato il percorso di ricerca del sistema.

Il nome è limitato a un massimo di 128 caratteri.

Windows

Il nome di callback è il nome di una libreria di collegamento dinamico, con suffisso il nome di una funzione che risiede in tale libreria. Il nome della funzione deve essere racchiuso tra parentesi. Il nome della libreria può essere facoltativamente preceduto da un percorso di directory e da un'unità:

```
[d:][path]library(function)
```

Se l'unità e il percorso non sono specificati, viene utilizzato il percorso di ricerca del sistema.

Il nome è limitato a un massimo di 128 caratteri.

z/OS

Il nome di callback è il nome di un modulo di caricamento valido per la specifica sul parametro EP della macro LINK o LOAD.

Il nome è limitato a un massimo di 8 caratteri.

z/OS CICS
Il nome di callback è il nome di un modulo di caricamento valido per la specificazione nel parametro PROGRAM della macro del comando EXEC CICS LINK.

Il nome è limitato a un massimo di 8 caratteri.

Il programma può essere definito come remoto utilizzando l'opzione REMOTESYTEM della definizione PROGRAM installata o tramite il programma di instradamento dinamico.

La regione CICS remota deve essere connessa a IBM MQ se il programma deve utilizzare le chiamate API IBM MQ . Si noti, tuttavia, che il campo CBCHOBJ nella struttura MQCBC non è valida in un sistema remoto.

Se si verifica un errore durante il tentativo di caricamento di *CallbackName*, all'applicazione viene restituito uno dei seguenti codici di errore:

- RC2495
- RC2496
- RC2497

Viene anche scritto un messaggio nel log degli errori contenente il nome del modulo per cui è stato tentato il caricamento e il codice di errore dal sistema operativo.

Questo è un campo di immissione. Il valore iniziale di questo campo è una stringa nulla o spazi vuoti.

CBDCALLBT (numero intero con segno a 10 cifre)

È il tipo di funzione di callback. Il valore deve essere uno tra:

CBTMC
Definisce questo callback come funzione di consumatore di messaggi.

Una funzione di callback del destinatario del messaggio viene chiamata quando un messaggio, che soddisfa i criteri di selezione specificati, è disponibile su un handle dell'oggetto e la connessione viene avviata.

CBTEH
Definisce questo callback come routine di eventi asincroni; non viene utilizzato per utilizzare i messaggi per un handle.

Hobj non è richiesto nella chiamata MQCB che definisce il gestore eventi e viene ignorato se specificato.

Il gestore eventi viene chiamato per le condizioni che influenzano l'intero ambiente del consumatore di messaggi. La funzione consumer viene richiamata senza un messaggio quando si verifica un evento, ad esempio un gestore code o l'arresto della connessione o la sospensione. Non viene richiamata per le condizioni che sono specifiche di un singolo consumatore di messaggi, ad esempio RC2016.

Gli eventi vengono consegnati all'applicazione, indipendentemente dal fatto che la connessione sia stata avviata o arrestata, tranne che nei seguenti ambienti:

- Ambiente CICS su z/OS
- applicazioni senza thread

Se il chiamante non inoltra uno di questi valori, la chiamata ha esito negativo con un codice motivo RC2483

Questo è sempre un campo di input. Il valore iniziale di questo campo è CBTMC.

CBDMML (numero intero con segno a 10 cifre)

Questa è la lunghezza in byte del messaggio più lungo che può essere letto dall'handle e fornito alla routine di callback. Se un messaggio ha una lunghezza maggiore, la routine di callback riceve *MaxMsgLength* byte del messaggio e il codice di errore:

- RC2080 o

- RC2079 se è stato specificato GMATM.

La lunghezza effettiva del messaggio viene fornita nel campo "CBCLEN (numero intero con segno a 10 cifre)" a pagina 1050 della struttura MQCBC.

Viene definito il seguente valore speciale:

CBDFM
La lunghezza del buffer viene regolata dal sistema per restituire i messaggi senza troncamento.

Se la memoria disponibile non è sufficiente per assegnare un buffer per ricevere il messaggio, il sistema richiama la funzione di callback con un codice di errore RC2071 .

Se, ad esempio, si richiede la conversione dei dati e la memoria disponibile non è sufficiente per convertire i dati del messaggio, il messaggio non convertito viene passato alla funzione di callback.

Questo è un campo di immissione. Il valore iniziale del campo *MaxMsgLength* è CBDFM.

CBDOPT (numero intero con segno a 10 cifre)

Struttura descrittore callback - Campo Opzioni.

È possibile specificare uno o tutti i seguenti valori. Per specificare più di un'opzione, aggiungere i valori insieme (non aggiungere la stessa costante più di una volta) o combinare i valori utilizzando l'operazione OR bit per bit (se il linguaggio di programmazione supporta le operazioni bit). Le combinazioni non valide sono annotate; tutte le altre combinazioni sono valide.

CBDFQ
La chiamata MQCB ha esito negativo se il gestore code è in stato di inattività.

Su z/OS, questa opzione forza anche la mancata riuscita della chiamata MQCB se la connessione (per un'applicazione CICS o IMS) è in stato di inattività.

Specificare GMFIQ, nelle opzioni MQGMO inoltrate alla chiamata MQCB, per causare la notifica ai consumatori di messaggi quando sono in fase di sospensione.

Opzioni di controllo: le seguenti opzioni controllano se la funzione di callback viene richiamata, senza un messaggio, quando lo stato del consumer cambia:

CBDRC
La funzione callback viene richiamata con il tipo di chiamata CBCTRC

.

CBDSC
La funzione callback viene richiamata con il tipo di chiamata CBCTSC.

CBDTC
La funzione callback viene richiamata con il tipo di chiamata CBCTTC.

CBDDC
La funzione di richiamata viene richiamata con il tipo di chiamata CBCTDC.

Consultare "CBCCALLT (numero intero con segno a 10 cifre)" a pagina 1048 per ulteriori dettagli su questi tipi di chiamata.

Opzione predefinita: se non è necessaria alcuna delle opzioni descritte, utilizzare la seguente opzione:

N. CBD
Utilizzare questo valore per indicare che non sono state specificate altre opzioni; tutte le opzioni assumono i propri valori predefiniti.

CBDNO è definito per aiutare la documentazione del programma; non è previsto che questa opzione venga utilizzata con altre, ma poiché il relativo valore è zero, tale utilizzo non può essere rilevato.

Questo è un campo di immissione. Il valore iniziale del campo *Options* è CBDNO.

CBDSID (numero intero con segno a 10 cifre)

Struttura descrittore callback - campo StrucId .

Questo è l'identificatore della struttura; il valore deve essere:

CBDSI
Identificativo per la struttura del descrittore di callback.

Questo è sempre un campo di input. Il valore iniziale di questo campo è CBDSI.

CBDVER (numero intero con segno a 10 cifre)

Struttura descrittore callback - Campo Versione.

Questo è il numero di versione della struttura; il valore deve essere:

CBDV1
Struttura del descrittore callback Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

CBDCV
Versione corrente della struttura del descrittore di callback.

Questo è sempre un campo di input. Il valore iniziale di questo campo è CBDV1.

Valori iniziali

Tabella 688. Campi in MQCBD

Nome campo	Nome della costante	Valore della costante
StrucId	CBDSI	'CBD¬'
Version	CBDV1	1
CallBackType	CBTMC	1
Options	N. CBD	0
CallbackArea	Nessuna	Byte null
CallbackFunction	Nessuna	Byte null
CallbackName	Nessuna	Spazi
MaxMsgLength	CBDFM	-1

Nota:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D* MQCBD Structure
D*
D*
D* Structure identifier
D  CBDSID                 1      4    INZ('CBD ')
D*
D* Structure version number
D  CBDVER                 5      8I 0 INZ(1)
D*
D* Callback function type
D  CBDCALLBT              9     12I 0 INZ(1)
D*
** Options controlling message
D* consumption
D  CBDOPT                13     16I 0 INZ(0)
D*
```

```
     D* User data passed to the function
     D  CBDCALLBA             17     32*
     D*
     D* FP: Callback function pointer
     D  CBDCALLBF             33     48*
     D*
     D* Callback name
     D  CBDCALLBN             49    176    INZ('\0')
     D*
     D* Maximum message length
     D  CBDMML               177    180I 0 INZ(-1)
```

IBM i MQCHARV (Variable Length String) su IBM i

Utilizzare la struttura MQCHARV per descrivere una stringa di lunghezza variabile.

Panoramica

Serie di caratteri e codifica: i dati in MQCHARV devono essere nella codifica del gestore code locale fornito da ENNAT e nella serie di caratteri del campo VCHRC all'interno della struttura. Se l'applicazione è in esecuzione come un IBM MQ MQI client, la struttura deve essere nella codifica del client. Alcune serie di caratteri hanno una rappresentazione che dipende dalla codifica. Se VCHRC è una di queste serie di caratteri, la codifica utilizzata è la stessa di quella degli altri campi in MQCHARV. La serie di caratteri identificata da VSCCSID può essere una serie di caratteri a doppio byte (DBCS).

Utilizzo: la struttura MQCHARV indirizza i dati che potrebbero essere discontigui con la struttura che li contiene. Per indirizzare questi dati, è possibile utilizzare i campi dichiarati con il tipo di dati puntatore.

- "Campi" a pagina 1059
- "Valori iniziali" a pagina 1060
- "Dichiarazione RPG" a pagina 1061
- "Ridefinizione di CSAPL" a pagina 1061

Campi

La struttura MQCHARV contiene i seguenti campi, descritti in **ordine alfabetico**:

VCHRC (numero intero con segno a 10 cifre)

Questo è l'identificativo della serie di caratteri della stringa di lunghezza variabile indirizzata dal campo VCHRP o VCHRO.

Il valore iniziale di questo campo è CSAPL. Viene definito da IBM MQ per indicare che deve essere modificato dal gestore code con l'identificativo della serie di caratteri true del gestore code. Ciò avviene nello stesso modo in cui CSQM si comporta. Di conseguenza, il valore CSAPL non è mai associato a una stringa di lunghezza variabile. Il valore iniziale di questo campo può essere modificato definendo un valore diverso per la costante CSAPL per l'unità di compilazione mediante i mezzi appropriati per il linguaggio di programmazione dell'applicazione.

VCHRL (numero intero con segno a 10 cifre)

La lunghezza in byte della stringa di lunghezza variabile indicata dal campo VCHRP o VCHRO.

Il valore iniziale di questo campo è 0. Il valore deve essere maggiore o uguale a zero o il seguente valore speciale riconosciuto:

VSNLT

Se VSNLT non viene specificato, i byte VCHRL vengono inclusi come parte della stringa. Se sono presenti caratteri null, non delimitano la stringa.

Se viene specificato VSNLT, la stringa è delimitata dal primo valore null rilevato nella stringa. Il valore null non viene incluso come parte di tale stringa.

Nota: Il carattere null utilizzato per terminare una stringa se viene specificato VSNLT è un valore null dal code set specificato da VCHRC.

Ad esempio, in UTF-16 (CCSID 1200, 13488 e 17584), questa è la codifica Unicode a 2 byte dove un valore null è rappresentato da un numero di 16 bit di tutti zeri. In UTF-16 è comune trovare byte singoli impostati su tutti zero che fanno parte di caratteri (ad esempio caratteri ASCII a 7 bit), ma le stringhe termineranno con un valore null solo quando due byte 'zero' si trovano su un limite di byte pari. È possibile ottenere due byte 'zero' su un limite dispari quando fanno parte di caratteri validi. Ad esempio, x '01' x '00' x '00' x '00' x '30' rappresenta due caratteri Unicode validi e non termina la stringa con valore null.

VCHRO (numero intero con segno a 10 cifre)

L'offset in byte della stringa di lunghezza variabile dall'inizio di MQCHARV o dalla struttura che la contiene.

Quando la struttura MQCHARV è integrata in un'altra struttura, questo valore è l'offset in byte della stringa di lunghezza variabile dall'inizio della struttura che contiene questa struttura MQCHARV. Quando la struttura MQCHARV non è incorporata in un'altra struttura, ad esempio, se viene specificata come parametro in una chiamata di funzione, l'offset è relativo all'avvio della struttura MQCHARV.

L'offset può essere positivo o negativo. È possibile utilizzare il campo VCHRP o VCHRO per specificare la stringa di lunghezza variabile, ma non entrambi.

Il valore iniziale di questo campo è 0.

VCHRP (puntatore)

Questo è un puntatore alla stringa a lunghezza variabile.

È possibile utilizzare il campo VCHRP o VCHRO per specificare la stringa di lunghezza variabile, ma non entrambi.

Il valore iniziale di questo campo è un puntatore null o byte null.

VCHRS (numero intero con segno a 10 cifre)

La dimensione in byte del buffer indirizzato dal campo VCHRP o VCHRO.

Quando la struttura MQCHARV viene utilizzata come un campo di output in una chiamata di funzione, questo campo deve essere inizializzato con la lunghezza del buffer fornito. Se il valore di VCHRL è maggiore di VCHRS, solo i byte di dati VCHRS verranno restituiti al chiamante nel buffer.

Il valore deve essere maggiore o uguale a zero o il seguente valore speciale riconosciuto:

VSUSL
Se viene specificato VSUSL, la lunghezza del buffer viene presa dal campo VCHRL nella struttura MQCHARV. Questo valore speciale non è appropriato quando la struttura viene utilizzata come campo di output e viene fornito un buffer. Questo è il valore iniziale di questo campo.

Valori iniziali

Tabella 689. Valori iniziali MQCHARV per le costanti

Nome campo	Nome della costante	Valore della costante
VCHRP	Nessuna	Puntatore null o byte null.
VCHRO	Nessuna	0
VCHRS	VSUSL	-1
VCHRL	Nessuna	0
VCHRC	CSAPL	-3

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQCHARV Structure
D*
D* Address of variable length string
D VCHRP                   1     16*
D* Offset of variable length string
D VCHRO                  17     20I 0
D* Size of buffer
D VCHRS                  21     24I 0
D* Length of variable length string
D VCHRL                  25     28I 0
D* CCSID of variable length string
D VCHRC                  29     32I 0
```

Ridefinizione di CSAPL

A differenza dei linguaggi di programmazione supportati su altre piattaforme, RPG non ha un modo di ridefinire una costante definita, quindi è necessario impostare ciascun VCHRC in modo specifico se si desidera utilizzare un valore diverso da CSAPL.

IBM i MQCIH (intestazioneCICS bridge) su IBM i

La struttura MQCIH descrive le informazioni che possono essere presenti all'inizio di un messaggio inviato a CICS bridge tramite IBM MQ for z/OS.

Panoramica

Nome formato: FMCICS.

Versione: la versione corrente di MQCIH è CIVER2. I campi che esistono solo nella versione più recente della struttura sono identificati come tali nelle descrizioni che seguono.

Il file COPY fornito contiene la versione più recente di MQCIH, con il valore iniziale del campo *CIVER* impostato su CIVER2.

Serie di caratteri e codifica: condizioni speciali si applicano alla serie di caratteri e alla codifica utilizzata per la struttura MQCIH e i dati del messaggio dell'applicazione:

- Le applicazioni che si connettono al gestore code che possiede la coda CICS bridge devono fornire una struttura MQCIH che si trova nella serie di caratteri e nella codifica del gestore code. Ciò si verifica perché la conversione dei dati della struttura MQCIH non viene eseguita in questo caso.
- Le applicazioni che si connettono ad altri gestori code possono fornire una struttura MQCIH che si trova in una delle serie di caratteri e codifiche supportate; la conversione di MQCIH viene eseguita dall'agent del canale dei messaggi di ricezione connesso al gestore code proprietario della coda CICS bridge .

  Nota: C'è un'eccezione a questo. Se il gestore code proprietario della coda CICS bridge utilizza CICS per l'accodamento distribuito, MQCIH deve essere nella serie di caratteri e nella codifica del gestore code proprietario della coda CICS bridge .
- I dati del messaggio dell'applicazione che seguono la struttura MQCIH devono essere nella stessa serie di caratteri e codifica della struttura MQCIH. i campi *CICSI* e *CIENC* nella struttura MQCIH non possono essere utilizzati per specificare la serie di caratteri e la codifica dei dati del messaggio dell'applicazione.

  Un'uscita di conversione dati deve essere fornita dall'utente per convertire i dati del messaggio dell'applicazione se i dati non sono uno dei formati integrati supportati dal gestore code.

Utilizzo: se i valori richiesti dall'applicazione sono uguali ai valori iniziali mostrati in Tabella 691 a pagina 1071e il bridge è in esecuzione con AUTH=LOCAL o AUTH=IDENTIFY, è possibile omettere la struttura MQCIH dal messaggio. In tutti gli altri casi, la struttura deve essere presente.

Il bridge accetta una struttura MQCIH version-1 o version-2 , ma per le transazioni 3270 deve essere utilizzata una struttura version-2 .

L'applicazione deve garantire che i campi documentati come campi "richiesta" abbiano valori appropriati nel messaggio inviato al bridge; questi campi sono di input per il bridge.

I campi documentati come "risposta" vengono impostati da CICS bridge nel messaggio di risposta che il bridge invia all'applicazione. Le informazioni sugli errori vengono restituite nei campi *CIRET*, *CIFNC*, *CICC*, *CIREA*e *CIAC* , ma non tutte sono impostate in tutti i casi. Tabella 690 a pagina 1062 mostra quali campi sono impostati per i diversi valori di *CIRET*.

Tabella 690. Contenuto dei campi di informazioni sull'errore nella struttura MQCIH

CIRET	***CIFNC***	***CICC***	***CIREA***	***CIAC***
CRC000	-	-	-	-
CRC003	-	-	FBC*	-
CRC002 CRC008	Nome chiamata IBM MQ	IBM MQ *CMPCOD*	IBM MQ *REASON*	-
CRC001 CRC006 CRC007 CRC009	EIBFN CICS	CICS EIBRESP	CICS EIBRESP2	-
CRC004 CRC005	-	-	-	CICS CODICE

- “Campi” a pagina 1062
- “Valori iniziali” a pagina 1071
- “Dichiarazione RPG” a pagina 1072

Campi

La struttura MQCIH contiene i campi riportati di seguito; i campi sono descritti in **ordine alfabetico**:

CIAC (stringa di caratteri a 4 byte)

Codice di interruzione.

Il valore restituito in questo campo è significativo solo se il valore del campo *CIRET* è CRC005 o CRC004. In caso contrario, *CIAC* contiene il valore ABCODE di CICS .

Questo è un campo di risposta. La lunghezza di questo campo è fornita da LNABNC. Il valore iniziale di questo campo è di 4 caratteri vuoti.

Questo è un indicatore che specifica se i descrittori ADS devono essere inviati sulle richieste SEND e RECEIVE BMS. Vengono definiti i seguenti valori:

DNONE
Non inviare o ricevere il descrittore ADS.

ADEND
Invia descrittore ADS.

AGGIUNGI
Ricevi descrittore ADS.

ADMSGF
Utilizzare il formato del messaggio per il descrittore ADS.

Ciò fa sì che il descrittore ADS venga inviato o ricevuto utilizzando la forma estesa del descrittore ADS. Il modulo lungo ha campi allineati su limiti di 4 byte.

Il campo *CIADS* deve essere impostato come segue:

- Se i descrittori ADS non vengono utilizzati, impostare il campo su ADNONE.
- Se i descrittori ADS *vengono* utilizzati e con lo *stesso* CCSID in ogni ambiente, impostare il campo sulla somma di ADSEND e ADRECV.

- Se i descrittori ADS *sono* in uso, ma con CCSID *differenti* in ciascun ambiente, impostare il campo sulla somma di ADSEND, ADRECV e ADMSGF.

Questo è un campo di richiesta utilizzato solo per le transazioni 3270. Il valore iniziale di questo campo è ADNONE.

CIADS (numero intero con segno a 10 cifre)

Descrittore ADS di invio / ricezione.

Questo è un indicatore che specifica se i descrittori ADS devono essere inviati sulle richieste SEND e RECEIVE BMS. Vengono definiti i seguenti valori:

DNONE
Non inviare o ricevere il descrittore ADS.

ADEND
Invia descrittore ADS.

AGGIUNGI
Ricevi descrittore ADS.

ADMSGF
Utilizzare il formato del messaggio per il descrittore ADS.

Ciò fa sì che il descrittore ADS venga inviato o ricevuto utilizzando la forma estesa del descrittore ADS. Il modulo lungo ha campi allineati su limiti di 4 byte.

Il campo *CIADS* deve essere impostato come segue:

- Se i descrittori ADS non vengono utilizzati, impostare il campo su ADNONE.
- Se i descrittori ADS *vengono* utilizzati e con lo *stesso* CCSID in ogni ambiente, impostare il campo sulla somma di ADSEND e ADRECV.
- Se i descrittori ADS *sono* in uso, ma con CCSID *differenti* in ciascun ambiente, impostare il campo sulla somma di ADSEND, ADRECV e ADMSGF.

Questo è un campo di richiesta utilizzato solo per le transazioni 3270. Il valore iniziale di questo campo è ADNONE.

CIAI (stringa di caratteri a 4 byte)

Tasto AID.

Questo è il valore iniziale della chiave AID quando la transazione viene avviata. È un valore a 1 byte, allineato a sinistra.

Questo è un campo di richiesta utilizzato solo per le transazioni 3270. La lunghezza di questo campo è fornita da LNATID. Il valore iniziale di questo campo è di 4 spazi.

CIAUT (stringa di caratteri a 8 byte)

Password o passticket.

Questa è una password o un passticket. Se l'autenticazione dell'identificativo utente è attiva per CICS bridge, *CIAUT* viene utilizzato con l'identificativo utente nel contesto di identità MQMD per autenticare il mittente del messaggio.

Questo è un campo di richiesta. La lunghezza di questo campo è fornita da LNAUTH. Il valore iniziale di questo campo è di 8 spazi.

CICC (numero intero con segno a 10 cifre)

Codice di completamento IBM MQ o CICS EIBRESP.

Il valore restituito in questo campo dipende da *CIRET* ; consultare Tabella 690 a pagina 1062.

Questo è un campo di risposta. Il valore iniziale di questo campo è CCOK.

CICNC (stringa di caratteri a 4 byte)

Codice transazione di fine anomala.

Questo è il codice di interruzione da utilizzare per terminare la transazione (normalmente una transazione conversazionale che richiede più dati). Altrimenti, questo campo è impostato su spazi vuoti.

Questo è un campo di richiesta utilizzato solo per le transazioni 3270. La lunghezza di questo campo è data da LNCNCL. Il valore iniziale di questo campo è di 4 spazi.

CICP (numero intero con segno a 10 cifre)

Posizione del cursore.

Questa è la posizione iniziale del cursore quando la transazione viene avviata. Successivamente, per le transazioni conversazionali, la posizione del cursore si trova nel vettore RECEIVE.

Questo è un campo di richiesta utilizzato solo per le transazioni 3270. Il valore iniziale di questo campo è 0. Questo campo non è presente se *CIVER* è minore di CIVER2.

CICSI (numero intero con segno a 10 cifre)

Riservato.

Questo è un campo riservato; il suo valore non è significativo. Il valore iniziale di questo campo è 0.

CICT (numero intero con segno a 10 cifre)

Indica se l'attività può essere conversazionale.

Si tratta di un indicatore che specifica se l'attività deve essere autorizzata a emettere richieste per ulteriori informazioni o se deve essere terminata in modo anomalo. Il valore deve essere uno dei seguenti.

SÌ
L'attività è colloquiale.

N. CTC
L'attività non è conversazionale.

Questo è un campo di richiesta utilizzato solo per le transazioni 3270. Il valore iniziale di questo campo è CTNO.

CIENC (numero intero con segno a 10 cifre)

Riservato.

Questo è un campo riservato; il suo valore non è significativo. Il valore iniziale di questo campo è 0.

CIEO (numero intero con segno a 10 cifre)

Offset dell'errore nel messaggio.

Questa è la posizione dei dati non validi rilevati dall'uscita bridge. Questo campo fornisce lo scostamento dall'inizio del messaggio alla posizione dei dati non validi.

Questo è un campo di risposta utilizzato solo per le transazioni 3270. Il valore iniziale di questo campo è 0. Questo campo non è presente se *CIVER* è minore di CIVER2.

CIFAC (stringa bit a 8 byte)

Token funzione bridge.

Questo è un token di funzione bridge a 8 byte. Lo scopo di un token della funzione bridge è consentire a più transazioni in una pseudoconversazione di utilizzare la stessa funzione bridge (terminale 3270 virtuale). Nel primo o unico messaggio in una pseudoconversazione, è necessario impostare un valore di FCNONE; ciò indica a CICS di allocare una nuova funzione bridge per questo messaggio. Un token della funzione bridge viene restituito nei messaggi di risposta quando viene specificato un *CIFKT* diverso da zero sul messaggio di input. I messaggi di input successivi possono quindi utilizzare lo stesso token della funzione bridge.

Viene definito il seguente valore speciale:

FCNONE
Nessun token BVT specificato.

Questo è sia un campo di richiesta che un campo di risposta utilizzato solo per le transazioni 3270. La lunghezza di questo campo è fornita da LNFAC. Il valore iniziale di questo campo è FCNONE.

CIFKT (numero intero firmato a 10 cifre)

Ora di rilascio della funzione bridge.

Questo è il periodo di tempo, in secondi, per cui la funzione bridge verrà conservata dopo la fine della transazione utente. Per le transazioni non conversazionali, il valore deve essere zero.

Questo è un campo di richiesta utilizzato solo per le transazioni 3270. Il valore iniziale di questo campo è 0.

CIFL (stringa di caratteri a 4 byte)

Attributi emulati del terminale.

Questo è il nome di un terminale installato che deve essere utilizzato come modello per la funzione bridge. Un valore vuoto indica che *CIFL* viene preso dalla definizione del profilo di transazione bridge oppure viene utilizzato un valore predefinito.

Questo è un campo di richiesta utilizzato solo per le transazioni 3270. La lunghezza di questo campo è fornita da LNFACL. Il valore iniziale di questo campo è di 4 spazi.

CIFLG (numero intero con segno a 10 cifre)

Indicatori.

Il valore deve essere:

CIFNON
Nessun indicatore.

Questo è un campo di richiesta. Il valore iniziale di questo campo è CIFNON.

CIFMT (stringa di caratteri a 8 byte)

Nome formato IBM MQ dei dati che seguono MQCIH.

Specifica il nome formato IBM MQ dei dati che seguono la struttura MQCIH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono le stesse del campo *MDFMT* in MQMD.

Questo nome formato viene utilizzato anche per il messaggio di risposta, se il campo *CIRFM* ha il valore FMNONE.

- Per le richieste DPL, *CIFMT* deve essere il nome formato della COMMAREA.
- Per le richieste 3270, *CIFMT* deve essere CSQCBDCIe *CIRFM* deve essere CSQCBDC0.

Le uscite di conversione dati per questi formati devono essere installate sul gestore code in cui devono essere eseguite.

Se il messaggio di richiesta risulta nella creazione di un messaggio di risposta di errore, il messaggio di risposta di errore ha un nome formato FMSTR.

Questo è un campo di richiesta. La lunghezza di questo campo è fornita da LNFMT. Il valore iniziale di questo campo è FMNONE.

CIFNC (stringa di caratteri a 4 byte)

IBM MQ nome chiamata o CICS funzione EIBFN.

Il valore restituito in questo campo dipende da *CIRET* ; consultare Tabella 690 a pagina 1062. I seguenti valori sono possibili quando *CIFNC* contiene un nome chiamata IBM MQ :

CFCONN
Chiamata MQCONN.

CFGET
Chiamata MQGET.

CFINQ
Chiamata MQINQ.

CFOPEN
Chiamata MQOPEN.

CFPUT
Chiamata MQPUT.

CFPUT1
Chiamata MQPUT1 .

CFNONE
Nessuna chiamata.

Questo è un campo di risposta. La lunghezza di questo campo è fornita da LNFUNC. Il valore iniziale di questo campo è CFNONE.

CIGWI (numero intero con segno a 10 cifre)

Intervallo di attesa per la chiamata MQGET emessa dall'attività bridge.

Questo campo è applicabile solo quando *CIUOW* ha il valore CUFRST. Consente all'applicazione di invio di specificare il tempo approssimativo, in millisecondi, in cui le chiamate MQGET emesse dal bridge devono attendere il secondo e i successivi messaggi di richiesta per l'unità di lavoro avviata da questo messaggio. Questo sostituisce l'intervallo di attesa predefinito utilizzato dal bridge. È possibile utilizzare i seguenti valori speciali:

FLWID
Intervallo di attesa predefinito.

Ciò fa sì che CICS bridge attenda il periodo specificato quando è stato avviato il bridge.

WIULIM
Intervallo di attesa illimitato.

Questo è un campo di richiesta. Il valore iniziale di questo campo è WIDFLT.

CIII (numero intero con segno a 10 cifre)

Riservato.

Questo è un campo riservato. Il valore deve essere 0. Questo campo non è presente se *CIVER* è minore di CIVER2.

CILEN (numero intero con segno a 10 cifre)

Lunghezza della struttura MQCIH.

Il valore deve essere uno dei seguenti.

CILEN1
Lunghezza della struttura dell'intestazione delle informazioni version-1 CICS .

CILEN2
Lunghezza della struttura dell'intestazione delle informazioni version-2 CICS .

La seguente costante specifica la lunghezza della versione corrente:

CILENC
La lunghezza della versione corrente della struttura dell'intestazione delle informazioni CICS .

Questo è un campo di richiesta. Il valore iniziale di questo campo è CILEN2.

CILT (numero intero con segno a 10 cifre)

Tipo di collegamento.

Indica il tipo di oggetto che il bridge deve provare a collegare. Il valore deve essere uno dei seguenti.

PROG LTD
Programma DPL.

LTTRAN
Transazione 3270.

Questo è un campo di richiesta. Il valore iniziale di questo campo è LTPROG.

CINTI (stringa di caratteri a 4 byte)

Transazione successiva da collegare.

Questo è il nome della transazione successiva restituita dalla transazione utente (di solito da EXEC CICS RETURN TRANSID). Se non vi è alcuna transazione successiva, questo campo viene impostato su spazi vuoti.

Questo è un campo di risposta utilizzato solo per le transazioni 3270. La lunghezza di questo campo è fornita da LNTRID. Il valore iniziale di questo campo è di 4 spazi.

CIODL (numero intero con segno a 10 cifre)

Lunghezza dati COMMAREA di emissione.

Questa è la lunghezza dei dati utente da restituire al client in un messaggio di risposta. Tale lunghezza comprende il nome di programma a 8-byte. La lunghezza della COMMAREA inoltrata al programma collegato è il massimo di questo campo e la lunghezza dei dati utente nel messaggio di richiesta, meno 8.

Nota: La lunghezza dei dati utente in un messaggio è la lunghezza del messaggio *escluso* la struttura MQCIH.

Se la lunghezza dei dati utente nel messaggio di richiesta è minore di *CIODL*, viene utilizzata l'opzione DATALENGTH del comando LINK ; ciò consente a LINK di essere spedito in modo efficiente in un'altra regione CICS .

È possibile utilizzare il seguente valore speciale:

OLINPT
La lunghezza di emissione è uguale alla lunghezza di immissione.

Questo valore potrebbe essere necessario anche se non viene richiesta alcuna risposta, per garantire che la COMMAREA passata al programma collegato abbia una dimensione sufficiente.

Questo è un campo di richiesta utilizzato solo per programmi DPL. Il valore iniziale di questo campo OLINPT.

CIREA (numero intero con segno a 10 cifre)

IBM MQ codice motivo o feedback o CICS EIBRESP2.

Il valore restituito in questo campo dipende da *CIRET* ; consultare Tabella 690 a pagina 1062.

Questo è un campo di risposta. Il valore iniziale di questo campo è RCNONE.

CIRET (numero intero con segno a 10 cifre)

Codice di ritorno dal bridge.

Questo è il codice di ritorno da CICS bridge che descrive il risultato dell'elaborazione eseguita dal ponte. I campi *CIFNC*, *CICC*, *CIREA*e *CIAC* possono contenere ulteriori informazioni (consultare Tabella 690 a pagina 1062). Il valore è uno dei seguenti:

CRC000
(0, X'000 ') Nessun errore.

CRC001
(1, X'001 ') L'istruzione EXEC CICS ha rilevato un errore.

CRC002
(2, X'002 ') La chiamata IBM MQ ha rilevato un errore.

CRC003
(3, X'003 ') CICS bridge ha rilevato un errore.

CRC004
(4, X'004 ') CICS bridge terminato in modo anomalo.

CRC005
(5, X'005 ') L'applicazione è terminata in modo anomalo.

CRC006
(6, X'006 ') Si è verificato un errore di sicurezza.

CRC007
(7, X'007 ') Programma non disponibile.

CRC008
(8, X'008 ') Il secondo o il successivo messaggio nell'unità di lavoro corrente non è stato ricevuto entro il tempo specificato.

CRC009
(9, X'009 ') Transazione non disponibile.

Questo è un campo di risposta. Il valore iniziale di questo campo è CRC000.

CIRFM (stringa di caratteri a 8 byte)

IBM MQ nome formato del messaggio di risposta.

Questo è il nome formato IBM MQ del messaggio di replica che verrà inviato in risposta al messaggio corrente. Le regole per la codifica sono le stesse del campo *MDFMT* in MQMD.

Questo è un campo di richiesta utilizzato solo per programmi DPL. La lunghezza di questo campo è fornita da LNFMT. Il valore iniziale di questo campo è FMNONE.

CIRSI (stringa di caratteri a 4 byte)

Riservato.

Questo è un campo riservato. Il valore deve essere di 4 spazi. La lunghezza di questo campo è fornita da LNRSID.

CIRS1 (stringa di caratteri a 8 byte)

Riservato.

Questo è un campo riservato. Il valore deve essere di 8 spazi.

CIRS2 (stringa di caratteri a 8 byte)

Riservato.

Questo è un campo riservato. Il valore deve essere di 8 spazi.

CIRS3 (stringa di caratteri a 8 byte)

Riservato.

Questo è un campo riservato. Il valore deve essere di 8 spazi.

CIRS4 (numero intero con segno a 10 cifre)

Riservato.

Questo è un campo riservato. Il valore deve essere 0. Questo campo non è presente se *CIVER* è minore di CIVER2.

CIRTI (stringa di caratteri a 4 byte)

Riservato.

Questo è un campo riservato. Il valore deve essere di 4 spazi. La lunghezza di questo campo è fornita da LNTRID.

CISC (stringa di caratteri a 4 byte)

Codice di inizio transazione.

Si tratta di un indicatore che specifica se il bridge emula una transazione terminale o una transazione START. Il valore deve essere uno dei seguenti.

STRINGA
Avvio.

DATI
Avviare i dati.

SCTERM
Termina input.

SCNONE
Nessuna.

Nella risposta dal bridge, questo campo è impostato sul codice di avvio appropriato per l'ID transazione successivo contenuto nel campo *CINTI* . I seguenti codici di avvio sono possibili nella risposta:

- STRINGA
- DATI
- SCTERM

Per CICS Transaction Server 1.2 questo campo è solo un campo di richiesta; il suo valore nella risposta non è definito.

Per CICS Transaction Server 1.3 e release successive, questo è sia un campo di richiesta che un campo di risposta.

Questo campo viene utilizzato solo per transazioni 3270. La lunghezza di questo campo è fornita da LNSTCO. Il valore iniziale di questo campo è SCNONE.

CISID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

IDCISV
Identificativo per la struttura dell'intestazione delle informazioni CICS .

Questo è un campo di richiesta. Il valore iniziale di questo campo è CISIDV.

CITES (numero intero con segno a 10 cifre)

Stato alla fine dell'attività.

Questo campo mostra lo stato della transazione utente alla fine dell'attività. Viene restituito uno dei seguenti valori:

TENOSA
Non sincronizzato.

La transazione utente non è stata ancora completata e non è stata sincronizzata. Il campo *MDMT* in MQMD è MTRQST in questo caso.

TECMIT
Commit unità di lavoro.

La transazione utente non è stata ancora completata, ma ha sincronizzato la prima unità di lavoro. Il campo *MDMT* in MQMD è MTDGRM in questo caso.

TEBACK
Backout unità di lavoro.

La transazione utente non è stata ancora completata. L'unità di lavoro corrente verrà ripristinata. Il campo *MDMT* in MQMD è MTDGRM in questo caso.

TEENDT
Termina attività.

La transazione utente è terminata (o è terminata in modo anomalo). Il campo *MDMT* in MQMD è MTRPLY in questo caso.

Questo è un campo di risposta utilizzato solo per le transazioni 3270. Il valore iniziale di questo campo è TENOSY.

CITI (stringa di caratteri a 4 byte)

Transazione da collegare.

Se *CILT* ha il valore LTTRAN, *CITI* è l'identificatore di transazione della transazione utente da eseguire; in questo caso è necessario specificare un valore non vuoto.

Se *CILT* ha il valore LTPROG, *CITI* è il codice transazione sotto il quale devono essere eseguiti tutti i programmi all'interno dell'unità di lavoro. Se il valore specificato è vuoto, viene utilizzato il CKBP (CICS DPL bridge default transaction code). Se il valore non è vuoto, deve essere stato definito in CICS come TRANSACTION locale con un programma iniziale di CSQCBP00. Questo campo è applicabile solo quando il valore di *CIUOW* è CUFRST o CUONLY.

Questo è un campo di richiesta. La lunghezza di questo campo è fornita da LNTRID. Il valore iniziale di questo campo è di 4 spazi.

CIUOW (numero intero con segno a 10 cifre)

Controllo unità di lavoro.

Controlla l'elaborazione dell'unità di lavoro eseguita da CICS bridge. È possibile richiedere al bridge di eseguire una singola transazione o uno o più programmi all'interno di un'unità di lavoro. Il campo indica se CICS bridge deve avviare un'unità di lavoro, eseguire la funzione richiesta all'interno dell'unità di lavoro corrente o terminare l'unità di lavoro eseguendone il commit o il backout. Sono supportate varie combinazioni, per ottimizzare i flussi di trasmissione dati.

Il valore deve essere uno dei seguenti.

CUONLY
Avviare l'unità di lavoro, eseguire la funzione, quindi eseguire il commit dell'unità di lavoro (DPL e 3270).

CONT
Ulteriori dati per l'unità di lavoro corrente (solo 3270).

CUFRST
Avviare l'unità di lavoro ed eseguire la funzione (solo DPL).

IDCMUM
Eseguire la funzione nell'unità di lavoro corrente (solo DPL).

CULAST
Eseguire la funzione, quindi eseguire il commit dell'unità di lavoro (solo DPL).

CUCMIT
Eseguire il commit dell'unità di lavoro (solo DPL).

CUBACK
Ripristinare l'unità di lavoro (solo DPL).

Questo è un campo di richiesta. Il valore iniziale di questo campo è CUONLY.

CIVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere uno dei seguenti.

CIVER1
Version-1 CICS struttura dell'intestazione delle informazioni.

CIVER2
Version-2 CICS struttura dell'intestazione delle informazioni.

I campi esistenti solo nella versione più recente della struttura vengono identificati come tali nelle descrizioni dei campi. La seguente costante specifica il numero di versione della versione corrente:

CIVERC
Versione corrente della struttura dell'intestazione delle informazioni CICS .

Questo è un campo di richiesta. Il valore iniziale di questo campo è CIVER2.

Valori iniziali

Tabella 691. Campi in MQCIH

Nome campo	Nome della costante	Valore della costante
CISID	IDCISV	'CIH¬'
CIVER	CIVER2	2
CILEN	CILEN2	180
CIENC	Nessuna	0
CICSI	Nessuna	0
CIFMT	FMNONE	Spazi
CIFLG	CIFNON	0
CIRET	CRC000	0
CICC	CCOK	0
CIREA	RCNONE	0
CIUOW	CUONLY	273
CIGWI	FLWID	-2
CILT	PROG LTD	1
CIODL	OLINPT	-1
CIFKT	Nessuna	0
CIADS	DNONE	0
CICT	N. CTC	0
CITES	TENOSA	0
CIFAC	FCNONE	Valori null
CIFNC	CFNONE	Spazi
CIAC	Nessuna	Spazi
CIAUT	Nessuna	Spazi
CIRS1	Nessuna	Spazi

Tabella 691. Campi in MQCIH (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
CIRFM	FMNONE	Spazi
CIRSI	Nessuna	Spazi
CIRTI	Nessuna	Spazi
CITI	Nessuna	Spazi
CIFL	Nessuna	Spazi
CIAI	Nessuna	Spazi
CISC	SCNONE	Spazi
CICNC	Nessuna	Spazi
CINTI	Nessuna	Spazi
CIRS2	Nessuna	Spazi
CIRS3	Nessuna	Spazi
CICP	Nessuna	0
CIEO	Nessuna	0
CIII	Nessuna	0
CIRS4	Nessuna	0

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQCIH Structure
D*
D* Structure identifier
D  CISID                  1      4    INZ('CIH ')
D* Structure version number
D  CIVER                  5      8I 0 INZ(2)
D* Length of MQCIH structure
D  CILEN                  9     12I 0 INZ(180)
D* Reserved
D  CIENC                 13     16I 0 INZ(0)
D* Reserved
D  CICSI                 17     20I 0 INZ(0)
D* MQ format name of data that followsMQCIH
D  CIFMT                 21     28    INZ('        ')
D* Flags
D  CIFLG                 29     32I 0 INZ(0)
D* Return code from bridge
D  CIRET                 33     36I 0 INZ(0)
D* MQ completion code or CICSEIBRESP
D  CICC                  37     40I 0 INZ(0)
D* MQ reason or feedback code, or CICSEIBRESP2
D  CIREA                 41     44I 0 INZ(0)
D* Unit-of-work control
D  CIUOW                 45     48I 0 INZ(273)
D* Wait interval for MQGET call issuedby bridge task
D  CIGWI                 49     52I 0 INZ(-2)
D* Link type
D  CILT                  53     56I 0 INZ(1)
D* Output COMMAREA data length
D  CIODL                 57     60I 0 INZ(-1)
D* Bridge facility release time
D  CIFKT                 61     64I 0 INZ(0)
```

```
     D* Send/receive ADS descriptor
     D  CIADS                 65     68I 0 INZ(0)
     D* Whether task can beconversational
     D  CICT                  69     72I 0 INZ(0)
     D* Status at end of task
     D  CITES                 73     76I 0 INZ(0)
     D* Bridge facility token
     D  CIFAC                 77     84    INZ(X'00000000000000-
     D                                     00')
     D* MQ call name or CICS EIBFNfunction
     D  CIFNC                 85     88    INZ('    ')
     D* Abend code
     D  CIAC                  89     92    INZ
     D* Password or passticket
     D  CIAUT                 93    100    INZ
     D* Reserved
     D  CIRS1                101    108    INZ
     D* MQ format name of reply message
     D  CIRFM                109    116    INZ('        ')
     D* Remote CICS system ID to use
     D  CIRSI                117    120    INZ
     D* CICS RTRANSID to use
     D  CIRTI                121    124    INZ
     D* Transaction to attach
     D  CITI                 125    128    INZ
     D* Terminal emulated attributes
     D  CIFL                 129    132    INZ
     D* AID key
     D  CIAI                 133    136    INZ
     D* Transaction start code
     D  CISC                 137    140    INZ('    ')
     D* Abend transaction code
     D  CICNC                141    144    INZ
     D* Next transaction to attach
     D  CINTI                145    148    INZ
     D* Reserved
     D  CIRS2                149    156    INZ
     D* Reserved
     D  CIRS3                157    164    INZ
     D* Cursor position
     D  CICP                 165    168I 0 INZ(0)
     D* Offset of error in message
     D  CIEO                 169    172I 0 INZ(0)
     D* Reserved
     D  CIII                 173    176I 0 INZ(0)
     D* Reserved
     D  CIRS4                177    180I 0 INZ(0)
     D*
```

IBM i MQCMHO (Create message handle options) su IBM i

La struttura **MQCMHO** consente alle applicazioni di specificare le opzioni che controllano il modo in cui vengono creati gli handle del messaggio.

Panoramica

Finalità
La struttura è un parametro di immissione sulla chiamata **MQCRTMH** .

Serie di caratteri e codifica
I dati in **MQCMHO** devono essere nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (ENNAT).

- “Campi” a pagina 1073
- “Valori iniziali” a pagina 1075
- “Dichiarazione RPG” a pagina 1075

Campi

La struttura MQCMHO contiene i seguenti campi; i campi sono descritti in ordine alfabetico:

CMOPT (numero intero con segno a 10 cifre)

È possibile specificare una delle seguenti opzioni:

CMVAL

Quando **MQSETMP** viene chiamato per impostare una proprietà in questo handle del messaggio, il nome della proprietà viene convalidato per garantire che:

- non contiene caratteri non validi.
- non inizia con "JMS" o "usr.JMS" ad eccezione di quanto segue:
  - JMSCorrelationID
  - JMSReplyTo
  - JMSType
  - JMSXGroupID
  - JMSXGroupSeq

  Questi nomi sono riservati per proprietà JMS .
- non è una delle seguenti parole chiave, in qualsiasi combinazione di lettere maiuscole o minuscole:
  - "E"
  - "COMPRESO"
  - "ESCAPE"
  - "FALSO"
  - "IN"
  - "È"
  - "come"
  - "NON"
  - "NULL"
  - "O"
  - "VERO"
- non inizia con "Body". o "Root." (eccetto per "Root.MQMD.").

Se la proprietà è definita da MQ("mq. *") e il nome viene riconosciuto, i campi descrittore della proprietà sono impostati sui valori corretti per la proprietà. Se la proprietà non viene riconosciuta, il campo *Support* del descrittore della proprietà è impostato su **PDSUPO** (per ulteriori informazioni, consultare PDSUP).

CMDEFV

Specifica che si verifica il livello predefinito di convalida dei nomi delle proprietà.

Il livello predefinito di convalida è equivalente a quello specificato da **CMVAL**.

In una release futura potrebbe essere definita un'opzione amministrativa che modificherà il livello di convalida che si verificherà quando viene definito **CMDEFV** .

Questo è il valore predefinito.

NOVA CM

Non viene eseguita alcuna convalida sul nome della proprietà. Consultare la descrizione di **CMVAL**.

Opzione predefinita: se non è richiesta alcuna delle opzioni precedentemente descritte in questa sezione, è possibile utilizzare la seguente opzione:

CMNONE

Tutte le opzioni assumono i valori predefiniti. Utilizzare questo valore per indicare che non sono state specificate altre opzioni. **CMNONE** supporta la documentazione del programma; non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **CMDEFV**.

CMSID (numero intero con segno a 10 cifre)

Questo è l'identificatore della struttura; il valore deve essere:

CSID V
Identificativo per la struttura delle opzioni di gestione del messaggio.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **CMSIDV**.

CMVER (numero intero con segno a 10 cifre)

Questo è il numero di versione della struttura; il valore deve essere:

CMVER1
Version-1 creare la struttura di opzioni dell'handle del messaggio.

La seguente costante specifica il numero di versione della versione corrente:

CMVERC
Versione corrente della struttura delle opzioni di creazione dell'handle del messaggio.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **CMVER1**.

Valori iniziali

Tabella 692. Campi in MQCMHO

Nome campo	Nome della costante	Valore della costante
CMSID	CSID V	'CMHO'
CMVER	CMVER1	1
CMOPT	CMDEFV	0

Dichiarazione RPG

```
D* MQCMHO Structure
D*
D*
D* Structure identifier
D  CMSID                  1      4    INZ('CMHO')
D*
D* Structure version number
D  CMVER                  5      8I 0 INZ(1)
D*
D* Options that control the action of MQCRTMH
D  CMOPT                  9     12I 0 INZ(0)
```

IBM i MQCNO (Opzioni di connessione) su IBM i

La struttura MQCNO consente all'applicazione di specificare le opzioni relative alla connessione al gestore code locale.

Panoramica

Scopo: la struttura è un parametro di input / output sulla chiamata MQCONNX.

Versione: la versione corrente di MQCNO è CNVER6. I campi che esistono solo nelle versioni più recenti della struttura sono identificati come tali nelle descrizioni che seguono.

Il file COPY fornito contiene la versione più recente di MQCNO supportata dall'ambiente, ma con il valore iniziale del campo CNVER impostato su CNVER1. Per utilizzare i campi che non sono presenti nella struttura version-1 , l'applicazione deve impostare il campo CNVER sul numero di versione della versione richiesta.

Serie di caratteri e codifica: i dati in MQCNO devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e la codifica del gestore code locale fornita da ENNAT.

- "Campi" a pagina 1076
- "Valori iniziali" a pagina 1081
- "Dichiarazione RPG" a pagina 1082

Campi

La struttura MQCNO contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

CCDTUL (numero intero con segno a 10 cifre)

CCDTUL è la lunghezza della stringa identificata da CCDTUP o CCDTUO che contiene un URL che identifica la posizione della tabella del canale di connessione client da utilizzare per la connessione.

Utilizzare CCDTUL solo quando l'applicazione che emette la chiamata MQCONNX è in esecuzione come IBM MQ MQI client.

Questa è un'alternativa programmatica all'impostazione delle variabili di ambiente MQCHLLIB e MQCHLTAB .

Se l'applicazione non è in esecuzione come client, CCDTUL viene ignorato.

Questo campo viene ignorato se CNVER è inferiore a CNVER6.

CCDTUO (numero intero con segno a 10 cifre)

CCDTUO è lo scostamento in byte, dall'inizio della struttura MQCNO, a una stringa che contiene un URL che identifica la posizione della tabella del canale di connessione client da utilizzare per la connessione. L'offset può essere positivo o negativo.

Utilizzare CCDTUL solo quando l'applicazione che emette la chiamata MQCONNX è in esecuzione come IBM MQ MQI client.

Importante: È possibile utilizzare solo uno tra CCDTUP e CCDTUO. La chiamata non riesce con codice di errore RC2600 se entrambi i campi sono diversi da zero.

Questa è un'alternativa programmatica all'impostazione delle variabili di ambiente MQCHLLIB e MQCHLTAB .

Se l'applicazione non è in esecuzione come client, CCDTUO viene ignorato.

Questo campo viene ignorato se CNVER è inferiore a CNVER6.

CCDTUP (puntatore)

CCDTUP è un puntatore facoltativo a una stringa che contiene un URL, per identificare l'ubicazione della tabella del canale di connessione client da utilizzare per la connessione.

Utilizzare CCDTUP solo quando l'applicazione che emette la chiamata MQCONNX viene eseguita come IBM MQ MQI client.

Importante: È possibile utilizzare solo uno tra CCDTUP e CCDTUO. La chiamata non riesce con codice di errore RC2600 se entrambi i campi sono diversi da zero.

Questa è un'alternativa programmatica all'impostazione delle variabili di ambiente MQCHLLIB e MQCHLTAB .

Se l'applicazione non è in esecuzione come client, CCDTUP viene ignorato.

Questo campo viene ignorato se CNVER è inferiore a CNVER6.

CNAN (stringa di caratteri a 28 byte)

Il nome impostato dall'applicazione per identificare il collegamento al gestore code. Il valore iniziale del campo è di caratteri null.

Questo campo viene ignorato se CNVER è inferiore a CNVER7.

CNCCO (numero intero con segno a 10 cifre)

È l'offset in byte di una struttura di definizione di canale MQCD dall'inizio della struttura MQCNO.

CNCCP (puntatore)

Questo è un puntatore ad una struttura di definizione di canale MQCD.

CNCONID (stringa di caratteri a 24 byte)

Identificativo di collegamento univoco. Questo campo consente al gestore code di identificare in modo affidabile un processo dell'applicazione assegnandogli un identificativo univoco alla prima connessione al gestore code.

Le applicazioni utilizzano l'identificativo di connessione per scopi di correlazione quando effettuano chiamate PUT e GET. A tutte le connessioni viene assegnato un identificativo dal gestore code, indipendentemente da come è stata stabilita la connessione.

È possibile utilizzare l'identificativo di connessione per forzare la fine di un'unità di lavoro di lunga durata. A tale scopo, specificare l'identificativo di connessione utilizzando il comando PCF 'Arresta connessione' o il comando MQSC STOP CONN. Per ulteriori informazioni sull'utilizzo di questi comandi, consultare i collegamenti correlati.

Il valore iniziale del campo è di 24 byte null.

CNCT (stringa bit a 128 byte)

Si tratta di una tag che il gestore code associa alle risorse interessate dall'applicazione durante questa connessione.

Tag di connessione gestore code.

Ogni applicazione o istanza dell'applicazione deve utilizzare un valore diverso per la tag, in modo che il gestore code possa serializzare correttamente l'accesso alle risorse interessate. Per ulteriori dettagli, consultare le descrizioni delle opzioni CN* CT*. La tag cessa di essere valida quando l'applicazione termina o emette la chiamata MQDISC.

Utilizzare il seguente valore speciale se non è richiesta alcuna tag:

CTNONE
Nessuna tag di collegamento specificata.

Il valore è zero binario per la lunghezza del campo.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da LNCTAG. Il valore iniziale di questo campo è CTNONE. Questo campo viene ignorato se CNVER è inferiore a CNVER3.

Utilizzare il campo ConnTag quando ci si connette a un gestore code z/OS .

CNNORES2 (stringa di caratteri a 4 byte)

Un campo riservato per inserire la struttura in un limite a 64 bit. Il valore iniziale del campo è zero binario per la lunghezza del campo.

Questo campo viene ignorato se CNVER è inferiore a CNVER7.

CNOPT (numero intero con segno a 10 cifre)

Opzioni che controllano l'operazione di MQCONNX.

Opzioni di bind

Le opzioni di collegamento controllano il tipo di collegamento IBM MQ utilizzato; specificarne solo una:

CNSBND
Collegamento standard.

L'opzione di bind standard fa in modo che l'applicazione e l'agente del gestore code locale vengano eseguiti in unità di esecuzione separate, generalmente in processi separati. La disposizione mantiene l'integrità del gestore code, ossia protegge il gestore code da programmi erranti.

Utilizzare CNSBND in situazioni in cui l'applicazione potrebbe non essere stata completamente testata o potrebbe non essere affidabile o non affidabile. CNSBND è il valore predefinito.

CNSBND è definito per aiutare la documentazione del programma. Non utilizzare questa opzione con altre opzioni che controllano il tipo di collegamento utilizzato; ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Questa opzione è supportata in tutti gli ambienti.

CNFBND
Collegamento percorso rapido.

L'opzione di collegamento del percorso rapido fa sì che l'applicazione e l'agent del gestore code locale facciano parte della stessa unità di esecuzione. Il percorso rapido è in contrasto con il bind standard, in cui l'applicazione e l'agent del gestore code locale vengono eseguiti in unità di esecuzione separate.

CNFBND viene ignorato se il gestore code non supporta questo tipo di bind; l'elaborazione continua come se l'opzione non fosse stata specificata.

CNFBND può essere vantaggioso in situazioni in cui più processi consumano più risorse rispetto alla risorsa generale utilizzata dall'applicazione.Un'applicazione che utilizza il bind del percorso rapido è nota come *applicazione sicura*.

Considerare i seguenti punti importanti quando si decide se utilizzare il collegamento del percorso rapido:

- **L'utilizzo dell'opzione CNFBND non impedisce a un'applicazione di modificare o danneggiare i messaggi e altre aree di dati appartenenti al gestore code. Utilizzare questa opzione solo in situazioni in cui sono stati completamente valutati questi problemi.**
- L'applicazione non deve utilizzare segnali asincroni o interruzioni del timer (come `sigkill`) con CNFBND. Esistono anche delle limitazioni sull'utilizzo dei segmenti di memoria condivisa.
- L'applicazione non deve avere più di un thread connesso al gestore code alla volta.
- L'applicazione deve utilizzare la chiamata MQDISC per disconnettersi dal gestore code.
- L'applicazione deve terminare prima di terminare il gestore code con il comando `endmqm` .

I seguenti punti si applicano all'utilizzo di CNFBND negli ambienti indicati:

- Su IBM i, il lavoro deve essere eseguito nel profilo utente QMQM che appartiene al gruppo QMQMADM . Inoltre, il programma non deve terminare in modo anomalo, altrimenti potrebbero verificarsi risultati imprevedibili.

Per ulteriori informazioni relative alle implicazioni dell'utilizzo di applicazioni attendibili, consultare Connessione a un gestore code utilizzando la chiamata MQCONNX e Limitazioni per le applicazioni attendibili.

CNSHBD
Binding condivisi.

L'opzione dei bind condivisi fa sì che l'applicazione e l'agente del gestore code locale vengano eseguiti in unità di esecuzione separate, in genere in processi separati. La disposizione mantiene l'integrità del gestore code, ossia protegge il gestore code da programmi erranti. Tuttavia, alcune risorse sono condivise tra l'applicazione e l'agent del gestore code locale. CNSHBD viene ignorato se il gestore code non supporta questo tipo di bind. L'elaborazione continua come se l'opzione non fosse stata specificata.

CNIBND
Binding isolati.

L'opzione dei collegamenti isolati fa sì che l'applicazione e l'agente del gestore code locale vengano eseguiti in unità di esecuzione separate, generalmente in processi separati. La disposizione mantiene l'integrità del gestore code, ossia protegge il gestore code da programmi erranti. Il processo dell'applicazione e l'agente del gestore code locale sono isolati l'un l'altra in quanto non condividono risorse. CNIBND viene ignorato se il gestore code non supporta questo tipo di bind. L'elaborazione continua come se l'opzione non fosse stata specificata.

Gestione delle opzioni di condivisione

Le seguenti opzioni controllano la condivisione di handle tra thread differenti (unità di elaborazione parallela) all'interno dello stesso processo. È possibile specificare solo una di queste opzioni.

CNHSN
Nessuna condivisione handle tra thread.

L'opzione di condivisione senza handle tra thread indica che gli handle di connessione e di oggetto possono essere utilizzati solo dal thread che ha causato l'allocazione dell'handle, ovvero il thread che ha emesso la chiamata MQCONN, MQCONNXo MQOPEN . Le maniglie non possono essere utilizzate da altri thread appartenenti allo stesso processo.

CNHSB
Gestione seriale di condivisione tra thread, con blocco delle chiamate.

La condivisione dell'handle seriale tra i thread, con il blocco delle chiamate, indica che gli handle di connessione e oggetto assegnati da un thread di un processo possono essere utilizzati da altri thread appartenenti allo stesso processo. Tuttavia, solo un sottoprocesso alla volta può utilizzare un determinato handle, ovvero è consentito solo l'utilizzo seriale di un handle. Se un sottoprocesso tenta di utilizzare una gestione che è già utilizzata da un altro sottoprocesso, la chiamata blocca (attende) fino a quando la gestione non diventa disponibile.

CNHSNB
Gestione seriale della condivisione tra thread, senza blocco delle chiamate.

L'opzione di condivisione dell'handle seriale tra i thread, senza blocco delle chiamate, è la stessa dell'opzione " *con l'opzione* blocking ", ad eccezione del fatto che, se l'handle è utilizzato da un altro thread, la chiamata viene completata immediatamente con CCFAIL e RC2219 invece di bloccare fino a quando l'handle non diventa disponibile.

Un thread può avere zero o un handle non condiviso, più zero o più handle condivisi:

- Ogni chiamata MQCONN o MQCONNX che specifica CNHSN restituisce un nuovo handle non condiviso sulla prima chiamata e lo stesso handle non condiviso sulle chiamate successive (supponendo che non intervenga alcuna chiamata MQDISC). Il codice di errore è RC2002 per le seconde e successive chiamate.
- Ogni chiamata MQCONNX che specifica CNHSB o CNHSNB restituisce un nuovo handle condiviso su ogni chiamata.

I gestori oggetti ereditano le stesse proprietà di condivisione del gestore connessioni specificato nella chiamata MQOPEN che ha creato il gestore oggetti. Inoltre, le unità di lavoro ereditano le stesse proprietà di condivisione dell'handle di connessione utilizzato per avviare l'unità di lavoro; se l'unità di lavoro viene avviata in un thread utilizzando un handle condiviso, l'unità di lavoro può essere aggiornata in un altro thread utilizzando lo stesso handle.

Se non si specifica un'opzione di condivisione della gestione, l'impostazione predefinita è determinata dall'ambiente:

- Nell'ambiente Microsoft Transaction Server (MTS), il valore predefinito è lo stesso di CNHSB.
- In altri ambienti, il valore predefinito è uguale a CNHSN.

Opzioni di riconnessione

Le opzioni di riconnessione determinano se una connessione è ricollegabile. Solo le connessioni client sono ricollegabili.

CNRCDF
L'opzione di riconnessione viene risolta nel valore predefinito. Se non è impostato alcun valore predefinito, il valore di questa opzione si risolve in DISABLED. Il valore dell'opzione viene passato al server e può essere interrogato da **PCF** e **MQSC**.

CNRC
L'applicazione può essere riconnessa a qualsiasi gestore code congruente con il valore del parametro MQCONNX **QMNAME** . Utilizzare l'opzione CNRC solo se non vi è alcuna affinità tra l'applicazione client e il gestore code con cui ha inizialmente stabilito una connessione. Il valore dell'opzione viene passato al server e può essere interrogato da **PCF** e **MQSC**.

CNRCD
Impossibile ricollegare l'applicazione. Il valore dell'opzione non viene passato al server.

CNRCQM
L'applicazione può essere riconnessa solo al gestore code con cui si è originariamente connessa. Utilizzare questo valore se un client può essere riconnesso, ma c'è un'affinità tra l'applicazione client e il gestore code con cui ha originariamente stabilito una connessione. Selezionare questo valore se si desidera che un client si riconnetta automaticamente all'istanza in standby di un gestore code altamente disponibile. Il valore dell'opzione viene passato al server e può essere interrogato da **PCF** e **MQSC**.

Utilizzare le opzioni CNRC, CNRCDe CNRCQM solo per connessioni client. Se le opzioni vengono utilizzate per una connessione di collegamento, MQCONNX non riesce con il codice di completamento MQCC_FAILED e il codice motivo, MQRC_OPTIONS_ERROR.

Opzione predefinita: se nessuna delle opzioni descritte è richiesta, è possibile utilizzare la seguente opzione:

CNNONE
Non è stata specificata alcuna opzione.

CNNONE viene definito per aiutare la documentazione del programma. Non è previsto che questa opzione venga utilizzata con qualsiasi altra opzione CN* , ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

CNSCO (numero intero con segno a 10 cifre)

Questo è lo scostamento in byte di una struttura MQSCO dall'inizio della struttura MQCNO.

Questo campo viene ignorato se CNVER è inferiore a CNVER4.

CNSCP (puntatore)

Questo è l'indirizzo di una struttura MQSCO.

Questo campo viene ignorato se CNVER è inferiore a CNVER4.

CNSECPO (numero intero con segno a 10 cifre)

Offset dei parametri di sicurezza. L'offset della struttura MQCSP utilizzata per specificare un ID utente e una password.

Il valore può essere positivo o negativo. Il valore iniziale di questo campo è 0.

Questo campo viene ignorato se CNVER è inferiore a CNVER5.

CNSECPP (puntatore)

Puntatore parametri di sicurezza. Indirizzo della struttura MQCSP utilizzato per specificare un ID utente e una password.

Il valore iniziale di questo campo è un puntatore null o byte null.

Questo campo viene ignorato se CNVER è inferiore a CNVER5.

CNSID (stringa di caratteri a 4 byte)

L'identificativo della struttura per la struttura MQCNO.

Il valore deve essere:

CNSIDV
Identificativo per la struttura delle opzioni di collegamento.

Questo è sempre un campo di input. Il valore iniziale di questo campo è CNSIDV.

CNVER (numero intero con segno a 10 cifre)

Il numero di versione della struttura per la struttura MQCNO.

Il valore deve essere:

CNVER6
Version-6 struttura connect - options.

Questa versione è supportata in tutti gli ambienti.

CNVER7
Version-7 struttura delle opzioni di connessione.

Questa versione è supportata in tutti gli ambienti.

La seguente costante specifica il numero di versione della versione corrente:

CNVERC
Versione corrente della struttura delle opzioni di connessione.

Questo è sempre un campo di input. Il valore iniziale di questo campo è CNVER7.

Valori iniziali

Tabella 693. Campi in MQCNO

Nome campo	Nome della costante	Valore della costante
CNSID	CNSIDV	'CNO¬
CNVER	CNVER5	1
CNOPT	CNNONE	0
CNCCO	Nessuna	0
CNCCP	Nessuna	Puntatore null o byte null
CNCT	CTNONE	Valori null
CNSCP	Nessuna	Puntatore null o byte null

Tabella 693. Campi in MQCNO (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
CNSCO	Nessuna	0
CNCONID	Nessuna	Valori null
CNSECPO	Nessuna	0
CNSECPP	Nessuna	Puntatore null o byte null
CCDTUL	Nessuna	0
CCDTUO	Nessuna	0
CCDTUP	Nessuna	Puntatore null o byte null

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*****************************************************************
     D**                                                            **
     D**                   IBM MQ for IBM i                         **
     D**                                                            **
     D**  FILE NAME:      CMQCNOG                                   **
     D**                                                            **
     D**  DESCRIPTION:    MQCNO Structure -- Connect Options        **
     D**                                                            **
     D****************************************************************
     D**  <N_OCO_COPYRIGHT>                                         **
     D**  Licensed Materials - Property of IBM                      **
     D**                                                            **
     D**  5724-H72                                                  **
     D**  (c) Copyright IBM Corp. 1993, 2024. All Rights Reserved.  **
     D**                                                            **
     D**  US Government Users Restricted Rights - Use, duplication or **
     D**  disclosure restricted by GSA ADP Schedule Contract with   **
     D**  IBM Corp.                                                 **
     D**  <NOC_COPYRIGHT>                                           **
     D****************************************************************
     D**                                                            **
     D**  FUNCTION:       This file declares the structure MQCNO,   **
     D**                  which is used by the main MQI.            **
     D**                                                            **
     D**  PROCESSOR:      RPG (ILE)                                 **
     D**                                                            **
     D****************************************************************
     D*
     D*
     D****************************************************************
     D** <BEGIN_BUILDINFO>                                          **
     D** Generated on:  08/02/16 13:50                             **
     D** Build Level:   L000000                                    **
     D** Build Type:    Production                                 **
     D** Pointer Size:  128 Bit                                    **
     D** Source File:                                              **
     D** CMQCNOG                                                   **
     D** <END_BUILDINFO>                                           **
     D****************************************************************
     D*
     D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     D*
     D*
     D* MQCNO Structure
     D*
     D* Structure identifier
     D  CNSID                  1      4    INZ('CNO ')
     D* Structure version number
     D  CNVER                  5      8I 0 INZ(1)
     D* Options that control the action of MQCONNX
     D  CNOPT                  9     12I 0 INZ(0)
     D* Ver:1 **
```

```
     D* Offset of MQCD structure for client connection
     D  CNCCO                 13     16I 0 INZ(0)
     D* Address of MQCD structure for client connection
     D  CNCCP                 17     32*   INZ(*NULL)
     D* Ver:2 **
     D* Queue managerconnection tag
     D  CNCT                  33    160    INZ(X'00000000000000000-
     D                                     00000000000000000000000000-
     D                                     00000000000000000000000000-
     D                                     00000000000000000000000000-
     D                                     00000000000000000000000000-
     D                                     00000000000000000000000000-
     D                                     00000000000000000000000000-
     D                                     00000000000000000000000000-
     D                                     00000000000000000000000000-
     D                                     00000000000000000000000000-
     D                                     00000000000000')
     D* Ver:3 **
     D* Address of MQSCO structure for client connection
     D  CNSCP                 161    176*   INZ(*NULL)
     D* Offset of MQSCO structure for client connection
     D  CNSCO                 177    180I 0 INZ(0)
     D* Ver:4 **
     D* Unique Connection Identifier
     D  CNCONID               181    204    INZ(X'00000000000000000-
     D                                     00000000000000000000000000-
     D                                     000000')
     D* Offset of MQCSP structure
     D  CNSECPO               205    208I 0 INZ(0)
     D* Address of MQCSP structure
     D  CNSECPP               209    224*   INZ(*NULL)
     D* Ver:5 **
     D* Address of CCDT URL string
     D  CNCCDTUP              225    240*   INZ(*NULL)
     D* Offset of CCDT URL string
     D  CNCCDTUO              241    244I 0 INZ(0)
     D* Length of CCDT URL
     D  CNCCDTUL              245    248I 0 INZ(0)
     D* Ver:6 **
     D*
     D*****************************************************************
     D**  End of CMQCNOG                                             **
     D*****************************************************************
```

IBM i MQCSP (Parametri di protezione) su IBM i

Riepilogo della struttura MQCSP per IBM i.

Panoramica

Scopo: La struttura MQCSP consente al servizio di autorizzazione di autenticare un ID utente e una password. Specificare la struttura dei parametri di sicurezza della connessione MQCSP su una chiamata MQCONNX.

Serie di caratteri e codifica: i dati in MQCSP devono essere nel set di caratteri fornito dall'attributo del gestore code **CodedCharSetId** e nel gestore code locale fornito da ENNAT.

- “Campi” a pagina 1083
- “Valori iniziali” a pagina 1085
- “Dichiarazione RPG” a pagina 1086

Campi

La struttura MQCSP contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

CSAUTHT (numero intero con segno a 10 cifre)

Questo è il tipo di autenticazione da eseguire.

I valori validi sono:

CSAN
Non utilizzare i campi ID utente e password.

CSAUIAP
Autenticare i campi ID utente e password.

Questo è un campo di immissione. Il valore iniziale di questo campo è CSAN.

CSCPPL (numero intero con segno a 10 cifre)

Questa è la lunghezza della password da utilizzare nell'autenticazione.

La lunghezza massima della password non dipende dalla piattaforma. Se la lunghezza della password è superiore a quella consentita, la richiesta di autenticazione ha esito negativo con un RC2035.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

CSCPPO (numero intero con segno a 10 cifre)

Questo è l'offset in byte della password da utilizzare nell'autenticazione.

L'offset può essere positivo o negativo.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

CSCPPP (puntatore)

Questo è l'indirizzo della password da utilizzare nell'autenticazione.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null.

CSCSPUIL (numero intero con segno a 10 cifre)

La lunghezza dell'ID utente da utilizzare nell'autenticazione.

La lunghezza massima dell'ID utente non dipende dalla piattaforma. Se la lunghezza dell'ID utente è maggiore di quella consentita, la richiesta di autenticazione ha esito negativo con un RC2035.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

CSCSPUIO (numero intero con segno a 10 cifre)

Questo è l'offset in byte dell'ID utente da utilizzare nell'autenticazione.

L'offset può essere positivo o negativo.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

CSCSPUIP (puntatore)

Questo è l'indirizzo dell'ID utente da utilizzare nell'autenticazione.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null. Questo campo viene ignorato se CSVER è inferiore a CSVER5.

CSINITKL (numero intero con segno a 10 cifre)

Questa è la lunghezza della chiave iniziale per il sistema di protezione password.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

CSINITKO (numero intero con segno a 10 cifre)

Questo è lo scostamento in byte della chiave iniziale per il sistema di protezione password. L'offset può essere positivo o negativo.

È possibile utilizzare *CSINITKO* o *CSINITKP* per specificare la chiave iniziale, ma non entrambe. Per ulteriori informazioni, consultare la descrizione del campo *CSINITKP* .

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

CSINITKP (puntatore)

Questo è l'indirizzo, in byte, della chiave iniziale per il sistema di protezione password.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null.

IBM MQ MQI clients può fornire il valore di alcuni campi come valori che sono stati codificati utilizzando il sistema di protezione con password IBM MQ . I seguenti campi possono contenere valori codificati:

- La password del repository delle chiavi, nella struttura MQSCO.

Una chiave iniziale viene utilizzata dall'algoritmo di codifica per codificare e decodificare questi valori. Se viene fornita una chiave iniziale quando i valori di questi campi vengono codificati utilizzando il programma di utilità **runmqicred** , la stessa chiave iniziale deve essere specificata dal client quando si connette al gestore code.

La chiave iniziale specificata utilizzando questo campo sovrascrive qualsiasi chiave iniziale specificata utilizzando la variabile di ambiente *MQS_MQI_KEYFILE* o la proprietà *MQIInitialKeyFile* nella stanza di sicurezza del file di configurazione del client.

È possibile utilizzare *CSINITKO* o *CSINITKP* per specificare la chiave iniziale, ma non entrambe.

CSRE1 (stringa di caratteri a 4 byte)

Un campo riservato, richiesto per l'allineamento del puntatore su IBM i.

Questo è un campo di immissione. Il valore iniziale di questo campo è null.

CSRS2 (stringa di caratteri a 8 byte)

Un campo riservato, richiesto per l'allineamento del puntatore su IBM i.

Questo è un campo di immissione. Il valore iniziale di questo campo è null.

CSSID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

CSSIDV
Identificativo per la struttura dei parametri di sicurezza.

CSVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere:

CSVER1
Struttura dei parametri di sicurezza Version-1 .

CSVER2
Struttura dei parametri di sicurezza Version-2 .

La seguente costante specifica il numero di versione della versione corrente:

CSVERC
Versione corrente della struttura dei parametri di sicurezza.

Questo è sempre un campo di input. Il valore iniziale di questo campo è CSVER1.

Valori iniziali

Tabella 694. Campi in MQCNO

Nome campo	Nome della costante	Valore della costante
CSSID	CSSIDV	'CSP¬'
CSVER	CSVER1	1
CSAUTHT	Nessuna	0

Tabella 694. Campi in MQCNO (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
CSRE1	Nessuna	Valori null
CSCSPUIP	Nessuna	Puntatore null
CSCSPUIO	Nessuna	0
CSCSPUIL	Nessuna	0
CSRS2	Nessuna	Valori null
CSCPPP	Nessuna	Puntatore null
CSCPPO	Nessuna	0
CSCPPL	Nessuna	0
CSINITKP	Nessuna	Puntatore null
CSINITKO	Nessuna	0
CSINITKL	Nessuna	0

Nota:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
      D*
      D* MQCSP Structure
      D*
      D* Structure identifier
      D  CSSID                 1      4    INZ('CSP ')
      D* Structure version number
      D  CSVER                 5      8I 0 INZ(1)
      D* Type of authentication
      D  CSAUTHT               9     12I 0 INZ(0)
      D* Reserved
      D  CSRE1                13     16    INZ(X'00000000')
      D* Address of user ID
      D  CSCSPUIP             17     32*   INZ(*NULL)
      D* Offset of user ID
      D  CSCSPUIO             33     36I 0 INZ(0)
      D* Length of user ID
      D  CSCSPUIL             37     40I 0 INZ(0)
      D* Reserved
      D  CSRS2                41     48    INZ(X'0000000000000000')
      D* Address of password
      D  CSCPPP               49     64*   INZ(*NULL)
      D* Offset of password
      D  CSCPPO               65     68I 0 INZ(0)
      D* Length of password
      D  CSCPPL               69     72I 0 INZ(0)
```

IBM i MQCTLO (Control callback options structure) su IBM i

Struttura che specifica la funzione di callback di controllo.

Panoramica

Finalità

La struttura MQCTLO viene utilizzata per specificare le opzioni relative a una funzione di callback di controllo.

La struttura è un parametro di input e output nella chiamata MQCTL .

Versione
La versione corrente di MQCTLO è CTLV1.

Serie di caratteri e codifica
I dati in MQCTLO devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da ENNAT. Tuttavia, se l'applicazione è in esecuzione come un client IBM MQ , la struttura deve essere nella serie di caratteri e nella codifica del client.

- "Campi" a pagina 1087
- "Valori iniziali" a pagina 1088
- "Dichiarazione RPG" a pagina 1088

Campi

La struttura MQCTLO contiene i seguenti campi; i campi sono descritti in ordine alfabetico:

COCONNAREA (numero intero con segno a 10 cifre)

Struttura opzioni di controllo - Campo ConnectionArea .

Questo è un campo disponibile per la funzione di callback da utilizzare.

Il gestore code non prende alcuna decisione in base al contenuto di questo campo e viene inoltrato non modificato dal campo CBCCONNAREA nella struttura MQCBC, che è un parametro sulla chiamata MQCB.

Questo campo viene ignorato per tutte le operazioni diverse da CTLSR e CTLSW.

Questo è un campo di input e output per la funzione di callback. Il valore iniziale di questo campo è un puntatore null o byte null.

COOPT (numero intero con segno a 10 cifre)

Opzioni che controllano l'azione di MQCTLO.

QCTL
Forzare l'esito negativo della chiamata MQCTLO se il gestore code o la connessione sono in stato di inattività.

Specificare GMFIQ, nelle opzioni MQGMO inoltrate alla chiamata MQCB, per causare la notifica ai consumatori di messaggi quando sono in fase di sospensione.

CTLTHR
Questa opzione informa il sistema che l'applicazione richiede che tutti i destinatari del messaggio, per lo stesso collegamento, vengano richiamati sullo stesso thread.

Opzione predefinita: se non è necessaria alcuna delle opzioni descritte, utilizzare la seguente opzione:

N. CTL
Utilizzare questo valore per indicare che non sono state specificate altre opzioni; tutte le opzioni assumono i propri valori predefiniti. CTLNO è definito per la documentazione del programma di aiuto; non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Questo è un campo di immissione. Il valore iniziale del campo *COOPT* è CTLNO.

CORSV (numero intero con segno a 10 cifre)

Questo è un campo riservato. Il valore iniziale di questo campo è un carattere vuoto.

COSID (numero intero con segno a 10 cifre)

Struttura opzioni di controllo - campo StrucId .

Questo è l'identificatore della struttura; il valore deve essere:

CTLSI
Identificativo per la struttura Opzioni di controllo.

Questo è sempre un campo di input. Il valore iniziale di questo campo è CTLSI.

COVER (numero intero con segno a 10 cifre)

Struttura delle opzioni di controllo - campo Versione.

Questo è il numero di versione della struttura; il valore deve essere:

CTLV1
Version-1 Struttura delle opzioni di controllo.

La seguente costante specifica il numero di versione della versione corrente:

CCTLCV
La versione corrente della struttura delle opzioni di controllo.

Questo è sempre un campo di input. Il valore iniziale di questo campo è CTLV1.

Valori iniziali

Tabella 695. Campi in MQCTLO		
Nome campo	**Nome della costante**	**Valore della costante**
COSID	CTLSI	'CTLO'
COVER	CTLV1	1
COOPT	N. CTL	Valori null
CORSV	Campo riservato	
COCONNAREA	Nessuna	Puntatore null o byte null

Dichiarazione RPG

```
D* MQCTLO Structure
D*
D*
D* Structure identifier
D  COSID                  1      4    INZ('CTLO')
D*
D* Structure version number
D  COVER                  5      8I 0 INZ(1)
D*
D* Options that control the action of MQCTL
D  COOPT                  9     12I 0 INZ(0)
D*
D* Reserved
D  CORSV                 13     16I 0 INZ(-1)
D*
D* MQCTL Data area passed to the function
D  COCONNAREA            17     32*   INZ(*NULL)
```

IBM i MQDH (Distribution header) su IBM i

La struttura MQDH descrive i dati aggiuntivi presenti in un messaggio quando tale messaggio è un messaggio dell'elenco di distribuzione memorizzato in una coda di trasmissione.

Panoramica

Scopo: un messaggio elenco di distribuzione è un messaggio inviato a più code di destinazione. I dati aggiuntivi sono costituiti dalla struttura MQDH seguita da un array di record MQOR e da un array di record MQPMR.

Questa struttura viene utilizzata da applicazioni specializzate che inseriscono i messaggi direttamente nelle code di trasmissione o che rimuovono i messaggi dalle code di trasmissione (ad esempio: agent del canale dei messaggi).

Questa struttura non dovrebbe essere utilizzata dalle normali applicazioni che vogliono semplicemente inserire i messaggi negli elenchi di distribuzione. Tali applicazioni devono utilizzare la struttura MQOD per definire le destinazioni nell'elenco di distribuzione e la struttura MQPMO per specificare le proprietà del messaggio o ricevere informazioni sui messaggi inviati alle singole destinazioni.

Serie di caratteri e codifica: i dati in MQDH devono essere nel set di caratteri fornito dall'attributo del gestore code di **CodedCharSetId** e nella codifica del gestore code locale fornito da ENNAT per il linguaggio di programmazione C.

La serie di caratteri e la codifica di MQDH devono essere impostate nei campi *MDCSI* e *MDENC* in:

- MQMD (se la struttura MQDH è all'inizio dei dati del messaggio) oppure
- La struttura dell'intestazione che precede la struttura MQDH (tutti gli altri casi).

Utilizzo: quando un'applicazione inserisce un messaggio in un elenco di distribuzione e alcune o tutte le destinazioni sono remote, il gestore code antepone i dati del messaggio dell'applicazione con le strutture MQXQH e MQDH e colloca il messaggio nella coda di trasmissione pertinente. I dati si verificano quindi nella seguente sequenza quando il messaggio si trova su una coda di trasmissione:

- Struttura MQXQH
- Struttura MQDH più array di record MQOR e MQPMR
- Dati messaggio applicazione

A seconda delle destinazioni, più di un messaggio di questo tipo potrebbe essere generato dal gestore code e posizionato su code di trasmissione differenti. In questo caso, le strutture MQDH in tali messaggi identificano sottoinsiemi differenti delle destinazioni definite dall'elenco di distribuzioni aperto dall'applicazione.

Un'applicazione che inserisce un messaggio dell'elenco di distribuzione direttamente in una coda di trasmissione deve essere conforme alla sequenza descritta in precedenza e deve accertarsi che la struttura MQDH sia corretta. Se la struttura MQDH non è valida, il gestore code può scegliere di non eseguire la chiamata MQPUT o MQPUT1 con codice motivo RC2135.

I messaggi possono essere memorizzati su una coda in formato elenco di distribuzione solo se la coda è definita come in grado di supportare i messaggi dell'elenco di distribuzione (consultare l'attributo della coda **DistLists** descritto in “Attributi per le code” a pagina 1406). Se un'applicazione inserisce un messaggio dell'elenco di distribuzione direttamente in una coda che non supporta gli elenchi di distribuzione, il gestore code suddivide il messaggio dell'elenco di distribuzione in singoli messaggi e inserisce invece quelli nella coda.

- “Campi” a pagina 1089
- “Valori iniziali” a pagina 1092
- “Dichiarazione RPG” a pagina 1093

Campi

La struttura MQDH contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

DHCNT (numero intero con segno a 10 cifre)

Numero di record MQOR presenti.

Definisce il numero di destinazioni. Un elenco di distribuzione deve contenere sempre almeno una destinazione, quindi *DHCNT* deve essere sempre maggiore di zero.

Il valore iniziale di questo campo è 0.

DHCSI (numero intero con segno a 10 cifre)

Identificativo della serie di caratteri dei dati che seguono i record MQOR e MQPMR.

Specifica l'identificativo della serie di caratteri dei dati che seguono gli array dei record MQOR e MQPMR; non si applica ai dati di carattere nella stessa struttura MQDH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. È possibile utilizzare il seguente valore speciale:

CINHT
Eredita l'identificativo della serie di caratteri di questa struttura.

I dati carattere nei dati *che seguono* questa struttura si trovano nella stessa serie di caratteri di questa struttura.

Il gestore code modifica questo valore nella struttura inviata nel messaggio nell'effettivo identificativo della serie di caratteri della struttura. Se non si verifica alcun errore, il valore CSINHT non viene restituito dalla chiamata MQGET.

CSINHT non può essere utilizzato se il valore del campo *MDPAT* in MQMD è ATBRKR.

Il valore iniziale di questo campo è CSUNDF.

DHENC (numero intero con segno a 10 cifre)

Codifica numerica dei dati che seguono i record MQOR e MQPMR.

Specifica la codificazione numerica dei dati che seguono gli array di record MQOR e MQPMR; non si applica ai dati numerici nella stessa struttura MQDH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati.

Il valore iniziale di questo campo è 0.

DHFLG (numero intero con segno a 10 cifre)

Indicatori generali.

È possibile specificare il seguente indicatore:

NUOVO
Generare nuovi identificativi di messaggio.

Questo indicatore indica che è necessario creare un nuovo identificativo messaggio per ogni destinazione nell'elenco di distribuzione. Questa opzione può essere impostata solo quando non sono presenti record del messaggio di inserimento o quando i record sono presenti ma non contengono il campo *PRMID* .

L'utilizzo di questo indicatore differisce la creazione degli identificatori del messaggio fino all'ultimo momento possibile, ossia il momento in cui il messaggio dell'elenco di distribuzione viene finalmente suddiviso in singoli messaggi. Ciò riduce al minimo la quantità di informazioni di controllo che devono essere trasmesse con il messaggio dell'elenco di distribuzione.

Quando un'applicazione inserisce un messaggio in un elenco di distribuzione, il gestore code imposta DHFNEW nell'MQDH che genera quando entrambe le seguenti istruzioni sono vere:

- Non ci sono record put - message forniti dall'applicazione o i record forniti non contengono il campo *PRMID* .
- Il campo *MDMID* in MQMD è MINONE oppure il campo *PMOPT* in MQPMO include PMNMID

Se non sono necessari indicatori, è possibile specificare quanto segue:

NON DHF
Nessun indicatore.

Questa costante indica che non è stato specificato alcun indicatore. DHFNON è definito per aiutare la documentazione del programma. Non è previsto che questa costante venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Il valore iniziale di questo campo è DHFNON.

DHFMT (stringa di caratteri a 8 byte)

Nome formato dei dati che seguono i record MQOR e MQPMR.

Specifica il nome del formato dei dati che seguono gli array dei record MQOD e MQPMR (a seconda di quale si verifica per ultimo).

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono le stesse del campo *MDFMT* in MQMD.

Il valore iniziale di questo campo è FMNONE.

DHLEN (numero intero con segno a 10 cifre)

Lunghezza della struttura MQDH più i seguenti record MQOR e MQPMR.

Questo è il numero di byte dall'inizio della struttura MQDH all'inizio dei dati dei messaggi che seguono gli array di record MQOR e MQPMR. I dati si trovano nella seguente sequenza:

- struttura MQDH
- Array di record di MQOR
- Array di record MQPMR
- Dati messaggio

Gli array dei record MQOR e MQPMR sono indirizzati dagli offset contenuti nella struttura MQDH. Se questi offset risultano in byte inutilizzati tra uno o più della struttura MQDH, gli array di record e i dati del messaggio, tali byte inutilizzati devono essere inclusi nel valore di *DHLEN*, ma il contenuto di tali byte non viene conservato dal gestore code. È valido che l'array di record MQPMR preceda l'array di record MQOR.

Il valore iniziale di questo campo è 0.

DHORO (numero intero con segno a 10 cifre)

Offset del primo record MQOR dall'inizio di MQDH.

Questo campo fornisce lo scostamento in byte del primo record nell'array di record oggetto MQOR che contengono i nomi delle code di destinazione. Ci sono *DHCNT* record in questo array. Questi record (più eventuali byte ignorati tra il primo record oggetto e il campo precedente) sono inclusi nella lunghezza fornita dal campo *DHLEN* .

Un elenco di distribuzione deve contenere sempre almeno una destinazione, quindi *DHORO* deve essere sempre maggiore di zero.

Il valore iniziale di questo campo è 0.

DHPRF (numero intero con segno a 10 cifre)

Indicatori che indicano quali campi MQPMR sono presenti.

È possibile specificare zero o più dei seguenti indicatori:

IDPFM
Il campo identificativo messaggio è presente.

IDPFC
Il campo Identificativo correlazione è presente.

IDGFP
Il campo identificativo gruppo è presente.

FPFB
Il campo Feedback è presente.

FPACC
Il campo Accounting - token è presente.

Se non è presente alcun campo MQPMR, è possibile specificare quanto segue:

FPNONE
Non sono presenti campi record di messaggi di immissione.

PFNONE è definito per aiutare la documentazione del programma. Non è previsto che questa costante venga utilizzata con altre, ma poiché il valore è zero, tale utilizzo non può essere rilevato.

Il valore iniziale di questo campo è PFNONE.

DHPRO (numero intero con segno a 10 cifre)

Offset del primo record MQPMR dall'inizio di MQDH.

Questo campo fornisce l'offset in byte del primo record nell'array di record di messaggi di inserimento MQPMR che contengono le proprietà del messaggio. Se presente, ci sono *DHCNT* record in questo array. Questi record (più eventuali byte saltati tra il record del primo messaggio di inserimento e il campo precedente) sono inclusi nella lunghezza fornita dal campo *DHLEN* .

I record del messaggio di inserimento sono facoltativi; se non viene fornito alcun record, *DHPRO* è zero e *DHPRF* ha il valore PFNONE.

Il valore iniziale di questo campo è 0.

DHSID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

DSSID
Identificativo per la struttura dell'intestazione di distribuzione.

Il valore iniziale di questo campo è DHSIDV.

DHVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere:

DHVER1
Numero di versione per la struttura dell'intestazione di distribuzione.

La seguente costante specifica il numero di versione della versione corrente:

DDHVERC
Versione corrente della struttura dell'intestazione di distribuzione.

Il valore iniziale di questo campo è DHVER1.

Valori iniziali

Tabella 696. Campi in MQDH

Nome campo	Nome della costante	Valore della costante
DHSID	DSSID	'DH¬¬'
DHVER	DHVER1	1
DHLEN	Nessuna	0

Tabella 696. Campi in MQDH (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
DHENC	Nessuna	0
DHCSI	SUNDF	0
DHFMT	FMNONE	Spazi
DHFLG	NON DHF	0
DHPRF	FPNONE	0
DHCNT	Nessuna	0
DHORO	Nessuna	0
DHPRO	Nessuna	0

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQDH Structure
D*
D* Structure identifier
D  DHSID                  1      4    INZ('DH  ')
D* Structure version number
D  DHVER                  5      8I 0 INZ(1)
D* Length of MQDH structure plusfollowing MQOR and MQPMR records
D  DHLEN                  9     12I 0 INZ(0)
D* Numeric encoding of data that followsthe MQOR and MQPMR records
D  DHENC                 13     16I 0 INZ(0)
D* Character set identifier of data thatfollows the MQOR and MQPMR
D* records
D  DHCSI                 17     20I 0 INZ(0)
D* Format name of data that follows theMQOR and MQPMR records
D  DHFMT                 21     28    INZ('        ')
D* General flags
D  DHFLG                 29     32I 0 INZ(0)
D* Flags indicating which MQPMR fieldsare present
D  DHPRF                 33     36I 0 INZ(0)
D* Number of MQOR records present
D  DHCNT                 37     40I 0 INZ(0)
D* Offset of first MQOR record from startof MQDH
D  DHORO                 41     44I 0 INZ(0)
D* Offset of first MQPMR record fromstart of MQDH
D  DHPRO                 45     48I 0 INZ(0)
```

IBM i MQDLH (Intestazione non instradabile) su IBM i

Panoramica

Finalità

La struttura MQDLH descrive le informazioni che precedano i dati del messaggio dell'applicazione dei messaggi sulla coda di messaggi non recapitabili (messaggi non recapitabili). Un messaggio può arrivare sulla coda dei messaggi non instradati perché il gestore code o l'agent del canale dei messaggi lo ha reindirizzato alla coda. Un'applicazione potrebbe inserire il messaggio direttamente sulla coda.

Nome formato

FMDLH

Serie di caratteri e codifica

MQDLH potrebbe essere all'inizio dei dati del messaggio dell'applicazione. In tal caso, i campi nella struttura MQDLH si trovano nella serie di caratteri e nella codifica fornita dai campi MDCSI e MDENC . In caso contrario, la serie di caratteri e la codifica vengono impostate dai campi MDCSI e MDENC nella struttura dell'intestazione che precede MQDLH.

La serie di caratteri deve avere caratteri a byte singolo per i caratteri validi nei nomi di coda.

Utilizzo

Le applicazioni che inserano i messaggi direttamente nella coda di messaggi non recapitabili devono aggiungere ai dati del messaggio una struttura MQDLH e inizializzare i campi con valori appropriati. Tuttavia, il gestore code non richiede la presenza di una struttura MQDLH o la specifica di valori validi per i campi.

Se un messaggio è troppo lungo per essere inserito nella coda di messaggi non recapitabili, l'applicazione deve considerare di effettuare una delle seguenti operazioni:

- Troncare i dati del messaggio per adattarli alla coda di messaggi non recapitabili.
- Registrare il messaggio nella memoria ausiliaria e inserire un messaggio di report di eccezione nella coda dei messaggi non instradabili che indica che il messaggio è troppo lungo.
- Eliminare il messaggio e restituire un errore al relativo creatore. Se il messaggio è critico. Eliminare il messaggio solo se è noto che il mittente ha ancora una copia del messaggio. Ad esempio, un messaggio ricevuto da un MCA (message channel agent) da un canale di comunicazione.

La scelta appropriata dipende dalla progettazione dell'applicazione.

Il gestore code esegue un'elaborazione speciale quando un messaggio che è un segmento viene inserito con una struttura MQDLH nella parte anteriore. Per ulteriori informazioni, consultare la descrizione della struttura MQMDE .

- "Inserimento di messaggi nella coda di messaggi non recapitabili" a pagina 1094
- "Richiamo dei messaggi dalla coda di messaggi non recapitabili" a pagina 1095
- "Campi" a pagina 1095
- "Valori iniziali" a pagina 1099
- "Dichiarazione RPG" a pagina 1099

Inserimento di messaggi nella coda di messaggi non recapitabili

Se un messaggio viene inserito nella coda di messaggi non instradati, la struttura MQMD utilizzata per la chiamata MQPUT o MQPUT1 deve essere identica al MQMD associato al messaggio. Il MQMD è in genere quello restituito dalla chiamata MQGET , ad eccezione dei seguenti casi:

- I campi MDCSI e MDENC devono essere impostati su qualsiasi serie di caratteri e codifica utilizzati per i campi nella struttura MQDLH .
- Il campo MDFMT deve essere impostato su FMDLH per indicare che i dati iniziano con una struttura MQDLH .
- I campi di contesto, MDACC, MDAID, MDAOD, MDPAN, MDPAT, MDPD, MDPTe MDUID devono essere impostati utilizzando un'opzione di contesto appropriata alle circostanze:
  - Un'applicazione che immette nella coda di messaggi non instradabili un messaggio non correlato ad alcun messaggio precedente deve utilizzare l'opzione PMDEFC . L'opzione PMDEFC fa sì che il gestore code imposti tutti i campi di contesto nel descrittore del messaggio sui valori predefiniti.
  - Un'applicazione server che inserisce nella coda di messaggi non instradabili un messaggio ricevuto deve utilizzare l'opzione PMPASA , per preservare le informazioni di contesto originali.
  - Un'applicazione server che immette nella coda di messaggi non recapitabili una risposta al messaggio ricevuto deve utilizzare l'opzione PMPASI . L'opzione PMPASI conserva le informazioni di identit ... ma imposta le informazioni di origine in modo che siano quelle dell'applicazione server.

– Un agente del canale dei messaggi che immette nella coda di messaggi non instradabili un messaggio ricevuto dal relativo canale di comunicazioni deve utilizzare l'opzione PMSETA . L'opzione PMSETA conserva le informazioni di contesto originali.

Nella struttura MQDLH , i campi devono essere impostati come segue:

- I campi DLCSI, DLENCe *DLFMT* devono essere impostati sui valori che descrivono i dati che seguono la struttura MQDLH . Questi valori sono generalmente i valori del descrittore del messaggio originale.
- I campi di contesto DLPAT, DLPAN, DLPDe DLPT devono essere impostati sui valori appropriati per l'applicazione che sta inserendo il messaggio nella coda di messaggi non instradabili. Questi valori non sono correlati al messaggio originale.
- Gli altri campi devono essere impostati come appropriato.

L'applicazione deve garantire che tutti i campi abbiano valori validi e che i campi carattere siano riempiti con spazi vuoti fino alla lunghezza definita del campo. I dati carattere non devono essere terminati prematuramente utilizzando un carattere null. Il gestore code non converte i caratteri null e successivi in spazi vuoti nella struttura MQDLH .

Richiamo dei messaggi dalla coda di messaggi non recapitabili

Le applicazioni che ricevono messaggi dalla coda di messaggi non recapitabili devono verificare che i messaggi inizino con una struttura MQDLH . L'applicazione può determinare se una struttura MQDLH è presente esaminando il campo MDFMT nel descrittore del messaggio MQMD. Se il campo ha il valore FMDLH, i dati del messaggio iniziano con una struttura MQDLH . I messaggi sulla coda di messaggi non recapitabili potrebbero essere troncati se in origine erano troppo lunghi per la coda a cui erano destinati.

Campi

La struttura MQDLH contiene i seguenti campi; i campi sono descritti in ordine alfabetico:

DLCSI (numero intero con segno a 10 cifre)

Identificativo della serie di caratteri dei dati che seguono MQDLH.

DLCSI specifica l'identificativo della serie di caratteri dei dati che seguono la struttura MQDLH . I dati provengono generalmente dal messaggio originale. Non si applica ai dati carattere nella struttura MQDLH stessa.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. È possibile utilizzare il seguente valore speciale:

CSINHT
Eredita l'identificativo della serie di caratteri di questa struttura.

I dati carattere nei dati che seguono questa struttura si trovano nella stessa serie di caratteri di questa struttura.

Il gestore code modifica questo valore nella struttura inviata nel messaggio nell'effettivo identificativo della serie di caratteri della struttura. Se non si verifica alcun errore, il valore CSINHT non viene restituito dalla chiamata MQGET .

CSINHT non può essere utilizzato se il valore del campo MDPAT in MQMD è ATBRKR.

Il valore iniziale di questo campo è CSUNDF.

DLDM (stringa di caratteri a 48 byte)

Il nome del gestore code della destinazione originaria.

Questo è il nome del gestore code che era la destinazione originale del messaggio.

La lunghezza di questo campo è fornita da LNQMN. Il valore iniziale di questo campo è di 48 caratteri vuoti.

DLDQ (stringa di caratteri a 48 byte)

Nome della coda di destinazione originale.

Questo è il nome della coda messaggi che era la destinazione originale del messaggio.

La lunghezza di questo campo è fornita da LNQN. Il valore iniziale di questo campo è di 48 caratteri vuoti.

DLENC (numero intero con segno a 10 cifre)

Codifica numerica dei dati che seguono MQDLH.

DLENC specifica la codifica numerica dei dati che seguono la struttura MQDLH . I dati provengono generalmente dal messaggio originale. Non si applica ai dati numerici nella struttura MQDLH stessa.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati.

Il valore iniziale di questo campo è 0.

DLFMT (stringa di caratteri a 8 byte)

Nome formato dei dati che seguono MQDLH.

Specifica il nome del formato dei dati che seguono la struttura MQDLH (generalmente i dati dal messaggio originale).

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono le stesse del campo MDFMT in MQMD.

La lunghezza di questo campo è fornita da LNFMT. Il valore iniziale di questo campo è FMNONE.

DLPAN (stringa di caratteri a 28 byte)

Nome dell'applicazione che inserisce il messaggio nella coda di messaggi non recapitabili (messaggio non recapito).

Il formato del nome dipende dal campo DLPAT . Consultare la descrizione del campo MDPAN in “MQMD (Message Descriptor) su IBM i” a pagina 1140.

Se si tratta del gestore code che reindirizza il messaggio alla coda di messaggi non instradati, DLPAN contiene i primi 28 caratteri del nome del gestore code. Il nome viene riempito con spazi vuoti, se necessario.

La lunghezza di questo campo è fornita da LNPAN. Il valore iniziale di questo campo è di 28 caratteri vuoti.

DLPAT (numero intero con segno a 10 cifre)

Tipo di applicazione che inserisce il messaggio nella coda di messaggi non recapitabili (messaggio non recapitato).

Questo campo ha lo stesso significato del campo MDPAT nel descrittore del messaggio MQMD (consultare “MQMD (Message Descriptor) su IBM i” a pagina 1140 per i dettagli).

Se si tratta del gestore code che reindirizza il messaggio alla coda di messaggi non instradati, DLPAT ha il valore ATQM.

Il valore iniziale di questo campo è 0.

DLPD (stringa di caratteri a 8 byte)

Data in cui il messaggio è stato inserito nella coda di messaggi non recapitabili (messaggi non recapitabili).

Il formato utilizzato per la data in cui questo campo viene generato dal gestore code è:

- YYYYMMDD

dove i caratteri rappresentano:

YYYY
anno (quattro cifre)

MM
mese dell'anno (da 01 a 12)

DD
giorno del mese (da 01 a 31)

GMT (Greenwich Mean Time) viene utilizzato per i campi DLPD e DLPT , a condizione che l'orologio di sistema sia impostato in modo accurato su GMT.

La lunghezza di questo campo è fornita da LNPDAT. Il valore iniziale di questo campo è di otto caratteri vuoti.

DLPT (stringa di caratteri a 8 byte)

L'ora in cui il messaggio è stato inserito nella coda dei messaggi non recapitabili (messaggi non recapitabili).

Il formato utilizzato per l'ora in cui questo campo viene generato dal gestore code è:

- HHMMSSTH

dove i caratteri rappresentano (in ordine):

HH
ore (da 00 a 23)

MM
minuti (da 00 a 59)

SS
secondi (da 00 a 59; consultare la nota più avanti in questo argomento)

T
decimi di secondo (da 0 a 9)

H
centesimi di secondo (da 0 a 9)

Nota: Se l'orologio di sistema è sincronizzato con uno standard di ora preciso, è possibile che 60 o 61 vengano restituiti per i secondi in DLPT. Il secondo supplementare si verifica quando i secondi bisestili vengono inseriti nello standard di tempo globale.

GMT (Greenwich Mean Time) viene utilizzato per i campi DLPD e DLPT , a condizione che l'orologio di sistema sia impostato in modo accurato su GMT.

La lunghezza di questo campo è fornita da LNPTIM. Il valore iniziale di questo campo è di otto caratteri vuoti.

DLREA (numero intero con segno a 10 cifre)

Il messaggio di errore è arrivato nella coda dei messaggi non recapitabili (non recapitati).

Ciò identifica il motivo per cui il messaggio è stato inserito nella coda di messaggi non instradabili invece che nella coda di destinazione originale. Deve essere uno dei valori FB* o RC* (ad esempio, RC2053). Consultare la descrizione del campo *MDFB* in "MQMD (Message Descriptor) su IBM i" a pagina 1140 per dettagli sui valori FB* comuni che possono verificarsi.

Se il valore è compreso tra FBIFST e FBILST, il codice di errore IMS effettivo può essere determinato sottraendo FBIERR dal valore del campo *DLREA* .

Alcuni valori FB* si verificano solo in questo campo. Sono relativi ai messaggi del repository, ai messaggi trigger o ai messaggi della coda di trasmissione che vengono trasferiti alla coda di messaggi non instradabili. Questi valori sono:

FBABEG
Impossibile avviare l'applicazione.

Un'applicazione che elabora un messaggio di trigger non è stata in grado di avviare l'applicazione denominata nel campo TMAI del messaggio di trigger; consultare “MQTM - Messaggio trigger” a pagina 1271.

FBATYP
Errore nel tipo di applicazione.

Un'applicazione che elabora un messaggio di trigger non è stata in grado di avviare l'applicazione perché il campo TMAT del messaggio di trigger non è valido; consultare “MQTM - Messaggio trigger” a pagina 1271.

FBBOCD
Canale ricevente del cluster eliminato.

Il messaggio si trovava su una coda di trasmissione cluster destinata a una coda cluster aperta con l'opzione FBIERR . Il canale ricevente del cluster remoto da utilizzare per trasmettere il messaggio alla coda di destinazione è stato eliminato prima che il messaggio potesse essere inviato. Poiché è stato specificato FBIERR , solo il canale selezionato quando la coda è stata aperta può essere utilizzato per trasmettere il messaggio. Poiché questo canale non è più disponibile, il messaggio è stato inserito nella coda di messaggi non recapitabili.

FBNARM
Il messaggio non è un messaggio del repository.

FBSBCX
Il messaggio è stato arrestato dall'uscita di definizione automatica del canale.

FBSBMX
Il messaggio è stato arrestato dall'uscita del messaggio di canale.

FBTM
Struttura MQTM non valida o mancante.

Il campo MDFMT in MQMD specifica FMTM, ma il messaggio non inizia con una struttura MQTM valida. Ad esempio, l'eye-catcher mnemonico *TMSID* potrebbe non essere valido. *TMVER* potrebbe non essere riconosciuto. La lunghezza del messaggio di trigger potrebbe non essere sufficiente per contenere la struttura MQTM .

FBXQME
Il formato del messaggio sulla coda di trasmissione non è corretto.

Un agente del canale dei messaggi ha rilevato che un messaggio nella coda di trasmissione non è nel formato corretto. L'agente del canale dei messaggi inserisce il messaggio nella coda di messaggi non recapitabili utilizzando questo codice di feedback.

Il valore iniziale di questo campo è RCNONE.

DLSID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

DLSIDV
Identificativo per la struttura dell'intestazione lettera non recapitabile.

Il valore iniziale di questo campo è DLSIDV.

DLVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere:

DLVER1
Numero di versione per la struttura dell'intestazione dei messaggi non recapitabili.

La seguente costante specifica il numero di versione della versione corrente:

DLVERC
La versione corrente della struttura dell'intestazione dei messaggi non instradabili.

Il valore iniziale di questo campo è DLVER1.

Valori iniziali

Tabella 697. Campi in MQDLH		
Nome campo	**Nome della costante**	**Valore della costante**
DLSID	DLSIDV	'DLH¬'
DLVER	DLVER1	1
DLREA	RCNONE	0
DLDQ	Nessuna	Spazi
DLDM	Nessuna	Spazi
DLENC	Nessuna	0
DLCSI	CSUNDF	0
DLFMT	FMNONE	Spazi
DLPAT	Nessuna	0
DLPAN	Nessuna	Spazi
DLPD	Nessuna	Spazi
DLPT	Nessuna	Spazi

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQDLH Structure
D*
D* Structure identifier
D  DLSID                  1      4    INZ('DLH ')
D* Structure version number
D  DLVER                  5      8I 0 INZ(1)
D* Reason message arrived on dead-letter(undelivered-message) queue
D  DLREA                  9     12I 0 INZ(0)
D* Name of original destination queue
D  DLDQ                  13     60    INZ
D* Name of original destination queue manager
D  DLDM                  61    108    INZ
D* Numeric encoding of data that followsMQDLH
D  DLENC                 109    112I 0 INZ(0)
D* Character set identifier of data thatfollows MQDLH
D  DLCSI                 113    116I 0 INZ(0)
D* Format name of data that followsMQDLH
D  DLFMT                 117    124    INZ('        ')
D* Type of application that put messageon dead-letter
D* (undelivered-message)queue
D  DLPAT                 125    128I 0 INZ(0)
D* Name of application that put messageon dead-letter
D* (undelivered-message)queue
D  DLPAN                 129    156    INZ
D* Date when message was put ondead-letter (undelivered-message)queue
D  DLPD                  157    164    INZ
D* Time when message was put on thedead-letter (undelivered-message)queue
D  DLPT                  165    172    INZ
```

IBM i MQDMHO (Delete message handle options) su IBM i

La struttura **MQDMHO** consente alle applicazioni di specificare le opzioni che controllano il modo in cui vengono eliminati gli handle dei messaggi.

Panoramica

Scopo: La struttura è un parametro di immissione sulla chiamata **MQDLTMH** .

Serie di caratteri e codifica: i dati in **MQDMHO** devono essere nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (ENNAT).

Campi

La struttura MQDMHO contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

DMOPT (numero intero con segno a 10 cifre)

Il valore deve essere:

DMNONE

Nessuna opzione specificata.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **DMNONE**.

DMSID (numero intero con segno a 10 cifre)

Questo è l'identificatore della struttura; il valore deve essere:

DMSIDV

Identificativo per la struttura delle opzioni di gestione del messaggio di eliminazione.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **DMSIDV**.

DMVER (numero intero con segno a 10 cifre)

Questo è il numero di versione della struttura; il valore deve essere:

DMVER1
Version-1 eliminare la struttura di opzioni dell'handle del messaggio.

La seguente costante specifica il numero di versione della versione corrente:

DMVERC
Versione corrente della struttura delle opzioni di gestione dei messaggi di eliminazione.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **DMVER1**.

Valori iniziali

Tabella 698. Campi in MQDMHO

Nome campo	Nome della costante	Valore della costante
DMSID	DMSIDV	'DMHO'
DMVER	DMVER1	1

Tabella 698. Campi in MQDMHO (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
DMOPT	DMNONE	0

Dichiarazione RPG

```
D* MQDMHO Structure
D*
D*
D* Structure identifier
D  DMSID                  1      4    INZ('DMHO')
D*
D* Structure version number
D  DMVER                  5      8I 0 INZ(1)
D*
D* Options that control the action of MQDLTMH
D  DMOPT                  9     12I 0 INZ(0)
```

IBM i MQDMPO (Opzioni di eliminazione proprietà messaggio) su IBM i

Struttura che definisce le opzioni della proprietà di eliminazione del messaggio.

Panoramica

Scopo: La struttura MQDMPO consente alle applicazioni di specificare le opzioni che controllano il modo in cui vengono eliminate le proprietà dei messaggi. La struttura è un parametro di input sulla chiamata MQDLTMP.

Serie di caratteri e codifica: i dati in MQDMPO devono trovarsi nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (ENNAT).

- "Campi" a pagina 1101
- "Valori iniziali" a pagina 1102
- "Dichiarazione RPG" a pagina 1102

Campi

La struttura MQDMPO contiene i seguenti campi; i campi sono descritti in ordine alfabetico:

DPOPT (numero intero con segno a 10 cifre)

Cancellare la struttura delle opzioni della proprietà del messaggio - campo DPOPT.

Opzioni di ubicazione: le opzioni riportate di seguito si riferiscono alla posizione relativa della proprietà rispetto al cursore della proprietà.

DPDELF
Elimina la prima proprietà che corrisponde al nome specificato.

DPDELC
Elimina la proprietà indicata dal cursore della proprietà; questa è la proprietà che è stata interrogata per ultima utilizzando l'opzione IPINQF o IPINQN.

Il cursore della proprietà viene reimpostato quando l'handle del messaggio viene riutilizzato. Viene anche reimpostato quando l'handle del messaggio viene specificato nel campo *HMSG* di MQGMO su una chiamata MQGET o la struttura MQPMO su una chiamata MQPUT.

Il cursore della proprietà viene reimpostato quando la gestione del messaggio viene riutilizzata o quando la gestione del messaggio viene specificata nel campo *HMSG* della struttura MQGMO

su una struttura MQGET su una chiamata MQGET o su una struttura MQPMO su una chiamata MQPUT.

La chiamata ha esito negativo con codice di completamento CCFAIL e motivo RC2471 se questa opzione viene utilizzata quando il cursore della proprietà non è ancora stato stabilito. Non riesce anche con questi codici se la proprietà indicata dal cursore della proprietà è già stata eliminata.

Se nessuna di queste opzioni è richiesta, è possibile utilizzare la seguente opzione:

DPNONE
Nessuna opzione specificata.

Il valore iniziale di questo campo di immissione è DPDELF.

DPSID (numero intero con segno a 10 cifre)

Eliminare la struttura delle opzioni della proprietà del messaggio - campo DPSID.

Questo è l'identificativo della struttura. Il valore deve essere:

DPSIDV
Identificativo per la struttura delle opzioni della proprietà del messaggio di eliminazione.

Questo campo è sempre un campo di input. Il valore iniziale di questo campo è DPSIDV.

DPVER (numero intero con segno a 10 cifre)

Eliminare la struttura delle opzioni della proprietà del messaggio - campo DPVER.

Questo è il numero di versione della struttura. Il valore deve essere:

DPVER1
Numero di versione per la struttura di opzioni della proprietà di eliminazione del messaggio.

La seguente costante specifica il numero di versione della versione corrente:

VERDP
Versione corrente della struttura di opzioni della proprietà di eliminazione del messaggio.

Questo campo è sempre un campo di input. Il valore iniziale di questo campo è DPVER1.

Valori iniziali

Tabella 699. Campi in MQDPMO

Nome campo	Nome della costante	Valore della costante
DPSID	DPSIDV	'DMPO'
DPVER	DPVER1	1
DPOPT	Opzioni che controllano l'azione di MQDLTMP	DPNONE

Dichiarazione RPG

```
D* MQDMPO Structure
D*
D*
D* Structure identifier
D  DPSID                  1      4    INZ('DMPO')
D*
D* Structure version number
D  DPVER                  5      8I 0 INZ(1)
D*
** Options that control the action of
D* MQDLTMP
D  DPOPT                  9     12I 0 INZ(0)
```

IBM i MQEPH (Embedded PCF header) in IBM i

Panoramica

Finalità
La struttura MQEPH descrive i dati aggiuntivi che sono presenti in un messaggio quando tale messaggio è un messaggio PCF (programmable command format). Il campo *EPPFH* definisce i parametri PCF che seguono questa struttura e ciò consente di seguire i dati del messaggio PCF con altre intestazioni.

Nome formato
EPFMT

Serie di caratteri e codifica
I dati in MQEPH devono essere nella serie di caratteri e nella codifica del gestore code locale; ciò viene fornito dall'attributo del gestore code **CCSID** .

Impostare la serie di caratteri e la codifica di MQEPH nei campi *MDCSI* e *MDENC* in:

- MQMD (se la struttura MQEPH si trova all'inizio dei dati del messaggio) oppure
- La struttura dell'intestazione che precede la struttura MQEPH (tutti gli altri casi).

Utilizzo
Non è possibile utilizzare strutture MQEPH per inviare comandi al server dei comandi o a qualsiasi altro server di accettazione PCF del gestore code.

Allo stesso modo, il server dei comandi o qualsiasi altro server di accettazione PCF del gestore code non genera risposte o eventi contenenti strutture MQEPH.

- "Campi" a pagina 1103
- "Valori iniziali" a pagina 1105
- "Dichiarazione RPG" a pagina 1105

Campi

La struttura MQEPH contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

EPCSI (numero intero con segno a 10 cifre)

Questo è l'identificativo della serie di caratteri dei dati che seguono la struttura MQEPH e i parametri PCF associati; non si applica ai dati carattere nella stessa struttura MQEPH.

Il valore iniziale di questo campo è EPCUND.

EPENC (numero intero con segno a 10 cifre)

Questa è la codifica numerica dei dati che seguono la struttura MQEPH e i parametri PCF associati; non si applica ai dati carattere nella struttura MQEPH stessa.

Il valore iniziale di questo campo è 0.

EPFLG (numero intero con segno a 10 cifre)

Sono disponibili i seguenti lavori:

EPNONE
Non è stato specificato alcun indicatore. *MDCSI* EPNONE è definito per aiutare la documentazione del programma. Non è previsto che questa costante venga utilizzata con altre, ma poiché il valore è zero, tale utilizzo non può essere rilevato.

EPCSEM
La serie di caratteri dei parametri che contengono i dati carattere viene specificata singolarmente all'interno del campo *CCSID* in ciascuna struttura. La serie di caratteri dei campi *EPSID* e *EPFMT* è definita da *CCSID* nella struttura dell'intestazione precedente alla struttura MQEPH o dal campo *MDCSI* in MQMD se MQEPH è all'inizio del messaggio.

Il valore iniziale di questo campo è EPNONE.

EPFMT (stringa di caratteri a 8 byte)

Questo è il nome formato dei dati che seguono la struttura MQEPH e i parametri PCF associati.

Il valore iniziale di questo campo è EPFMNO.

EPLEN (numero intero con segno a 10 cifre)

Questa è la quantità di dati che precedono la successiva struttura dell'intestazione. Comprendono:

- La lunghezza dell'intestazione MQEPH
- La lunghezza di tutti i parametri PCF che seguono l'intestazione
- Qualsiasi riempimento vuoto che segue tali parametri

EPLEN deve essere un multiplo di 4.

La parte a lunghezza fissa della struttura è definita da EPSTLF.

Il valore iniziale di questo campo è 68.

EPPCFH (MQCFH)

Si tratta dell'intestazione PCF (Programmable Command Format), che definisce i parametri PCF che seguono la struttura MQEPH. Ciò consente di seguire i dati del messaggio PCF con altre intestazioni.

L'intestazione PCF viene definita inizialmente con i valori seguenti:

Tabella 700. Campi in EPPCFH

Nome campo	Nome della costante	Valore della costante
EP3TYP	CFTNON	0
EP3LEN	FHLENV	36
EP3VER	FHVER3	3
EP3CMD	CMNONE	0
EP3SEQ	Nessuna	1
EP3CTL	CCLST	1
EEP3CC	CCOK	0
EP3REA	RCNONE	0
EP3CNT	Nessuna	0

L'applicazione deve modificare EP3TYP da CFTNON a un tipo di struttura valido per l'utilizzo dell'intestazione PCF incorporata.

EPSID (stringa di caratteri a 4 byte)

Il valore deve essere:

IDEPST
Identificativo per la struttura dell'intestazione PCF incorporata.

Il valore iniziale di questo campo è EPSTID.

EPVER (numero intero con segno a 10 cifre)

Il valore può essere:

EPVER1
Numero versione per la struttura di intestazione PCF incorporata.

La seguente costante specifica il numero di versione della versione corrente:

EPVER3
La versione corrente della struttura dell'intestazione PCF incorporata.

Il valore iniziale di questo campo è EPVER3.

Valori iniziali

Tabella 701. Campi in MQEPH		
Nome campo	**Nome della costante**	**Valore della costante**
EPSID	IDEPST	'EP¬¬'
EPVER	EPVER1	1
EPLEN	ESTLF	68
EPENC	Nessuna	0
EPCSI	EPCUND	0
EPFMT	N. FPF	Spazi
EPFLG	EPNONE	0
EPPCFH	Nomi e valori come definiti in Tabella 700 a pagina 1104	0

Nota:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQEPH Structure
D*
D* Structure identifier
D  EPSID                  1      4
D* Structure version number
D  EPVER                  5      8I 0
D* Total lenght of MQEPH including MQCFHand parameter structures
D* that follow
D  EPLEN                  9     12I 0
D* Numeric encoding of data that follows last PCF parameter structure
D  EPENC                 13     16I 0
D* Character set identifier of data that follows last PCF parameter
D* structure
D  EPCSI                 17     20I 0
D* Format name of data that follows last PCF parameter structure
D  EPFMT                 21     28
D* Flags
D  EPFLG                 29     32I 0
D* Programmable Command Format Header
D  EP3TYP                33     36I 0
D  EP3LEN                37     40I 0
D  EP3VER                41     44I 0
D  EP3CMD                45     48I 0
D  EP3SEQ                49     52I 0
D  EP3CTL                53     56I 0
D  EP3CC                 57     60I 0
D  EP3REA                61     64I 0
D  EP3CNT                65     68I 0
```

IBM i MQGMO (opzioni Get - message) su IBM i

La struttura MQGMO consente all'applicazione di specificare le opzioni che controllano la modalità di rimozione dei messaggi dalle code.

Panoramica

Finalità
La struttura è un parametro di input / output nella chiamata MQGET.

Versione
La versione corrente di MQGMO è GMVER4. I campi che esistono solo nelle versioni più recenti della struttura sono identificati come tali nelle descrizioni che seguono.

Il file COPY fornito contiene la versione più recente di MQGMO supportata dall'ambiente, ma con il valore iniziale del campo *GMVER* impostato su GMVER1. Per utilizzare i campi che non sono presenti nella struttura version-1 , l'applicazione deve impostare il campo *GMVER* sul numero di versione della versione richiesta.

Serie di caratteri e codifica
I dati in MQGMO devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da ENNAT. Tuttavia, se l'applicazione è in esecuzione come un client IBM MQ , la struttura deve essere nella serie di caratteri e nella codifica del client.

- "Campi" a pagina 1106
- "Valori iniziali" a pagina 1127
- "Dichiarazione RPG" a pagina 1127

Campi

La struttura MQGMO contiene i seguenti campi; i campi sono descritti in ordine alfabetico:

GMGST (stringa di caratteri a 1 byte)

Indicatore che indica se il messaggio richiamato si trova in un gruppo.

Ha uno dei seguenti valori:

GSNIG
Il messaggio non è in un gruppo.

GSMIG
Il messaggio si trova in un gruppo, ma non è l'ultimo nel gruppo.

GSLMIG
Il messaggio è l'ultimo nel gruppo.

Questo valore è anche il valore restituito se il gruppo è composto da un solo messaggio.

Questo campo è un campo di output. Il valore iniziale di questo campo è GSNIG. Questo campo viene ignorato se *GMVER* è minore di GMVER2.

GMMH (numero intero con segno a 10 cifre)

handle del messaggio

Se viene specificata l'opzione GMPRAQ e l'attributo della coda PRPCTL non è impostato su PRPRFH, questo è l'handle di un messaggio che viene popolato con le proprietà del messaggio richiamato dalla coda. L'handle viene creato da una chiamata MQCRTMH. Tutte le proprietà già associate all'handle vengono eliminate prima di richiamare un messaggio.

È anche possibile specificare il seguente valore:
MQHM_NONE
Nessun handle del messaggio fornito.

Non è richiesto alcun descrittore di messaggi nella chiamata MQGET se viene fornito un handle del messaggio valido e utilizzato nell'output per contenere le proprietà del messaggio, il descrittore del messaggio associato all'handle del messaggio viene utilizzato per i campi di input.

Se viene specificato un descrittore del messaggio nella chiamata MQGET, ha sempre la precedenza sul descrittore del messaggio associato a un handle del messaggio.

Se viene specificato GMPRRF o GMPRAQ e l'attributo della coda PRPCTL è PRPRFH, la chiamata ha esito negativo con codice di errore RC2026 quando non viene specificato alcun parametro del descrittore del messaggio.

Al ritorno dalla chiamata MQGET, le proprietà e il descrittore del messaggio associati a questo handle del messaggio vengono aggiornati in modo da riflettere lo stato del messaggio richiamato (oltre al descrittore del messaggio se ne è stato fornito uno nella chiamata MQGET). Le proprietà del messaggio possono essere interrogate utilizzando la chiamata MQINQMP.

Fatta eccezione per le estensioni del descrittore del messaggio, quando presente, una proprietà che può essere interrogata con la chiamata MQINQMP non è contenuta nei dati del messaggio; se il messaggio sulla coda conteneva proprietà nei dati del messaggio, queste vengono rimosse dai dati del messaggio prima che i dati vengano restituiti all'applicazione.

Se non viene fornito alcun handle del messaggio o la versione è inferiore a GMVER4 , è necessario fornire un descrittore del messaggio valido nella chiamata MQGET. Tutte le proprietà del messaggio (tranne quelle contenute nel descrittore del messaggio) vengono restituite nei dati del messaggio in base al valore delle opzioni della proprietà nella struttura MQGMO e nell'attributo della coda PRPCTL.

Questo campo è sempre un campo di input. Il valore iniziale di questo campo è HMNONE. Questo campo viene ignorato se *GMVER* è inferiore a GMVER4.

GMMO (numero intero con segno a 10 cifre)

Opzioni che controllano i criteri di selezione utilizzati per MQGET.

Queste opzioni consentono all'applicazione di scegliere quali campi nel parametri **MSGDSC** vengono utilizzati per selezionare il messaggio restituito dalla chiamata MQGET. L'applicazione imposta le opzioni richieste in questo campo e imposta i campi corrispondenti nel parametro **MSGDSC** sui valori richiesti per tali campi. Solo i messaggi che hanno tali valori in MQMD per il messaggio sono candidati per il richiamo utilizzando tale parametro **MSGDSC** nella chiamata MQGET. I campi per i quali non viene specificata l'opzione di corrispondenza vengono ignorati quando si seleziona il messaggio da restituire. Se nessun criterio di selezione deve essere utilizzato nella chiamata MQGET (ovvero, qualsiasi messaggio è accettabile), *GMMO* deve essere impostato su MONONE.

Se viene specificato GMLOGO, solo alcuni messaggi possono essere restituiti dalla successiva chiamata MQGET:

- Se non è presente alcun gruppo corrente o messaggio logico, solo i messaggi che hanno *MDSEQ* uguale a 1 e *MDOFF* uguale a 0 sono idonei per la restituzione. In questa situazione, è possibile utilizzare una o più delle seguenti opzioni per selezionare quale dei messaggi idonei è quello restituito:
  - MOMSGI
  - MOCORI
  - MOGRPI
- Se esiste un gruppo corrente o un messaggio logico, solo il messaggio successivo nel gruppo o il segmento successivo nel messaggio logico è idoneo per la restituzione e non può essere modificato specificando le opzioni MO*.

In entrambi i casi, le opzioni di corrispondenza non applicabili possono essere ancora specificate, ma il valore del campo pertinente nel parametro **MSGDSC** deve corrispondere al valore del campo corrispondente nel messaggio da restituire; la chiamata ha esito negativo con codice di errore RC2247 se questa condizione non è soddisfatta.

GMMO viene ignorato se si specifica GMMUC o GMBRWC.

È possibile specificare una o più delle seguenti opzioni:

MOMSGI
Richiamare il messaggio con l'identificativo di messaggio specificato.

Questa opzione specifica che il messaggio da recuperare deve avere un identificativo di messaggio che corrisponda al valore del campo *MDMID* nel parametro **MSGDSC** della chiamata MQGET. Questa

corrispondenza è in aggiunta a tutte le altre corrispondenze che potrebbero essere applicate (ad esempio, l'identificativo di correlazione).

Se questa opzione non viene specificata, il campo *MDMID* nel parametro **MSGDSC** viene ignorato e qualsiasi identificativo del messaggio corrisponde.

Nota: L'identificativo del messaggio MINONE è un valore speciale che corrisponde a qualsiasi identificativo del messaggio in MQMD per il messaggio. Pertanto, specificare MOMSGI con MINONE equivale a non specificare MOMSGI.

MOCORI
Richiamare il messaggio con l'identificativo di correlazione specificato.

Questa opzione specifica che il messaggio da richiamare deve avere un identificativo di correlazione che corrisponda al valore del campo *MDCID* nel parametro **MSGDSC** della chiamata MQGET. Questa corrispondenza si aggiunge a qualsiasi altra corrispondenza che potrebbe essere applicata (ad esempio, l'identificativo del messaggio).

Se questa opzione non viene specificata, il campo *MDCID* nel parametro **MSGDSC** viene ignorato e qualsiasi identificativo di correlazione corrisponde.

Nota: L'identificativo di correlazione CINONE è un valore speciale che corrisponde a qualsiasi identificativo di correlazione in MQMD per il messaggio. Pertanto, specificare MOCORI con CINONE equivale a non specificare MOCORI.

MOGRPI
Richiamare il messaggio con l'identificativo gruppo specificato.

Questa opzione specifica che il messaggio da richiamare deve avere un ID gruppo che corrisponda al valore del campo *MDGID* nel parametro **MSGDSC** della chiamata MQGET. Questa corrispondenza è in aggiunta a tutte le altre corrispondenze che potrebbero essere applicate (ad esempio, l'identificativo di correlazione).

Se questa opzione non è specificata, il campo *MDGID* nel parametro **MSGDSC** viene ignorato e qualsiasi identificativo di gruppo corrisponde.

Nota: L'identificativo del gruppo GINONE è un valore speciale che corrisponde a qualsiasi identificativo del gruppo nell'MQMD per il messaggio. Pertanto, specificare MOGRPI con GINONE equivale a non specificare MOGRPI.

MOSEQN
Richiamare il messaggio con il numero di sequenza messaggio specificato.

Questa opzione specifica che il messaggio da richiamare deve avere un numero di sequenza del messaggio che corrisponda al valore del campo *MDSEQ* nel parametro **MSGDSC** della chiamata MQGET. Questa corrispondenza è in aggiunta a tutte le altre corrispondenze che potrebbero essere applicate (ad esempio, l'identificativo del gruppo).

Se questa opzione non viene specificata, il campo *MDSEQ* nel parametro **MSGDSC** viene ignorato e qualsiasi numero di sequenza del messaggio corrisponde.

MOOFF
Richiamare il messaggio con l'offset specificato.

Questa opzione specifica che il messaggio da richiamare deve avere un offset che corrisponda al valore del campo *MDOFF* nel parametro **MSGDSC** della chiamata MQGET. Questa corrispondenza è in aggiunta a tutte le altre corrispondenze che potrebbero essere applicate (ad esempio, il numero di sequenza del messaggio).

Se questa opzione non viene specificata, il campo *MDOFF* nel parametro **MSGDSC** viene ignorato e qualsiasi offset corrisponde.

Se non viene specificata alcuna delle opzioni descritte, è possibile utilizzare la seguente opzione:

MONONE
Nessuna corrispondenza.

Questa opzione specifica che non devono essere utilizzate corrispondenze nella selezione del messaggio da restituire; pertanto, tutti i messaggi sulla coda sono idonei per il richiamo (ma soggetti al controllo delle opzioni GMAMSA, GMASGA e GMCMPM).

MONONE è definito per aiutare la documentazione del programma. Non è previsto che questa opzione venga utilizzata con qualsiasi altra opzione MO*, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Questo è un campo di immissione. Il valore iniziale di questo campo è MOMSG con MOCORI. Questo campo viene ignorato se *GMVER* è minore di GMVER2.

Nota: Il valore iniziale del campo *GMMO* è definito per la compatibilità con i gestori code di versioni precedenti. Tuttavia, quando si legge una serie di messaggi da una coda senza utilizzare i criteri di selezione, questo valore iniziale richiede che l'applicazione reimposti i campi *MDMID* e *MDCID* su MINONE e CINONE prima di ogni chiamata MQGET. La necessità di reimpostare *MDMID* e *MDCID* può essere evitata impostando *GMVER* su GMVER2e *GMMO* su MONONE.

GMOPT (numero intero con segno di 10 cifre)

Opzioni che controllano l'azione di MQGET.

È possibile specificare zero o più delle seguenti opzioni descritte. Se ne è richiesto più di uno, è possibile aggiungere i valori (non aggiungere la stessa costante più di una volta). Vengono annotate le combinazioni di opzioni non valide; tutte le altre combinazioni sono valide.

Opzioni di attesa: le seguenti opzioni sono relative all'attesa dell'arrivo dei messaggi sulla coda:

GMWT
Attendere l'arrivo del messaggio.

L'applicazione deve attendere l'arrivo di un messaggio appropriato. Il tempo massimo di attesa dell'applicazione è specificato in *GMWI*.

Se le richieste MQGET sono inibite o le richieste MQGET vengono inibite durante l'attesa, l'attesa viene annullata e la chiamata viene completata con CCFAIL e codice di errore RC2016, indipendentemente dal fatto che vi siano messaggi idonei sulla coda.

Questa opzione può essere utilizzata con le opzioni GMBRWF o GMBRWN.

Se diverse applicazioni sono in attesa sulla stessa coda condivisa, l'applicazione o le applicazioni che vengono attivate quando arriva un messaggio appropriato vengono descritte più avanti in questa sezione.

Nota: Nella seguente descrizione, una chiamata MQGET di esplorazione è una chiamata che specifica una delle opzioni di esplorazione, ma non GMLK; una chiamata MQGET che specifica l'opzione GMLK viene considerata come una chiamata non di esplorazione.

- Se una o più chiamate MQGET non sono in attesa, ma nessuna chiamata MQGET è in attesa, ne viene attivata una.
- Se una o più chiamate MQGET di ricerca sono in attesa, ma nessuna chiamata MQGET non di ricerca è in attesa, tutte vengono attivate.
- Se una o più chiamate MQGET non sfogliate e una o più chiamate MQGET non sfogliate sono in attesa, viene attivata una chiamata MQGET non sfogliata e nessuna, alcune o tutte le chiamate MQGET di esplorazione. (Il numero di chiamate MQGET di ricerca attivate non può essere previsto, perché dipende dalle considerazioni di pianificazione del sistema operativo e da altri fattori.)

Se più di una chiamata MQGET non browse è in attesa sulla stessa coda, ne viene attivata solo una; in questa situazione il gestore code tenta di dare la priorità alle chiamate non browse in attesa nel seguente ordine:

1. Richieste get - wait specifiche che possono essere soddisfatte solo da determinati messaggi, ad esempio, quelli con uno specifico *MDMID* o *MDCID* (o entrambi).
2. Richieste get - wait generali che possono essere soddisfatte da qualsiasi messaggio.

Si segnalano i seguenti punti:

- All'interno della prima categoria, non viene assegnata alcuna priorità aggiuntiva alle richieste get - wait più specifiche, ad esempio quelle che specificano *MDMID* e *MDCID*.
- All'interno di entrambe le categorie, non è possibile prevedere quale applicazione è selezionata. In particolare, l'applicazione in attesa più lunga non è necessariamente quella selezionata.
- La lunghezza del percorso e le considerazioni sulla pianificazione della priorità del sistema operativo possono indicare che un'applicazione in attesa con priorità del sistema operativo inferiore a quella prevista richiama il messaggio.
- Inoltre, è possibile che un'applicazione che non è in attesa richiami il messaggio piuttosto che uno.

GMWT viene ignorato se specificato con GMBRWC o GMMUC; non viene generato alcun errore.

GMNWT
Restituisci immediatamente se non è presente alcun messaggio adatto.

L'applicazione non deve attendere se non è disponibile alcun messaggio adatto. Questo è il contrario dell'opzione GMWT ed è definito per aiutare la documentazione del programma. È il valore predefinito se non viene specificato nessuno dei due.

GMFIQ
Errore se il gestore code è in fase di sospensione.

Questa opzione forza l'esito negativo della chiamata MQGET se il gestore code è in stato di inattività.

Se questa opzione viene specificata insieme a GMWT, e l'attesa è in sospeso nel momento in cui il gestore code entra nello stato di sospensione:

- L'attesa viene annullata e la chiamata restituisce il codice di completamento CCFAIL con codice motivo RC2161 .

Se GMFIQ non è specificato e il gestore code entra nello stato di inattività, l'attesa non viene annullata.

Opzioni punto di sincronizzazione: le seguenti opzioni sono relative alla partecipazione della chiamata MQGET all'interno di un'unità di lavoro:

GMSYP
Richiamare il messaggio con il controllo del punto di sincronizzazione.

La richiesta è di operare all'interno dei normali protocolli di unità di lavoro. Il messaggio è contrassegnato come non disponibile per altre applicazioni, ma viene eliminato dalla coda solo quando viene eseguito il commit dell'unità di lavoro. Il messaggio viene reso nuovamente disponibile se viene eseguito il backout dell'unità di lavoro.

Se questa opzione o GMNSYP non è specificata, la richiesta di acquisizione non si trova all'interno di un'unità di lavoro.

Questa opzione non è valida con nessuna delle seguenti opzioni:

- GMBRWF
- GMBRWC
- GMBRWN
- GMLK
- GMNSYP
- GMPSYP
- GMUNLK

GMPSYP
Richiamare il messaggio con il controllo del punto di sincronizzazione se il messaggio è persistente.

La richiesta è di operare all'interno dei normali protocolli dell'unità di lavoro, ma solo se il messaggio richiamato è persistente. Un messaggio persistente ha il valore PEPER nel campo *MDPER* in MQMD.

- Se il messaggio è persistente, il gestore code elabora la chiamata come se l'applicazione avesse specificato GMSYP.
- Se il messaggio non è persistente, il gestore code elabora la chiamata come se l'applicazione avesse specificato GMNSYP (per i dettagli, consultare la seguente sezione).

Questa opzione non è valida con nessuna delle seguenti opzioni:

- GMBRWF
- GMBRWC
- GMBRWN
- GMCMM
- GMNSYP
- GMSYP
- GMUNLK

GMNSYP
Richiama il messaggio senza controllo del punto di sincronizzazione.

La richiesta è di operare al di fuori dei normali protocolli di unità di lavoro. Il messaggio viene eliminato dalla coda immediatamente (a meno che non si tratti di una richiesta di ricerca). Il messaggio non può essere reso nuovamente disponibile eseguendo il backout dell'unità di lavoro.

Questa opzione viene assunta se viene specificato GMBRWF o GMBRWN.

Se questa opzione e GMSYP non vengono specificati, la richiesta di acquisizione non si trova all'interno di un'unità di lavoro.

Questa opzione non è valida con nessuna delle seguenti opzioni:

- GMSYP
- GMPSYP

Opzioni di ricerca: le seguenti opzioni sono relative alla ricerca dei messaggi sulla coda:

GMBRWF
Sfoglia dall'inizio della coda.

Quando una coda viene aperta con opzione OOBRW, viene stabilito un cursore di ricerca, posizionato logicamente prima del primo messaggio sulla coda. Le successive chiamate MQGET che specificano l'opzione GMBRWF, GMBRWN o GMBRWC possono essere utilizzate per richiamare i messaggi dalla coda in modo non distruttivo. Il cursore di ricerca contrassegna la posizione, all'interno dei messaggi sulla coda, da cui la successiva chiamata MQGET con GMBRWN ricerca un messaggio adatto.

Una chiamata MQGET con GMBRWF fa sì che la precedente posizione del cursore di ricerca venga ignorata. Viene richiamato il primo messaggio sulla coda che soddisfa le condizioni specificate nel descrittore del messaggio. Il messaggio rimane nella coda e il cursore di ricerca è posizionato su questo messaggio.

Dopo questa chiamata, il cursore di ricerca viene posizionato sul messaggio che è stato restituito. Se il messaggio viene rimosso dalla coda prima dell'emissione della successiva chiamata MQGET con GMBRWN, il cursore di ricerca rimane nella posizione nella coda occupata dal messaggio, anche se tale posizione è ora vuota.

L'opzione GMMUC può quindi essere utilizzata con una chiamata MQGET non browse, se necessario, per rimuovere il messaggio dalla coda.

Il cursore di esplorazione non viene spostato da una chiamata MQGET non di esplorazione utilizzando lo stesso handle *HOBJ* . Inoltre, non viene spostato da una chiamata MQGET di

esplorazione che restituisce un codice di completamento di CCFAIL o un codice motivo di RC2080 .

L'opzione GMLK può essere specificata insieme a questa opzione, in modo che il messaggio visualizzato venga bloccato.

GMBRWF può essere specificato con qualsiasi combinazione valida delle opzioni GM* e MO* che controllano l'elaborazione dei messaggi in gruppi e segmenti di messaggi logici.

Se viene specificato GMLOGO, i messaggi vengono esaminati in ordine logico. Se tale opzione viene omessa, i messaggi vengono esaminati in ordine fisico. Quando GMBRWF viene specificato, è possibile passare dall'ordine logico all'ordine fisico, ma le chiamate MQGET successive che utilizzano GMBRWN devono esplorare la coda nello stesso ordine della chiamata più recente che ha specificato GMBRWF per l'handle della coda.

Le informazioni sul gruppo e sul segmento che il gestore code conserva per le chiamate MQGET che sfogliano i messaggi sulla coda, sono separate dal gruppo e dalle informazioni sul segmento che il gestore code conserva per le chiamate MQGET che rimuovono i messaggi dalla coda. Quando viene specificato GMBRWF, il gestore code ignora le informazioni sul gruppo e sul segmento per la ricerca e esegue la scansione della coda come se non fosse presente alcun gruppo corrente e nessun messaggio logico corrente. Se la chiamata MQGET ha esito positivo (codice di completamento CCOK o CCWARN), le informazioni sul gruppo e sul segmento per la ricerca vengono impostate su quelle del messaggio restituito; se la chiamata ha esito negativo, le informazioni sul gruppo e sul segmento rimangono le stesse che erano prima della chiamata.

Questa opzione non è valida con nessuna delle seguenti opzioni:

- GMBRWC
- GMBRWN
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

È anche un errore se la coda non è stata aperta per la ricerca.

GMBRWN
Ricerca dalla posizione corrente nella coda.

Il cursore di esplorazione è avanzato al successivo messaggio sulla coda che soddisfa i criteri di selezione specificati nella chiamata MQGET. Il messaggio viene restituito all'applicazione, ma rimane nella coda.

Dopo che una coda è stata aperta per la ricerca, la prima chiamata di ricerca che utilizza l'handle ha lo stesso effetto se specifica l'opzione GMBRWF o GMBRWN.

Se il messaggio viene rimosso dalla coda prima che venga emessa la successiva chiamata MQGET con GMBRWN, il cursore di esplorazione rimane logicamente nella posizione nella coda occupata dal messaggio, anche se tale posizione è ora vuota.

I messaggi vengono memorizzati nella coda in due modi:

- FIFO entro la priorità (MSPRIO) oppure
- FIFO indipendentemente dalla priorità (MSFIFO)

L'attributo della coda **MsgDeliverySequence** indica quale metodo si applica (consultare “Attributi per le code” a pagina 1406 per i dettagli).

Se la coda ha un *MsgDeliverySequence* di MSPRIO e un messaggio arriva sulla coda con una priorità superiore a quella attualmente indicata dal cursore di ricerca, tale messaggio non viene trovato durante lo sweep corrente della coda utilizzando GMBRWN. Può essere trovato solo dopo che il cursore di ricerca è stato reimpostato con GMBRWF (o riaprendo la coda).

L'opzione GMMUC può essere utilizzata in un secondo momento con una chiamata MQGET non browse, se necessario, per rimuovere il messaggio dalla coda.

Il cursore di esplorazione non viene spostato dalle chiamate MQGET non di esplorazione utilizzando lo stesso handle *HOBJ* .

L'opzione GMLK può essere specificata insieme a questa opzione, in modo che il messaggio visualizzato venga bloccato.

GMBRWN può essere specificato con qualsiasi combinazione valida delle opzioni GM* e MO* che controllano l'elaborazione dei messaggi in gruppi e segmenti di messaggi logici.

Se viene specificato GMLOGO, i messaggi vengono esaminati in ordine logico. Se tale opzione viene omessa, i messaggi vengono esaminati in ordine fisico. Quando GMBRWF viene specificato, è possibile passare dall'ordine logico all'ordine fisico, ma le chiamate MQGET successive che utilizzano GMBRWN devono esplorare la coda nello stesso ordine della chiamata più recente che ha specificato GMBRWF per l'handle della coda. La chiamata ha esito negativo con codice di errore RC2259 se questa condizione non viene soddisfatta.

Nota: È necessario prestare particolare attenzione se viene utilizzata una chiamata MQGET per esplorare oltre la fine di un gruppo di messaggi (o un messaggio logico non in un gruppo) quando GMLOGO non è specificato. Ad esempio, se l'ultimo messaggio nel gruppo precede il primo messaggio nel gruppo sulla coda, utilizzando GMBRWN per la ricerca oltre la fine del gruppo, specificando MOSEQN con *MDSEQ* impostato su 1 (per trovare il primo messaggio del gruppo successivo) restituisce nuovamente il primo messaggio nel gruppo già esplorato. Ciò potrebbe verificarsi immediatamente o in un numero di chiamate MQGET successive (se sono presenti gruppi che intervengono).

La possibilità di un loop infinito può essere evitata aprendo la coda due volte per sfogliare:

- Utilizzare il primo handle per ricercare solo il primo messaggio in ciascun gruppo.
- Utilizzare il secondo handle per ricercare solo i messaggi all'interno di un determinato gruppo.
- Utilizzare le opzioni MO* per spostare il secondo cursore di ricerca nella posizione del primo cursore di ricerca, prima di esaminare i messaggi nel gruppo.
- Non utilizzare GMBRWN per esplorare oltre la fine di un gruppo.

Le informazioni sul gruppo e sul segmento che il gestore code conserva per le chiamate MQGET che sfogliano i messaggi sulla coda, sono separate dal gruppo e dalle informazioni sul segmento che conserva per le chiamate MQGET che rimuovono i messaggi dalla coda.

Questa opzione non è valida con nessuna delle seguenti opzioni:

- GMBRWF
- GMBRWC
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

È anche un errore se la coda non è stata aperta per la ricerca.

GMBRWC
Sfoglia messaggio sotto il cursore di ricerca.

Questa opzione fa in modo che il messaggio puntato dal cursore di esplorazione venga richiamato in modo non distruttivo, indipendentemente dalle opzioni MO* specificate nel campo *GMMO* in MQGMO.

Il messaggio indicato dal cursore di ricerca è quello che è stato richiamato per ultimo utilizzando l'opzione GMBRWF o GMBRWN. La chiamata ha esito negativo se nessuna di queste chiamate è stata emessa per questa coda da quando è stata aperta o se il messaggio che si trovava sotto il cursore di ricerca è stato richiamato in modo distruttivo.

La posizione del cursore di ricerca non viene modificata da questa chiamata.

L'opzione GMMUC può quindi essere utilizzata con una chiamata MQGET non browse, se necessario, per rimuovere il messaggio dalla coda.

Il cursore di esplorazione non viene spostato da una chiamata MQGET non di esplorazione utilizzando lo stesso handle *HOBJ* . Non viene nemmeno spostato da una chiamata MQGET di esplorazione che restituisce un codice di completamento di CCFAIL o un codice motivo di RC2080.

Se GMBRWC è specificato con GMLK:

- Se c'è già un messaggio bloccato, deve essere quello sotto il cursore, in modo che venga restituito senza sbloccarlo e ribloccarlo; il messaggio rimane bloccato.
- Se non è presente alcun messaggio bloccato, il messaggio sotto il cursore di ricerca (se è presente) viene bloccato e restituito all'applicazione; se non è presente alcun messaggio sotto il cursore di ricerca, la chiamata ha esito negativo.

Se GMBRWC viene specificato senza GMLK:

- Se esiste già un messaggio bloccato, deve essere quello sotto il cursore. Questo messaggio viene restituito all'applicazione e sbloccato. Poiché il messaggio è ora sbloccato, non vi è alcuna garanzia che possa essere esplorato di nuovo o richiamato in modo distruttivo (potrebbe essere richiamato in modo distruttivo da un'altra applicazione che riceve i messaggi dalla coda).
- Se non è presente alcun messaggio bloccato, il messaggio sotto il cursore di ricerca (se è presente) viene restituito all'applicazione; se non è presente alcun messaggio sotto il cursore di ricerca, la chiamata ha esito negativo.

Se GMCMPM viene specificato con GMBRWC, il cursore di ricerca deve identificare un messaggio con un campo *MDOFF* in MQMD uguale a zero. Se questa condizione non viene soddisfatta, la chiamata ha esito negativo con codice di errore RC2246 .

Le informazioni sul gruppo e sul segmento che il gestore code conserva per le chiamate MQGET che sfogliano i messaggi sulla coda, sono separate dal gruppo e dalle informazioni sul segmento che conserva per le chiamate MQGET che rimuovono i messaggi dalla coda.

Questa opzione non è valida con nessuna delle seguenti opzioni:

- GMBRWF
- GMBRWN
- GMMUC
- GMSYP
- GMPSYP
- GMUNLK

È anche un errore se la coda non è stata aperta per la ricerca.

GMMUC
Richiamare il messaggio sotto il cursore di ricerca.

Questa opzione fa sì che il messaggio puntato dal cursore di esplorazione venga richiamato, indipendentemente dalle opzioni MO* specificate nel campo *GMMO* in MQGMO. Il messaggio viene rimosso dalla coda.

Il messaggio indicato dal cursore di ricerca è quello che è stato richiamato per ultimo utilizzando l'opzione GMBRWF o GMBRWN.

Se GMCMPM viene specificato con GMMUC, il cursore di esplorazione deve identificare un messaggio con un campo *MDOFF* in MQMD che sia zero. Se questa condizione non viene soddisfatta, la chiamata ha esito negativo con codice di errore RC2246 .

Questa opzione non è valida con nessuna delle seguenti opzioni:

- GMBRWF

- GMBRWC
- GMBRWN
- GMUNLK

È anche un errore se la coda non è stata aperta sia per la ricerca che per l'input. Se il cursore di esplorazione non punta attualmente a un messaggio richiamabile, viene restituito un errore dalla chiamata MQGET.

Opzioni di blocco: le seguenti opzioni sono relative al blocco dei messaggi sulla coda:

GMLK
Messaggio di blocco.

Questa opzione blocca il messaggio visualizzato, in modo che il messaggio diventi invisibile a qualsiasi altro handle aperto per la coda. L'opzione può essere specificata solo se viene specificata anche una delle seguenti opzioni:

- GMBRWF
- GMBRWN
- GMBRWC

È possibile bloccare un solo messaggio per ogni gestore code, ma questo può essere un messaggio logico o un messaggio fisico:

- Se viene specificato GMCMPM, tutti i segmenti del messaggio che costituiscono il messaggio logico vengono bloccati nell'handle della coda (se sono tutti presenti nella coda e sono disponibili per il richiamo).
- Se GMCMPM non è specificato, solo un singolo messaggio fisico è bloccato per l'handle della coda. Se questo messaggio è un segmento di un messaggio logico, il segmento bloccato impedisce ad altre applicazioni di utilizzare GMCMPM per richiamare o sfogliare il messaggio logico.

Il messaggio bloccato è sempre quello sotto il cursore di esplorazione e il messaggio può essere rimosso dalla coda da una successiva chiamata MQGET che specifica l'opzione GMMUC. Anche altre chiamate MQGET che utilizzano l'handle della coda possono rimuovere il messaggio (ad esempio, una chiamata che specifica l'identificativo del messaggio bloccato).

Se la chiamata restituisce il codice di completamento CCFAIL o CCWARN con codice motivo RC2080, nessun messaggio è bloccato.

Se l'applicazione decide di non rimuovere il messaggio dalla coda, il blocco viene rilasciato da:

- Emissione di un'altra chiamata MQGET per questo handle, con GMBRWF o GMBRWN specificati (con o senza GMLK); il messaggio viene sbloccato se la chiamata viene completata con CCOK o CCWARN, ma rimane bloccato se la chiamata viene completata con CCFAIL. Tuttavia, si applicano le seguenti eccezioni:
  - Il messaggio non viene sbloccato se CCWARN viene restituito con RC2080.
  - Il messaggio viene sbloccato se CCFAIL viene restituito con RC2033.

  Se viene specificato anche GMLK, il messaggio restituito è bloccato. Se GMLK non è specificato, non c'è alcun messaggio bloccato dopo la chiamata.

  Se viene specificato GMWT e non è immediatamente disponibile alcun messaggio, lo sblocco del messaggio originale si verifica prima dell'inizio dell'attesa (a condizione che la chiamata sia altrimenti priva di errore).
- Emissione di un'altra chiamata MQGET per questo handle, con GMBRWC (senza GMLK); il messaggio viene sbloccato se la chiamata viene completata con CCOK o CCWARN, ma rimane bloccato se la chiamata viene completata con CCFAIL. Tuttavia, si applica la seguente eccezione:
  - Il messaggio non viene sbloccato se CCWARN viene restituito con RC2080.

- Emissione di un'altra chiamata MQGET per questo handle con GMUNLK.
- Emissione di una chiamata MQCLOSE per questo handle (esplicitamente o implicitamente alla fine dell'applicazione).

Non è richiesta alcuna opzione di apertura speciale per specificare questa opzione, diversa da OOBRW, che è necessaria per specificare l'opzione di ricerca di accompagnamento.

Questa opzione non è valida con nessuna delle seguenti opzioni:

- GMSYP
- GMPSYP
- GMUNLK

GMUNLK
Sblocca messaggio.

Il messaggio da sbloccare deve essere stato precedentemente bloccato da una chiamata MQGET con l'opzione GMLK. Se non esiste alcun messaggio bloccato per questo handle, la chiamata viene completata con CCWARN e RC2209 .

I parametri **MSGDSC**, **BUFLEN**, **BUFFER**e **DATLEN** non vengono controllati o modificati se viene specificato GMUNLK. Nessun messaggio viene restituito in *BUFFER*.

Non è richiesta alcuna opzione di apertura speciale per specificare questa opzione (anche se OOBRW è necessario per emettere la richiesta di blocco in primo luogo).

Questa opzione non è valida con le opzioni tranne le seguenti:

- GMNWT
- GMNSYP

Si presuppone che entrambe queste opzioni siano specificate o meno.

Opzioni dati messaggio: le seguenti opzioni sono relative all'elaborazione dei dati del messaggio quando il messaggio viene letto dalla coda:

GMATM
Consente il troncamento dei dati del messaggio.

Se il buffer del messaggio è troppo piccolo per contenere il messaggio completo, questa opzione consente alla chiamata MQGET di riempire il buffer con la quantità di messaggio che il buffer può contenere, emettere un codice di completamento di avvertenza e completare l'elaborazione. Ciò significa:

- Quando si sfogliano i messaggi, il cursore di esplorazione viene avanzato al messaggio restituito.
- Quando si rimuovono i messaggi, il messaggio restituito viene rimosso dalla coda.
- Se non si verifica alcun altro errore, viene restituito il codice di errore RC2079 .

Senza questa opzione, il buffer viene ancora riempito con la quantità di messaggi che può contenere, viene emesso un codice di completamento di avvertenza, ma l'elaborazione non viene completata. Ciò significa:

- Quando si sfogliano i messaggi, il cursore di esplorazione non è avanzato.
- Quando si rimuovono i messaggi, il messaggio non viene rimosso dalla coda.
- Se non si verifica alcun altro errore, viene restituito il codice di errore RC2080 .

GMCONV
Convertire i dati del messaggio.

Questa opzione richiede la conversione dei dati dell'applicazione nel messaggio in modo che siano conformi ai valori *MDCSI* e *MDENC* specificati nel parametro **MSGDSC** nella chiamata MQGET, prima che i dati vengano copiati nel parametro **BUFFER** .

Il campo *MDFMT* specificato quando il messaggio è stato inserito viene assunto dal processo di conversione per identificare la natura dei dati nel messaggio. La conversione dei dati del messaggio viene effettuata dal gestore code per i formati integrati e da un'uscita scritta dall'utente per altri formati.

- Se la conversione viene eseguita correttamente, i campi *MDCSI* e *MDENC* specificati nel parametro **MSGDSC** non vengono modificati al ritorno dalla chiamata MQGET.
- Se la conversione non può essere eseguita con esito positivo (ma la chiamata MQGET viene altrimenti completata senza errori), i dati del messaggio vengono restituiti non convertiti e i campi *MDCSI* e *MDENC* in *MSGDSC* sono impostati sui valori per il messaggio non convertito. In questo caso il codice di completamento è CCWARN.

In entrambi i casi, quindi, questi campi descrivono l'identificativo della serie di caratteri e la codifica dei dati del messaggio restituiti nel parametro **BUFFER** .

Consultare il campo *MDFMT* descritto in "MQMD (Message Descriptor) su IBM i" a pagina 1140 per un elenco dei nomi di formato per cui il gestore code esegue la conversione.

Opzioni di gruppi e segmenti: le seguenti opzioni sono relative all'elaborazione di messaggi in gruppi e segmenti di messaggi logici. Queste definizioni potrebbero essere utili per comprendere le opzioni:

Messaggio fisico
Questa è l'unità di informazioni più piccola che può essere inserita o rimossa da una coda; spesso corrisponde alle informazioni specificate o richiamate in una singola chiamata MQPUT, MQPUT1o MQGET. Ogni messaggio fisico ha il proprio descrittore di messaggio (MQMD). Generalmente, i messaggi fisici sono distinti da valori differenti per l'identificativo del messaggio (campo*MDMID* in MQMD), sebbene ciò non venga applicato dal gestore code.

Messaggio logico
Questa è una singola unità di informazioni sull'applicazione. In assenza di restrizioni di sistema, un messaggio logico sarebbe lo stesso di un messaggio fisico. Ma quando i messaggi logici sono grandi, i vincoli di sistema potrebbero rendere consigliabile o necessario suddividere un messaggio logico in due o più messaggi fisici, denominati segmenti.

Un messaggio logico che è stato segmentato è costituito da due o più messaggi fisici che hanno lo stesso identificativo del gruppo non null (campo*MDGID* in MQMD) e lo stesso numero di sequenza del messaggio (campo*MDSEQ* in MQMD). I segmenti sono distinti da valori differenti per l'offset del segmento (campo*MDOFF* in MQMD), che fornisce l'offset dei dati nel messaggio fisico dall'inizio dei dati nel messaggio logico. Poiché ogni segmento è un messaggio fisico, i segmenti in un messaggio logico generalmente hanno identificativi di messaggio differenti.

Un messaggio logico che non è stato segmentato, ma per cui la segmentazione è stata consentita dall'applicazione mittente, ha anche un identificativo di gruppo non null, anche se in questo caso esiste solo un messaggio fisico con tale identificativo di gruppo se il messaggio logico non appartiene a un gruppo di messaggi. I messaggi logici per i quali la segmentazione è stata inibita dall'applicazione di invio hanno un identificativo di gruppo nullo (GINONE), a meno che il messaggio logico non appartenga ad un gruppo di messaggi.

Gruppo di messaggi
Questa è una serie di uno o più messaggi logici che hanno lo stesso identificativo di gruppo non null. I messaggi logici nel gruppo sono distinti da valori differenti per il numero di sequenza del messaggio, che è un numero intero compreso tra 1 e n, dove n è il numero di messaggi logici nel gruppo. Se uno o più messaggi logici sono segmentati, nel gruppo sono presenti più di n messaggi fisici.

GMLOGO
I messaggi in gruppi e segmenti di messaggi logici vengono restituiti in ordine logico.

Questa opzione controlla l'ordine in cui i messaggi vengono restituiti dalle successive chiamate MQGET per l'handle della coda. L'opzione deve essere specificata su ciascuna di queste chiamate per avere un effetto.

Se GMLOGO è specificato per le successive chiamate MQGET per l'handle della coda, i messaggi nei gruppi vengono restituiti nell'ordine fornito dai numeri di sequenza dei messaggi e i segmenti

dei messaggi logici vengono restituiti nell'ordine fornito dagli offset dei segmenti. Questo ordine potrebbe essere diverso dall'ordine in cui tali messaggi e segmenti si verificano nella coda.

Nota: La specifica di GMLOGO non ha conseguenze negative sui messaggi che non appartengono ai gruppi e che non sono segmenti. In effetti, tali messaggi vengono trattati come se ciascuno appartenesse a un gruppo di messaggi costituito da un solo messaggio. Quindi è perfettamente sicuro specificare GMLOGO quando si richiamano messaggi dalle code che potrebbero contenere una combinazione di messaggi in gruppi, segmenti di messaggi e messaggi non segmentati non in gruppi.

Per restituire i messaggi nell'ordine richiesto, il gestore code conserva le informazioni sul gruppo e sul segmento tra le successive chiamate MQGET. Queste informazioni identificano il gruppo di messaggi corrente e il messaggio logico corrente per il gestore code, la posizione corrente all'interno del gruppo e il messaggio logico e se i messaggi vengono richiamati all'interno di un'unità di lavoro. Poiché il gestore code conserva queste informazioni, non è necessario che l'applicazione imposti le informazioni sul gruppo e sul segmento prima di ogni chiamata MQGET. In particolare, significa che l'applicazione non deve impostare i campi *MDGID*, *MDSEQ*e *MDOFF* in MQMD. Tuttavia, l'applicazione deve impostare correttamente l'opzione GMSYP o GMNSYP su ogni chiamata.

Quando la coda viene aperta, non esiste alcun gruppo di messaggi corrente e nessun messaggio logico corrente. Un gruppo di messaggi diventa il gruppo di messaggi corrente quando un messaggio con l'indicatore MFMIG viene restituito dalla chiamata MQGET. Con GMLOGO specificato su chiamate successive, tale gruppo rimane il gruppo corrente fino a quando non viene restituito un messaggio che ha:

- MFLMIG senza MFSEG (ovvero, l'ultimo messaggio logico nel gruppo non è segmentato) oppure
- MFLMIG con MFLSEG (ossia, il messaggio restituito è l'ultimo segmento dell'ultimo messaggio logico nel gruppo).

Quando viene restituito un messaggio di questo tipo, il gruppo di messaggi viene terminato e, una volta completata con esito positivo la chiamata MQGET, non esiste più un gruppo corrente. In modo simile, un messaggio logico diventa il messaggio logico corrente quando un messaggio che ha l'indicatore MFSEG viene restituito dalla chiamata MQGET e tale messaggio logico viene terminato quando viene restituito il messaggio che ha l'indicatore MFLSEG.

Se non viene specificato alcun criterio di selezione, le chiamate MQGET successive restituiscono (nell'ordine corretto) i messaggi per il primo gruppo di messaggi sulla coda, quindi i messaggi per il secondo gruppo di messaggi e così via, fino a quando non sono disponibili ulteriori messaggi. È possibile selezionare i gruppi di messaggi particolari restituiti specificando una o più delle seguenti opzioni nel campo *GMMO* :

- MOMSGI
- MOCORI
- MOGRPI

Tuttavia, queste opzioni sono valide solo quando non è presente alcun gruppo di messaggi o messaggio logico corrente; consultare il campo *GMMO* descritto in questo argomento.

La Tabella 702 a pagina 1119 mostra i valori dei campi *MDMID*, *MDCID*, *MDGID*, *MDSEQ*e *MDOFF* che il gestore code cerca quando tenta di trovare un messaggio da restituire alla chiamata MQGET. Ciò si applica sia alla rimozione dei messaggi dalla coda che alla visualizzazione dei messaggi sulla coda. Le colonne della tabella hanno i seguenti significati:

ORD LOG
Indica se l'opzione GMLOGO è specificata nella chiamata.

Grp corrente
Indica se esiste un gruppo di messaggi corrente prima della chiamata.

Messaggio di log corrente
Indica se esiste un messaggio logico corrente prima della chiamata.

Altre colonne
Mostra i valori che il gestore code cerca. "Precedente" indica il valore restituito per il campo nel precedente messaggio per l'handle della coda.

Tabella 702. Opzioni MQGET relative ai messaggi in gruppi e segmenti di messaggi logici

Opzioni specificate	**Stato messaggio di log e gruppo prima della chiamata**		**Valori che il gestore code cerca**				
ORD LOG	**Grp Cur**	**Messaggio log Cur**	*MDMID*	*MDCID*	*MDGID*	*MDSEQ*	*MDOFF*
Sì	No	No	Controllato da *GMMO*	Controllato da *GMMO*	Controllato da *GMMO*	1	0
Sì	No	Sì	Qualsiasi identificativo di messaggio	Qualsiasi identificativo di correlazione	Identificativo gruppo precedente	1	Offset precedente + lunghezza segmento precedente
Sì	Sì	No	Qualsiasi identificativo di messaggio	Qualsiasi identificativo di correlazione	Identificativo gruppo precedente	Numero di sequenza precedente + 1	0
Sì	Sì	Sì	Qualsiasi identificativo di messaggio	Qualsiasi identificativo di correlazione	Identificativo gruppo precedente	Numero sequenza precedente	Offset precedente + lunghezza segmento precedente
No	Entrambi	Entrambi	Controllato da *GMMO*	Controllato da *GMMO*	Controllato da *GMMO*	Controllato da *GMMO*	Controllato da *GMMO*

Quando più gruppi di messaggi sono presenti nella coda e sono idonei per la restituzione, i gruppi vengono restituiti nell'ordine determinato dalla posizione nella coda del primo segmento del primo messaggio logico in ogni gruppo (ossia, i messaggi fisici che hanno numeri di sequenza di messaggi pari a 1 e gli offset pari a 0, determinano l'ordine in cui vengono restituiti i gruppi idonei).

L'opzione GMLOGO influisce sulle unità di lavoro come segue:

- Se il primo messaggio logico o segmento in un gruppo viene richiamato all'interno di un'unità di lavoro, tutti gli altri messaggi logici e segmenti nel gruppo devono essere richiamati all'interno di un'unità di lavoro, se viene utilizzato lo stesso handle di coda. Tuttavia, non è necessario recuperarli all'interno della stessa unità di lavoro. Ciò consente a un gruppo di messaggi costituito da molti messaggi fisici di essere suddiviso in due o più unità di lavoro consecutive per la gestione della coda.
- Se il primo messaggio logico o segmento in un gruppo non viene richiamato all'interno di un'unità di lavoro, nessuno degli altri messaggi logici e segmenti nel gruppo può essere richiamato all'interno di un'unità di lavoro, se viene utilizzato lo stesso handle di coda.

Se queste condizioni non vengono soddisfatte, la chiamata MQGET ha esito negativo con codice motivo RC2245 .

Quando viene specificato GMLOGO, l'MQGMO fornito sulla chiamata MQGET non deve essere inferiore a GMVER2e l'MQMD deve essere inferiore a MDVER2. Se questa condizione non viene soddisfatta, la chiamata ha esito negativo con il codice di errore RC2256 o RC2257 , come appropriato.

Se GMLOGO non viene specificato per chiamate MQGET successive per l'handle della coda, i messaggi vengono restituiti indipendentemente dal fatto che appartengano a gruppi di messaggi o che siano segmenti di messaggi logici. Ciò significa che i messaggi o i segmenti di un particolare gruppo o messaggio logico potrebbero essere restituiti fuori ordine oppure potrebbero essere mescolati con messaggi o segmenti di altri gruppi o messaggi logici o con messaggi che non sono in gruppi e non sono segmenti. In questa situazione, i particolari messaggi restituiti da successive chiamate MQGET sono controllati dalle opzioni MO* specificate su tali chiamate (consultare il campo *GMMO* descritto in "MQGMO (opzioni Get - message) su IBM i" a pagina 1105 per i dettagli di queste opzioni).

Questa è la tecnica che può essere utilizzata per riavviare un gruppo di messaggi o un messaggio logico nel mezzo, dopo che si è verificato un errore di sistema. Quando il sistema viene riavviato, l'applicazione può impostare i campi *MDGID*, *MDSEQ*, *MDOFF*e *GMMO* sui valori appropriati, quindi emettere la chiamata MQGET con GMSYP o GMNSYP impostati come necessario, ma senza specificare GMLOGO. Se questa chiamata ha esito positivo, il gestore code conserva le informazioni sul gruppo e sul segmento e le successive chiamate MQGET che utilizzano tale handle di coda possono specificare GMLOGO come normale.

Le informazioni sul gruppo e sul segmento che il gestore code conserva per la chiamata MQGET sono separate dalle informazioni sul gruppo e sul segmento che conserva per la chiamata MQPUT. Inoltre, il gestore code conserva informazioni separate per:

- Chiamate MQGET che rimuovono messaggi dalla coda.
- Chiamate MQGET che sfogliano i messaggi sulla coda.

Per qualsiasi handle di coda fornito, l'applicazione è libera di combinare chiamate MQGET che specificano GMLOGO con chiamate MQGET che non lo fanno, ma i seguenti punti devono essere annotati:

- Se GMLOGO non è specificato, ogni chiamata MQGET eseguita correttamente fa in modo che il gestore code imposti le informazioni sul gruppo salvato e sul segmento sui valori corrispondenti al messaggio restituito; questo sostituisce le informazioni sul segmento e sul gruppo esistenti conservate dal gestore code per l'handle della coda. Vengono modificate solo le informazioni appropriate all'azione della chiamata (sfoglia o rimuovi).
- Se GMLOGO non è specificato, la chiamata non ha esito negativo se è presente un gruppo di messaggi corrente o un messaggio logico; la chiamata, tuttavia, potrebbe avere esito positivo con un codice di completamento CCWARN. Tabella 703 a pagina 1120 mostra i vari casi che possono verificarsi. In questi casi, se il codice di completamento non è CCOK, il codice di errore è uno dei seguenti:
  - RC2241
  - RC2242
  - RC2245

  Nota: Il gestore code non controlla le informazioni sul gruppo e sul segmento quando si sfoglia una coda o quando si chiude una coda che è stata aperta per la ricerca ma non per l'immissione; in questi casi il codice di completamento è sempre CCOK (supponendo che non vi siano altri errori).

Tabella 703. Risultato quando la chiamata MQGET o MQCLOSE non è congruente con le informazioni sul gruppo e sul segmento

La chiamata corrente è	La chiamata precedente era MQGET con GMLOGO	La chiamata precedente era MQGET senza GMLOGO
MQGET con GMLOGO	CCNON RIUSCITO	CCNON RIUSCITO
MQGET senza GMLOGO	AVVCCN	CCOK

Tabella 703. Risultato quando la chiamata MQGET o MQCLOSE non è congruente con le informazioni sul gruppo e sul segmento (Continua)

La chiamata corrente è	La chiamata precedente era MQGET con GMLOGO	La chiamata precedente era MQGET senza GMLOGO
MQCLOSE con un gruppo non terminato o un messaggio logico	AVVCCN	CCOK

Le applicazioni che desiderano semplicemente richiamare i messaggi e i segmenti in ordine logico sono consigliate per specificare GMLOGO, poiché questa è l'opzione più semplice da utilizzare. Questa opzione allevia l'applicazione della necessità di gestire le informazioni sul gruppo e sul segmento, poiché il gestore code gestisce tali informazioni. Tuttavia, le applicazioni specializzate potrebbero richiedere un controllo maggiore di quello fornito dall'opzione GMLOGO e ciò può essere ottenuto non specificando tale opzione. In questo caso, l'applicazione deve verificare che i campi *MDMID*, *MDCID*, *MDGID*, *MDSEQ*e *MDOFF* in MQMD, e le opzioni MO* in *GMMO* in MQGMO, siano impostati correttamente, prima di ogni chiamata MQGET.

Ad esempio, un'applicazione che desidera inoltrare i messaggi fisici che riceve, indipendentemente dal fatto che tali messaggi si trovino in gruppi o segmenti di messaggi logici, non deve specificare GMLOGO. Ciò è dovuto al fatto che in una rete complessa con più percorsi tra i gestori code di invio e di ricezione, i messaggi fisici potrebbero arrivare fuori ordine. Non specificando GMLOGO e il PMLOGO corrispondente nella chiamata MQPUT, l'applicazione di inoltro può recuperare e inoltrare ogni messaggio fisico non appena arriva, senza dover attendere il successivo in ordine logico per arrivare.

GMLOGO può essere specificato con una qualsiasi delle altre opzioni GM* e con varie opzioni MO* in circostanze appropriate.

GMCMM
Sono richiamabili solo i messaggi logici completi.

Questa opzione specifica che solo un messaggio logico completo può essere restituito dalla chiamata MQGET. Se il messaggio logico è segmentato, il gestore code riassembla i segmenti e restituisce il messaggio logico completo all'applicazione; il fatto che il messaggio logico sia stato segmentato non è evidente per l'applicazione che lo richiama.

Nota: Questa è l'unica opzione che consente al gestore code di riassemblare i segmenti di messaggi. Se non specificato, i segmenti vengono restituiti singolarmente all'applicazione se sono presenti nella coda (e soddisfano gli altri criteri di selezione specificati nella chiamata MQGET). Le applicazioni che non desiderano ricevere singoli segmenti devono quindi sempre specificare GMCMPM.

Per utilizzare questa opzione, l'applicazione deve fornire un buffer sufficientemente grande da contenere il messaggio completo oppure specificare l'opzione GMATM.

Se la coda contiene messaggi segmentati con alcuni dei segmenti mancanti (forse perché sono stati ritardati nella rete e non sono ancora arrivati), specificando GMCMPM si impedisce il richiamo dei segmenti appartenenti a messaggi logici incompleti. Tuttavia, tali segmenti di messaggi contribuiscono ancora al valore dell'attributo della coda **CurrentQDepth** ; ciò significa che potrebbero non essere presenti messaggi logici richiamabili, anche se *CurrentQDepth* è maggiore di zero.

Per i messaggi permanenti, il gestore code può riassemblare i segmenti solo all'interno di un'unità di lavoro:

- Se la chiamata MQGET funziona all'interno di un'unità di lavoro definita dall'utente, viene utilizzata tale unità di lavoro. Se la chiamata non riesce durante il processo di riassemblaggio, il gestore code reinstalla sulla coda tutti i segmenti che sono stati rimossi durante il riassemblaggio. Tuttavia, l'errore non impedisce il corretto commit dell'unità di lavoro.

- Se la chiamata opera al di fuori di un'unità di lavoro definita dall'utente e non esiste alcuna unità di lavoro definita dall'utente, il gestore code crea un'unità di lavoro solo per la durata della chiamata. Se la chiamata ha esito positivo, il gestore code esegue automaticamente il commit dell'unità di lavoro (non è necessario che l'applicazione esegua questa operazione). Se la chiamata ha esito negativo, il gestore code esegue il backout dell'unità di lavoro.
- Se la chiamata opera all'esterno di un'unità di lavoro definita dall'utente, ma esiste un'unità di lavoro definita dall'utente, il gestore code non è in grado di eseguire il riassemblaggio. Se il messaggio non richiede il riassemblaggio, la chiamata può ancora avere esito positivo. Ma se il messaggio non richiede il riassemblaggio, la chiamata ha esito negativo con il codice di errore RC2255 .

Per i messaggi non persistenti, il gestore code non richiede che sia disponibile un'unità di lavoro per eseguire il riassemblaggio.

Ogni messaggio fisico che è un segmento ha il proprio descrittore di messaggi. Per i segmenti che costituiscono un singolo messaggio logico, la maggior parte dei campi nel descrittore del messaggio è la stessa per tutti i segmenti nel messaggio logico - in genere sono solo i campi *MDMID*, *MDOFF*e *MDMFL* che differiscono tra i segmenti nel messaggio logico. Tuttavia, se un segmento viene posizionato su una coda di messaggi non instradabili in un gestore code intermedio, il gestore DLQ richiama il messaggio specificando l'opzione GMCONV e ciò potrebbe causare la modifica della serie di caratteri o della codifica del segmento. Se il gestore DLQ invia correttamente il segmento, il segmento potrebbe avere una serie di caratteri o una codifica diversa dagli altri segmenti nel messaggio logico quando il segmento arriva al gestore code di destinazione.

Un messaggio logico composto da segmenti in cui i campi *MDCSI*, *MDENC*o entrambi differiscono non può essere riassemblato dal gestore code in un singolo messaggio logico. Invece, il gestore code riassembla e restituisce i primi segmenti consecutivi all'inizio del messaggio logico che hanno gli stessi identificativi e codifiche della serie di caratteri e la chiamata MQGET viene completata con il codice di completamento CCWARN e il codice motivo RC2243 o RC2244 , come appropriato. Ciò si verifica indipendentemente dal fatto che GMCONV sia specificato o meno. Per recuperare i restanti segmenti, l'applicazione deve emettere nuovamente la chiamata MQGET senza l'opzione GMCMPM, richiamandone uno per uno. GMLOGO può essere utilizzato per richiamare i rimanenti segmenti in ordine.

È inoltre possibile per un'applicazione che inserisce segmenti per impostare altri campi nel descrittore del messaggio su valori che differiscono tra i segmenti. Tuttavia, non vi è alcun vantaggio se l'applicazione ricevente utilizza GMCMPM per richiamare il messaggio logico. Quando il gestore code riassembla un messaggio logico, restituisce nel descrittore del messaggio i valori del descrittore del messaggio per il primo segmento; l'unica eccezione è il campo *MDMFL* , che il gestore code imposta per indicare che il messaggio riassemblato è l'unico segmento.

Se GMCMPM viene specificato per un messaggio di report, il gestore code esegue un'elaborazione speciale. Il gestore code controlla la coda per vedere se tutti i messaggi di report di quel tipo relativi ai diversi segmenti nel messaggio logico sono presenti nella coda. Se lo sono, possono essere richiamati come un singolo messaggio specificando GMCMPM. Affinché ciò sia possibile, i messaggi di report devono essere generati da un gestore code o da un MCA che supporta la segmentazione oppure l'applicazione di origine deve richiedere almeno 100 byte di dati del messaggio (ovvero, devono essere specificate le opzioni RO* D o RO* F appropriate). Se per un segmento è presente meno dell'intera quantità di dati dell'applicazione, i byte mancanti vengono sostituiti da valori null nel messaggio di report restituito.

Se GMCMPM è specificato con GMMUC o GMBRWC, il cursore di ricerca deve essere posizionato su un messaggio con un campo *MDOFF* in MQMD che ha un valore pari a 0. Se questa condizione non viene soddisfatta, la chiamata ha esito negativo con codice di errore RC2246 .

GMCMPM implica GMASGA, che non è pertanto necessario specificare.

GMCMPM può essere specificato con una qualsiasi delle altre opzioni GM* ad eccezione di GMPSYP e con una qualsiasi delle opzioni MO* ad eccezione di MOOFFS.

GMAMSA
Tutti i messaggi nel gruppo devono essere disponibili.

Questa opzione specifica che i messaggi in un gruppo diventano disponibili per il richiamo solo quando tutti i messaggi nel gruppo sono disponibili. Se la coda contiene gruppi di messaggi con alcuni dei messaggi mancanti (probabilmente perché sono stati ritardati nella rete e non sono ancora arrivati), la specifica di GMAMSA impedisce il richiamo dei messaggi appartenenti a gruppi incompleti. Tuttavia, tali messaggi contribuiscono ancora al valore dell'attributo della coda **CurrentQDepth** ; ciò significa che potrebbero non essere presenti gruppi di messaggi richiamabili, anche se **CurrentQDepth** è maggiore di zero. Se non sono presenti altri messaggi richiamabili, il codice di errore RC2033 viene restituito dopo la scadenza dell'intervallo di attesa specificato (se presente).

L'elaborazione di GMAMSA dipende dal fatto che GMLOGO sia specificato o meno:

- Se vengono specificate entrambe le opzioni, la GMAMSA influisce solo quando non è presente alcun gruppo corrente o messaggio logico. Se è presente un gruppo corrente o un messaggio logico, GMAMSA viene ignorato. Ciò significa che GMAMSA può rimanere attivo quando si elaborano i messaggi in ordine logico.
- Se GMAMSA viene specificato senza GMLOGO, GMAMSA ha sempre effetto. Ciò significa che l'opzione deve essere disattivata dopo che il primo messaggio nel gruppo è stato rimosso dalla coda, per poter rimuovere i restanti messaggi nel gruppo.

Il corretto completamento di una chiamata MQGET che specifica GMAMSA significa che al momento in cui è stata emessa la chiamata MQGET, tutti i messaggi nel gruppo erano nella coda. Tuttavia, tenere presente che altre applicazioni sono ancora in grado di rimuovere i messaggi dal gruppo (il gruppo non è bloccato all'applicazione che richiama il primo messaggio nel gruppo).

Se questa opzione non viene specificata, i messaggi appartenenti ai gruppi possono essere richiamati anche quando il gruppo è incompleto.

GMAMSA implica GMASGA, che non è pertanto necessario specificare.

GMAMSA può essere specificato con una qualsiasi delle altre opzioni GM* e con una qualsiasi delle opzioni MO*.

GMASGA
Tutti i segmenti in un messaggio logico devono essere disponibili.

Questa opzione indica che i segmenti in un messaggio logico diventano disponibili per il recupero solo quando sono disponibili tutti i segmenti nel messaggio logico. Se la coda contiene messaggi segmentati con alcuni dei segmenti mancanti (forse perché sono stati ritardati nella rete e non sono ancora arrivati), specificando GMASGA si impedisce il richiamo dei segmenti appartenenti a messaggi logici incompleti. Tuttavia, tali segmenti contribuiscono ancora al valore dell'attributo della coda **CurrentQDepth** ; ciò significa che potrebbe non essere presente alcun messaggio logico richiamabile, anche se **CurrentQDepth** è maggiore di zero. Se non sono presenti altri messaggi richiamabili, il codice di errore RC2033 viene restituito dopo la scadenza dell'intervallo di attesa specificato (se presente).

L'elaborazione di GMASGA dipende dal fatto che GMLOGO sia specificato o meno:

- Se vengono specificate entrambe le opzioni, GMASGA ha effetto solo quando non è presente alcun messaggio logico corrente. Se è presente un messaggio logico corrente, GMASGA viene ignorato. Ciò significa che GMASGA può rimanere attivo durante l'elaborazione dei messaggi in ordine logico.
- Se GMASGA è specificato senza GMLOGO, GMASGA ha sempre un effetto. Ciò significa che l'opzione deve essere disattivata dopo che il primo segmento nel messaggio logico è stato rimosso dalla coda, in modo da poter rimuovere i restanti segmenti nel messaggio logico.

Se questa opzione non viene specificata, i segmenti del messaggio possono essere richiamati anche quando il messaggio logico è incompleto.

Mentre sia GMCMPM che GMASGA richiedono che tutti i segmenti siano disponibili prima di poter essere richiamati, il primo restituisce il messaggio completo, mentre il secondo consente di richiamare i segmenti uno per uno.

Se GMASGA viene specificato per un messaggio di report, il gestore code esegue un'elaborazione speciale. Il gestore code controlla la coda per verificare se è presente almeno un messaggio di report per ciascuno dei segmenti che costituiscono il messaggio logico completo. In tal caso, viene soddisfatta la condizione GMASGA. Tuttavia, il gestore code non controlla il tipo di messaggi di report presenti, pertanto potrebbe essere presente una combinazione di tipi di report nei messaggi di report relativi ai segmenti del messaggio logico. Di conseguenza, il successo di GMASGA non implica il successo di GMCMPM. Se esiste una combinazione di tipi di report presenti per i segmenti di un particolare messaggio logico, tali messaggi di report devono essere richiamati uno alla volta.

GMASGA può essere specificato con una qualsiasi delle altre opzioni GM* e con una qualsiasi delle opzioni MO*.

Opzione predefinita: se non è richiesta alcuna delle opzioni descritte, è possibile utilizzare la seguente opzione:

GMNONE
Nessuna opzione specificata.

Questo valore può essere utilizzato per indicare che non sono state specificate altre opzioni; tutte le opzioni assumono i valori predefiniti. GMNONE è definito per aiutare la documentazione del programma; non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Il valore iniziale del campo *GMOPT* è GMNWT.

GMRE1 (stringa di caratteri a 1 byte)

Riservato.

Questo è un campo riservato. Il valore iniziale di questo campo è un carattere vuoto. Questo campo viene ignorato se *GMVER* è minore di GMVER2.

GMRL (numero intero con segno a 10 cifre)

Lunghezza dei dati del messaggio restituiti (byte).

Si tratta di un campo di output impostato dal gestore code sulla lunghezza in byte dei dati del messaggio restituiti dalla chiamata MQGET nel parametro **BUFFER** . Se il gestore code non supporta questa funzione, *GMRL* viene impostato sul valore RLUNDF.

Quando i messaggi vengono convertiti tra codifiche o serie di caratteri, i dati del messaggio a volte possono modificare la dimensione. Al ritorno dalla chiamata MQGET:

- Se *GMRL* non è RLUNDF, il numero di byte dei dati del messaggio restituiti viene fornito da *GMRL*.
- Se *GMRL* ha il valore RLUNDF, il numero di byte dei dati del messaggio restituiti è generalmente fornito dal valore più piccolo di *BUFLEN* e *DATLEN*, ma può essere inferiore a questo se la chiamata MQGET viene completata con il codice motivo RC2079 . In questo caso, i byte non significativi nel parametro **BUFFER** vengono impostati su valori null.

Viene definito il seguente valore speciale:

RLUNDF
Lunghezza dei dati restituiti non definita.

Il valore iniziale di questo campo è RLUNDF. Questo campo viene ignorato se *GMVER* è inferiore a GMVER3.

GMRQN (stringa di caratteri a 48 byte)

Nome risolto della coda di destinazione.

Questo è un campo di output impostato dal gestore code sul nome locale della coda da cui è stato richiamato il messaggio, come definito nel gestore code locale. Questo è diverso dal nome utilizzato per aprire la coda se:

- È stata aperta una coda alias (in tal caso, viene restituito il nome della coda locale a cui è stato risolto l'alias) oppure
- È stata aperta una coda modello (in tal caso, viene restituito il nome della coda locale dinamica).

La lunghezza di questo campo è fornita da LNQN. Il valore iniziale di questo campo è di 48 caratteri vuoti.

GMRS2 (stringa di caratteri a 1 byte)

Riservato.

Questo è un campo riservato. Il valore iniziale di questo campo è un carattere vuoto. Questo campo viene ignorato se *GMVER* è inferiore a GMVER4.

GMSEG (stringa di caratteri a 1 byte)

Indicatore che indica se è consentita un'ulteriore segmentazione per il messaggio richiamato.

Ha uno dei seguenti valori:

SEGIHB
Segmentazione non consentita.

SEGALW
Segmentazione consentita.

Questo è un campo di output. Il valore iniziale di questo campo è SEGIHB. Questo campo viene ignorato se *GMVER* è minore di GMVER2.

GMSG1 (numero intero con segno a 10 cifre)

Segnale.

Questo è un campo riservato; il suo valore non è significativo. Il valore iniziale di questo campo è 0.

GMSG2 (numero intero con segno a 10 cifre)

Identificativo del segnale.

Questo è un campo riservato; il suo valore non è significativo.

GMSID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

GMSIDV
Identificativo per la struttura delle opzioni get - message.

Questo campo è sempre un campo di input. Il valore iniziale di questo campo è GMSIDV.

GMSST (stringa di caratteri a 1 byte)

Indicatore che specifica se il messaggio richiamato è un segmento di un messaggio logico.

Ha uno dei seguenti valori:

SSNSEG
Il messaggio non è un segmento.

SSEG
Il messaggio è un segmento, ma non è l'ultimo segmento del messaggio logico.

SSLSEG
Il messaggio è l'ultimo segmento del messaggio logico.

Questo è anche il valore restituito se il messaggio logico è composto da un solo segmento.

Questo campo è un campo di output. Il valore iniziale di questo campo è SSNSEG. Questo campo viene ignorato se *GMVER* è minore di GMVER2.

GMTOK (stringa di bit a 16 byte)

Token messaggio.

Questo è un campo riservato; il suo valore non è significativo. Viene definito il seguente valore speciale:

NON MTK
Nessun token del messaggio.

Il valore è zero binario per la lunghezza del campo.

La lunghezza di questo campo è fornita da LNMTOK. Il valore iniziale di questo campo è MTKNON. Questo campo viene ignorato se *GMVER* è inferiore a GMVER3.

GMVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere uno dei seguenti.

GMVER1
Struttura delle opzioni Version-1 get - message.

GMVER2
Struttura delle opzioni Version-2 get - message.

GMVER3
Version-3 struttura delle opzioni get - message.

GMVER4
Struttura delle opzioni Version-4 get - message.

I campi che esistono solo nelle versioni più recenti della struttura vengono identificati come tali nelle descrizioni dei campi. La seguente costante specifica il numero di versione della versione corrente:

GMVERC
Versione corrente della struttura delle opzioni get - message.

Questo campo è sempre un campo di input. Il valore iniziale di questo campo è GMVER1.

GMVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere uno dei seguenti.

GMVER1
Struttura delle opzioni Version-1 get - message.

GMVER2
Struttura delle opzioni Version-2 get - message.

GMVER3
Version-3 struttura delle opzioni get - message.

GMVER4
Struttura delle opzioni Version-4 get - message.

I campi che esistono solo nelle versioni più recenti della struttura vengono identificati come tali nelle descrizioni dei campi. La seguente costante specifica il numero di versione della versione corrente:

GMVERC
Versione corrente della struttura delle opzioni get - message.

Questo campo è sempre un campo di input. Il valore iniziale di questo campo è GMVER1.

GMWI (numero intero con segno a 10 cifre)

Intervallo di attesa.

Questo è il tempo approssimativo, espresso in millisecondi, in cui la chiamata MQGET attende l'arrivo di un messaggio adatto (ovvero, un messaggio che soddisfa i criteri di selezione specificati nel parametro **MSGDSC** della chiamata MQGET; per ulteriori dettagli, consultare il campo *MDMID* descritto in "MQMD (Message Descriptor) su IBM i" a pagina 1140). Se non è arrivato alcun messaggio adatto dopo questo periodo di tempo, la chiamata viene completata con CCFAIL e codice motivo RC2033.

GMWI viene utilizzato con l'opzione GMWT. Viene ignorato se questa opzione non viene specificata. Se viene specificato, *GMWI* deve essere maggiore o uguale a zero o il seguente valore speciale:

WIULIM
Intervallo di attesa illimitato.

Il valore iniziale di questo campo è 0.

Valori iniziali

Tabella 704. Campi in MQGMO

Nome campo	Nome della costante	Valore della costante
GMSID	GMSIDV	'GMO¬'
GMVER	GMVER1	1
GMOPT	GMNWT	0
GMWI	Nessuna	0
GMSG1	Nessuna	0
GMSG2	Nessuna	0
GMRQN	Nessuna	Spazi
GMMO	MOMSGI + MOCORI	3
GMGST	GSNIG	''
GMSST	SSNSEG	''
GMSEG	SEGIHB	''
GMRE1	Nessuna	''
GMTOK	NON MTK	Valori null
GMRL	RLUNDF	-1
GMRS2	Nessuna	''
GMMH	HMNONE	0

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQGMO Structure
D*
D* Structure identifier
D  GMSID                  1      4    INZ('GMO ')
D* Structure version number
D  GMVER                  5      8I 0 INZ(1)
D* Options that control the action ofMQGET
D  GMOPT                  9     12I 0 INZ(0)
D* Wait interval
```

```
     D  GMWI                  13     16I 0 INZ(0)
     D* Signal
     D  GMSG1                 17     20I 0 INZ(0)
     D* Signal identifier
     D  GMSG2                 21     24I 0 INZ(0)
     D* Resolved name of destination queue
     D  GMRQN                 25     72    INZ
     D* Options controlling selection criteriaused for MQGET
     D  GMMO                  73     76I 0 INZ(3)
     D* Flag indicating whether messageretrieved is in a group
     D  GMGST                 77     77    INZ(' ')
     D* Flag indicating whether messageretrieved is a segment of a
     D* logicalmessage
     D  GMSST                 78     78    INZ(' ')
     D* Flag indicating whether furthersegmentation is allowed for themessage
     D* retrieved
     D  GMSEG                 79     79    INZ(' ')
     D* Reserved
     D  GMRE1                 80     80    INZ
     D* Message token
     D  GMTOK                 81     96    INZ(X'00000000000000-
     D                                     000000000000000000')
     D* Length of message data returned(bytes)
     D  GMRL                  97    100I 0 INZ(-1)
     D* Reserved
     D  GMRS2                101    104I 0 INZ(0)
     D* Message handle
     D  GMMH                 105    112I 0 INZ(0)
```

IBM i MQIIH (IMS information header) su IBM i

La struttura MQIIH descrive le informazioni che devono essere presenti all'inizio di un messaggio inviato al bridge IMS tramite IBM MQ for z/OS.

Panoramica

Nome formato: FMIMS.

Serie di caratteri e codifica: le condizioni speciali si applicano alla serie di caratteri e alla codifica utilizzati per la struttura MQIIH e i dati del messaggio dell'applicazione:

- Le applicazioni che si connettono al gestore code proprietario della coda bridge IMS devono fornire una struttura MQIIH che si trova nella serie di caratteri e nella codifica del gestore code. Ciò è dovuto al fatto che la conversione dei dati della struttura MQIIH non viene eseguita in questo caso.
- Le applicazioni che si collegano ad altri gestori code possono fornire una struttura MQIIH che si trova in una delle serie di caratteri e codifiche supportate; la conversione di MQIIH viene eseguita dall'agent del canale dei messaggi di ricezione connesso al gestore code che possiede la coda bridge IMS .

  Nota: C'è un'eccezione a questo. Se il gestore code proprietario della coda bridge IMS sta utilizzando CICS per l'accodamento distribuito, MQIIH deve essere nella serie di caratteri e nella codifica del gestore code proprietario della coda bridge IMS .
- I dati del messaggio dell'applicazione che seguono la struttura MQIIH devono essere nella stessa serie di caratteri e nella stessa codifica della struttura MQIIH. I campi *IICSI* e *IIENC* nella struttura MQIIH non possono essere usati per specificare la serie di caratteri e la codifica dei dati del messaggio dell'applicazione.

  Un'uscita di conversione dati deve essere fornita dall'utente per convertire i dati del messaggio dell'applicazione se i dati non sono uno dei formati integrati supportati dal gestore code.

- “Autenticazione di passtickets per applicazioni bridge IMS” a pagina 1129
- “Campi” a pagina 1129
- “Valori iniziali” a pagina 1132
- “Dichiarazione RPG” a pagina 1132

Autenticazione di passtickets per applicazioni bridge IMS

Ora è possibile per gli amministratori di IBM MQ specificare il nome dell'applicazione da utilizzare per l'autenticazione dei passticket, per le applicazioni bridge IMS . A tale scopo, il nome dell'applicazione viene specificato come un nuovo attributo PTKTAPPL per la definizione dell'oggetto STGCLASS, come una stringa alfanumerica da 1 a 8 caratteri.

Un valore vuoto indica che l'autenticazione si verifica come con le release precedenti di IBM MQ, ovvero, non viene immesso alcun nome applicazione nella richiesta di autenticazione e viene utilizzato il valore MVSxxxx.

Un valore di 1-8 caratteri alfanumerici deve seguire le regole per i nomi delle applicazioni passticket come descritto nelle pubblicazioni RACF .

Gli amministratori IBM MQ e RACF devono concordare i nomi di applicazione validi da utilizzare. L'amministratore RACF deve creare un profilo nella classe PTKTDATA che fornisce l'accesso READ agli ID utente di tutte le applicazioni a cui deve essere concesso l'accesso. L'amministratore di IBM MQ deve creare o modificare le definizioni STGCLASS richieste che specificano il nome dell'applicazione da utilizzare per l'autenticazione passticket.

Per informazioni correlate, consultare il manuale *Script (MQSC) Command Reference*.

Campi

La struttura MQIIH contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

IIAUT (stringa di caratteri a 8 byte)

RACF password o passticket.

Questo è facoltativo; se specificato, viene utilizzato con l'ID utente nel contesto di sicurezza MQMD per creare un UTOKEN inviato a IMS per fornire un contesto di sicurezza. Se non viene specificato, l'ID utente viene utilizzato senza verifica. Ciò dipende dall'impostazione degli switch RACF , che possono richiedere la presenza di un programma di autenticazione.

Viene ignorato se il primo byte è vuoto o null. È possibile utilizzare il seguente valore speciale:

IUNON
Nessuna autenticazione.

La lunghezza di questo campo è fornita da LNAUTH. Il valore iniziale di questo campo è IAUNON.

IICMT (stringa di caratteri a 1 byte)

Modalità di commit.

Consultare il manuale *OTMA Reference* per ulteriori informazioni sulle modalità di commit IMS . Il valore deve essere uno dei seguenti.

ICMCTS
Commit e invio.

Questa modalità implica una doppia accodamento dell'output, ma tempi di occupazione della regione più brevi. Le transazioni interattive e Fast - path non possono essere eseguite con questa modalità.

ICMSTC
Inviare quindi eseguire il commit.

Il valore iniziale di questo campo è ICMCTS.

IICSI (numero intero con segno a 10 cifre)

Riservato.

Questo è un campo riservato; il suo valore non è significativo. Il valore iniziale di questo campo è 0.

IIENC (numero intero con segno a 10 cifre)

Riservato.

Questo è un campo riservato; il suo valore non è significativo. Il valore iniziale di questo campo è 0.

IIFLG (numero intero con segno a 10 cifre)

Indicatori.

Il valore deve essere:

IINONE
Nessun indicatore.

Il valore iniziale di questo campo è IINONE.

IIFMT (stringa di caratteri a 8 byte)

Nome formato IBM MQ dei dati che seguono MQIIH.

Specifica il nome del formato IBM MQ dei dati che seguono la struttura MQIIH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono le stesse del campo *MDFMT* in MQMD.

La lunghezza di questo campo è fornita da LNFMT. Il valore iniziale di questo campo è FMNONE.

IILEN (numero intero con segno a 10 cifre)

Lunghezza della struttura MQIIH.

Il valore deve essere:

IILEN1
Lunghezza della struttura dell'intestazione delle informazioni IMS .

Il valore iniziale di questo campo è IILEN1.

IILTO (stringa di caratteri a 8 byte)

Sovrascrittura terminale logico.

Questo è posizionato nel campo IO PCB. È facoltativo; se non viene specificato, viene utilizzato il nome TPIPE. Viene ignorato se il primo byte è vuoto o null.

La lunghezza di questo campo è fornita da LNLTOV. Il valore iniziale di questo campo è di 8 caratteri vuoti.

IIMMN (stringa di caratteri a 8 byte)

Nome della mappa dei servizi di formato del messaggio.

Questo è posizionato nel campo IO PCB. È facoltativo. In input rappresenta il MID, in output rappresenta il MOD. Viene ignorato se il primo byte è vuoto o null.

La lunghezza di questo campo è data da LNMFMN. Il valore iniziale di questo campo è di 8 caratteri vuoti.

IIRFM (stringa di caratteri a 8 byte)

IBM MQ nome formato del messaggio di risposta.

Questo è il nome formato IBM MQ del messaggio di replica che verrà inviato in risposta al messaggio corrente. Le regole per la codifica sono le stesse del campo *MDFMT* in MQMD.

La lunghezza di questo campo è fornita da LNFMT. Il valore iniziale di questo campo è FMNONE.

IIRSV (stringa di caratteri a 1 byte)

Riservato.

Questo è un campo riservato; deve essere vuoto.

IISEC (stringa di caratteri a 1 byte)

Ambito di sicurezza.

Ciò indica l'elaborazione della sicurezza IMS richiesta. Vengono definiti i seguenti valori:

ISSCHK
Controllare l'ambito di sicurezza.

Un ACEE viene creato nella regione di controllo, ma non nella regione dipendente.

ISSFUL
Ambito di sicurezza completo.

Un ACEE memorizzato nella cache viene creato nella control region e un ACEE non memorizzato nella cache viene creato nella regione dipendente. Se si utilizza ISSFUL, è necessario assicurarsi che l'ID utente per cui viene creato l'ACEE abbia accesso alle risorse utilizzate nella regione dipendente.

Se ISSCHK e ISSFUL non vengono specificati per questo campo, viene assunto ISSCHK.

Il valore iniziale di questo campo è ISSCHK.

IISID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

IIDISV
Identificativo per la struttura dell'intestazione delle informazioni IMS .

Il valore iniziale di questo campo è IISIDV.

IITID (stringa bit a 16 byte)

Identificativo dell'istanza della transazione.

Questo campo viene utilizzato dai messaggi di output di IMS , quindi viene ignorato al primo input. Se *IITST* è impostato su ITSIC, questo deve essere fornito nell'input successivo e in tutti gli input successivi, per consentire a IMS di correlare i messaggi alla conversazione corretta. È possibile utilizzare il seguente valore speciale:

INATTIVO
Nessun ID istanza transazione.

La lunghezza di questo campo è fornita da LNTIID. Il valore iniziale di questo campo è ITINON.

IITST (stringa di caratteri a 1 byte)

Stato della transazione.

Indica lo stato della conversazione IMS . Questo viene ignorato al primo input perché non esiste alcuna conversazione. Negli input successivi indica se una conversazione è attiva o meno. All'output viene impostato da IMS. Il valore deve essere uno dei seguenti.

ITSIC
Nella conversazione.

ITSNIC
Non in conversazione.

ITSARC
Restituire i dati di stato della transazione in formato architettato.

Questo valore viene utilizzato solo con il comando IMS `/DISPLAY TRAN` . Ciò fa sì che i dati di stato della transazione vengano restituiti nel formato progettato IMS invece che nel formato carattere. Per ulteriori dettagli, consultare Scrittura IMS dei programmi di transazione tramite IBM MQ .

Il valore iniziale di questo campo è ITSNIC.

IIVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere:

IIVER1
Numero di versione per la struttura dell'intestazione delle informazioni IMS .

La seguente costante specifica il numero di versione della versione corrente:

IIVERC
Versione corrente della struttura dell'intestazione delle informazioni IMS .

Il valore iniziale di questo campo è IIVER1.

Valori iniziali

Tabella 705. Campi in MQIIH

Nome campo	Nome della costante	Valore della costante
IISID	IIDISV	'IIH¬'
IIVER	IIVER1	1
IILEN	IILEN1	84
IIENC	Nessuna	0
IICSI	Nessuna	0
IIFMT	FMNONE	Spazi
IIFLG	IINONE	0
IILTO	Nessuna	Spazi
IIMMN	Nessuna	Spazi
IIRFM	FMNONE	Spazi
IIAUT	IUNON	Spazi
IITID	INATTIVO	Valori null
IITST	ITSNIC	''
IICMT	ICMCTS	'0'
IISEC	ISSCHK	'C'
IIRSV	Nessuna	''

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
     D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     D*
     D* MQIIH Structure
     D*
     D* Structure identifier
     D  IISID                  1      4    INZ('IIH ')
     D* Structure version number
     D  IIVER                  5      8I 0 INZ(1)
     D* Length of MQIIH structure
     D  IILEN                  9     12I 0 INZ(84)
```

```
     D* Reserved
     D  IIENC                 13     16I 0 INZ(0)
     D* Reserved
     D  IICSI                 17     20I 0 INZ(0)
     D* MQ format name of data that followsMQIIH
     D  IIFMT                 21     28    INZ('        ')
     D* Flags
     D  IIFLG                 29     32I 0 INZ(0)
     D* Logical terminal override
     D  IILTO                 33     40    INZ
     D* Message format services map name
     D  IIMMN                 41     48    INZ
     D* MQ format name of reply message
     D  IIRFM                 49     56    INZ('        ')
     D* RACF password or passticket
     D  IIAUT                 57     64    INZ('        ')
     D* Transaction instance identifier
     D  IITID                 65     80    INZ(X'00000000000000-
     D                                     000000000000000000')
     D* Transaction state
     D  IITST                 81     81    INZ(' ')
     D* Commit mode
     D  IICMT                 82     82    INZ('0')
     D* Security scope
     D  IISEC                 83     83    INZ('C')
     D* Reserved
     D  IIRSV                 84     84    INZ
```

IBM i MQIMPO (Interroga opzioni proprietà messaggio) su IBM i

La struttura MQIMPO consente alle applicazioni di specificare le opzioni che controllano la modalità di interrogazione delle proprietà dei messaggi.

Panoramica

Scopo: La struttura è un parametro di immissione sulla chiamata MQINQMP.

Serie di caratteri e codifica: i dati in MQIMPO devono essere nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (ENNAT).

- "Campi" a pagina 1133
- "Valori iniziali" a pagina 1139
- "Dichiarazione RPG" a pagina 1139

Campi

La struttura MQIMPO contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

IPOPT (numero intero con segno a 10 cifre)

Le opzioni riportate di seguito controllano l'operazione di MQINQMP. È possibile specificare una o più di queste opzioni. Per specificare più di un'opzione, aggiungere i valori insieme (non aggiungere la stessa costante più di una volta) o combinare i valori utilizzando l'operazione OR bit per bit (se il linguaggio di programmazione supporta le operazioni bit). Sono indicate le combinazioni di opzioni non valide; sono valide tutte le altre combinazioni.

Opzioni dati valore: le seguenti opzioni si riferiscono all'elaborazione dei dati valore quando la proprietà viene richiamata dal messaggio.

IPCVAL
Questa opzione richiede che il valore della proprietà sia convertito in modo da essere conforme ai valori di *IPREQCSI* e *IPREQENC* specificati prima della chiamata MQINQMP restituisce il valore della proprietà nell'area *Value* .

- Se la conversione ha esito positivo, i campi *IPRETCSI* e *IPRETENC* vengono impostati sullo stesso valore di *IPREQCSI* e *IPREQENC* al ritorno dalla chiamata MQINQMP.

- Se la conversione non riesce, ma la chiamata MQINQMP viene altrimenti completata senza errori, il valore della proprietà viene restituito non convertito.

  Se la proprietà è una stringa, i campi *IPRETCSI* e *IPRETENC* sono impostati sulla serie di caratteri e sulla codifica della stringa non convertita.

  Il codice di completamento è CCWARN in questo caso, con codice di errore RC2466. Il cursore della proprietà è avanzato rispetto alla proprietà restituita.

Se il valore della proprietà si espande durante la conversione e supera la dimensione del parametro **Value** , il valore viene restituito non convertito, con codice di completamento CCFAIL; il codice motivo è impostato su RC2469.

Il parametro **DataLength** della chiamata MQINQMP restituisce la lunghezza in cui il valore della proprietà sarebbe stato convertito, per consentire all'applicazione di determinare la dimensione del buffer richiesta per contenere il valore della proprietà convertita. Il cursore della proprietà non viene modificato.

Questa opzione richiede inoltre che:

- Se il nome della proprietà contiene un carattere jolly e
- Il campo *IPRETNAMECHRP* viene inizializzato con un indirizzo o uno scostamento per il nome restituito,

il nome restituito viene convertito per essere conforme ai valori *IPREQCSI* e *IPREQENC* .

- Se la conversione ha esito positivo, il campo *VSCCSID* di *IPRETNAMECHRP* e la codifica del nome restituito vengono impostati sul valore di input di *IPREQCSI* e *IPREQENC*.
- Se la conversione non riesce, ma la chiamata MQINQMP viene altrimenti completata senza errori o avvertenze, il nome restituito non viene convertito. Il codice di completamento è CCWARN in questo caso, con codice di errore RC2492.

  Il cursore della proprietà è avanzato rispetto alla proprietà restituita. RC2466 viene restituito se il valore e il nome non vengono convertiti.

Se il nome restituito si espande durante la conversione e supera la dimensione del campo *VSBufsize* di *RequestedName*, la stringa restituita rimane non convertita, con codice di completamento CCFAIL e il codice motivo è impostato su RC2465.

Il campo *VSLength* della struttura MQCHARV restituisce la lunghezza in cui il valore della proprietà sarebbe stato convertito, in modo da consentire all'applicazione di determinare la dimensione del buffer richiesto per contenere il valore della proprietà convertita. Il cursore della proprietà non viene modificato.

IPCTYP

Questa opzione richiede che il valore della proprietà venga convertito dal tipo di dati corrente, nel tipo di dati specificato nel parametro **Type** della chiamata MQINQMP.

- Se la conversione riesce, il parametro **Type** non viene modificato al ritorno della chiamata MQINQMP.
- Se la conversione ha esito negativo, ma la chiamata MQINQMP viene altrimenti completata senza errori, la chiamata ha esito negativo con il motivo RC2470. Il cursore della proprietà non viene modificato.

Se la conversione del tipo di dati causa l'espansione del valore durante la conversione e il valore convertito supera la dimensione del parametro **Value** , il valore viene restituito non convertito, con codice di completamento CCFAIL e il codice di errore è impostato su RC2469.

Il parametro **DataLength** della chiamata MQINQMP restituisce la lunghezza in cui il valore della proprietà sarebbe stato convertito, per consentire all'applicazione di determinare la dimensione del buffer richiesta per contenere il valore della proprietà convertita. Il cursore della proprietà non viene modificato.

Se il valore del parametro **Type** della chiamata MQINQMP non è valido, la chiamata ha esito negativo con motivo RC2473.

Se la conversione del tipo di dati richiesto non è supportata, la chiamata ha esito negativo con motivo RC2470. Sono supportate le seguenti conversioni del tipo di dati:

Tabella 706. Conversioni di tipi di dati supportate

Tipo dati proprietà	Tipi di dati di destinazione supportati
TIPOnota di carico	TYPSTR, TYPI8, TYPI16, TYPI32, TYPI64
TYPBST	TIPOSTR
TYPI8	TYPSTR, TYPI16, TYPI32, TYPI64
TYPI16	TYPSTR, TYPI32, TYPI64
TYPI32	TYPSTR, TYPI64
TYPI64	TIPOSTR
TYPF32	TYPSTR, TYPF64
TYPF64	TIPOSTR
TIPOSTR	TYPBOL, TYPI8, TYPI16, TYPI32, TYPI64, TYPF32, TYPF64
TYPNUL	Nessuna

Le regole generali che disciplinano le conversioni supportate sono le seguenti:

- I valori delle proprietà numeriche possono essere convertiti da un tipo di dati ad un altro, purché non si perda alcun dato durante la conversione.

  Ad esempio, il valore di una proprietà con tipo di dati TYPI32 può essere convertito in un valore con tipo di dati TYPI64, ma non può essere convertito in un valore con tipo di dati TYPI16.
- Un valore di proprietà di qualsiasi tipo di dati può essere convertito in una stringa.
- Un valore della proprietà stringa può essere convertito in qualsiasi altro tipo di dati, a condizione che la stringa sia formattata correttamente per la conversione. Se un'applicazione tenta di convertire un valore della proprietà stringa non formattato correttamente, IBM MQ restituisce il codice motivo RC2472.
- Se un'applicazione tenta una conversione non supportata, IBM MQ restituisce il codice di errore RC2470.

Le regole specifiche per convertire un valore di proprietà da un tipo di dati ad un altro sono le seguenti:

- Quando si converte un valore della proprietà TYPBOL in una stringa, il valore TRUE viene convertito nella stringa "TRUE" e il valore false viene convertito nella stringa "FALSE".
- Quando si converte un valore della proprietà TYPBOL in un tipo di dati numerico, il valore TRUE viene convertito in uno e il valore FALSE viene convertito in zero.
- Quando si converte un valore della proprietà stringa in un valore TYPBOL, la stringa "TRUE", o "1", viene convertita in TRUE e la stringa "FALSE", o "0", viene convertita in FALSE.

  Notare che i termini "TRUE" e "FALSE" non sono sensibili al maiuscolo / minuscolo.

  Qualsiasi altra stringa non può essere convertita; IBM MQ restituisce il codice motivo RC2472.
- Quando si converte un valore della proprietà stringa in un valore con tipo di dati TYPI8, TYPI16, TYPI32 o TYPI64, la stringa deve avere il seguente formato:

```
[blanks][sign]digits
```

I significati dei componenti della stringa sono i seguenti:

blanks
Caratteri vuoti iniziali facoltativi

sign
Un segno più (+) o segno meno (-) facoltativo.

digits
Una sequenza contigua di caratteri cifra (0-9). Deve essere presente almeno un carattere cifra.

Dopo la sequenza di caratteri cifra, la stringa può contenere altri caratteri che non sono caratteri cifra, ma la conversione si interrompe non appena viene raggiunto il primo di questi caratteri. Si presuppone che la stringa rappresenti un numero intero decimale.

IBM MQ restituisce il codice di errore RC2472 se la stringa non è formattata correttamente.

- Quando si converte un valore della proprietà stringa in un valore con tipo di dati TYPF32 o TYPF64, la stringa deve avere il formato seguente:

```
[blanks][sign]digits[.digits][e_char[e_sign]e_digits]
```

I significati dei componenti della stringa sono i seguenti:

blanks
Caratteri vuoti iniziali facoltativi

sign
Un segno più (+) o segno meno (-) facoltativo.

digits
Una sequenza contigua di caratteri cifra (0-9). Deve essere presente almeno un carattere cifra.

e_char
Un carattere esponente, che è "E" o "e".

e_sign
Un segno più (+) o un segno meno (-) facoltativo per l'esponente.

e_digits
Una sequenza contigua di caratteri cifra (0-9) per l'esponente. Deve essere presente almeno un carattere cifra se la stringa contiene un carattere esponente.

Dopo la sequenza di caratteri cifra, o i caratteri facoltativi che rappresentano un esponente, la stringa può contenere altri caratteri che non sono caratteri cifra, ma la conversione si interrompe non appena viene raggiunto il primo di questi caratteri. Si presuppone che la stringa rappresenti un numero a virgola mobile decimale con un esponente che sia una potenza di 10.

IBM MQ restituisce il codice di errore RC2472 se la stringa non è formattata correttamente.

- Quando si converte un valore di proprietà numerico in una stringa, il valore viene convertito nella rappresentazione di stringa del valore come numero decimale, non la stringa che contiene il carattere ASCII per tale valore. Ad esempio, il valore intero 65 viene convertito nella stringa "65", non nella stringa "A".
- Quando si converte un valore della proprietà stringa di byte in una stringa, ogni byte viene convertito in due caratteri esadecimali che rappresentano il byte. Ad esempio, l'array di byte {0xF1, 0x12, 0x00, 0xFF} viene convertito nella stringa "F11200FF".

IPQLEN
Interrogare il tipo e la lunghezza del valore della proprietà. La lunghezza viene restituita dal parametro **DataLength** della chiamata MQINQMP. Il valore della proprietà non viene restituito.

Se viene specificato un buffer *ReturnedName* , il campo *VSLength* della struttura MQCHARV viene riempito con la lunghezza del nome della proprietà. Il nome della proprietà non viene restituito.

Opzioni di iterazione: le seguenti opzioni si riferiscono all'iterazione sulle proprietà, utilizzando un nome con un carattere jolly

IPINQF
Analizzare la prima proprietà che corrisponde al nome specificato. Dopo questa chiamata, viene stabilito un cursore sulla proprietà restituita.

Questo è il valore predefinito.

L'opzione IPINQC può essere successivamente utilizzata con una chiamata MQINQMP, se richiesta, per analizzare nuovamente la stessa proprietà.

Notare che è presente un solo cursore della proprietà; pertanto, se il nome della proprietà, specificato nella chiamata MQINQMP, cambia, il cursore viene reimpostato.

Questa opzione non è valida con una delle seguenti opzioni:

IPINQN
IPINQC

IPINQN
Interroga la proprietà successiva che corrisponde al nome specificato, continuando la ricerca dal cursore della proprietà. Il cursore è avanzato alla proprietà restituita.

Se questa è la prima chiamata MQINQMP per il nome specificato, viene restituita la prima proprietà che corrisponde al nome specificato.

L'opzione IPINQC può essere successivamente utilizzata con una chiamata MQINQMP, se richiesta, per analizzare nuovamente la stessa proprietà.

Se la proprietà sotto il cursore è stata eliminata, MQINQMP restituisce la proprietà corrispondente successiva a quella che è stata eliminata.

Se viene aggiunta una proprietà che corrisponde al carattere jolly, mentre è in corso un'iterazione, la proprietà potrebbe essere restituita o meno durante il completamento dell'iterazione. La proprietà viene restituita quando l'iterazione viene riavviata utilizzando IPINQF.

Una proprietà corrispondente al carattere jolly che è stato eliminato, mentre l'iterazione era in corso, non viene restituita dopo la sua eliminazione.

Questa opzione non è valida con una delle seguenti opzioni:

IPINQF
IPINQC

IPINQC
Richiamare il valore della proprietà indicata dal cursore della proprietà. La proprietà a cui punta il cursore della proprietà è quella che è stata interrogata per ultima, utilizzando l'opzione IPINQF o IPINQN.

Il cursore della proprietà viene reimpostato quando viene riutilizzato l'handle del messaggio, quando l'handle del messaggio viene specificato nel campo *MsgHandle* di MQGMO su una chiamata MQGET o quando l'handle del messaggio viene specificato nei campi *OriginalMsgHandle* o *NewMsgHandle* della struttura MQPMO su una chiamata MQPUT.

Se questa opzione viene utilizzata quando il cursore della proprietà non è ancora stato stabilito o se la proprietà indicata dal cursore della proprietà è stata eliminata, la chiamata ha esito negativo con codice di completamento CCFAIL e motivo RC2471.

Questa opzione non è valida con una delle seguenti opzioni:

IPINQF
IPINQN

Se nessuna delle opzioni precedentemente descritte è richiesta, è possibile utilizzare la seguente opzione:

IPNONE
Utilizzare questo valore per indicare che non sono state specificate altre opzioni; tutte le opzioni assumono i propri valori predefiniti.

IPNONE aiuta la documentazione del programma; non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Questo è sempre un campo di input. Il valore iniziale di questo campo è IPINQF.

IPREQCSI (numero intero con segno a 10 cifre)

La serie di caratteri in cui deve essere convertito il valore della proprietà richiesta se il valore è una stringa di caratteri. Questa è anche la serie di caratteri in cui *ReturnedName* deve essere convertito quando viene specificato IPCVAL o IPCTYP.

Il valore iniziale di questo campo è CSAPL.

IPREQENC (numero intero con segno a 10 cifre)

Questa è la codifica in cui il valore della proprietà richiesta deve essere convertito quando viene specificato IPCVAL o IPCTYP.

Il valore iniziale di questo campo è ENNAT.

IPRE1 (numero intero con segno a 10 cifre)

Questo è un campo riservato. Il valore iniziale di questo campo è un carattere vuoto.

IPRETCSI (numero intero con segno a 10 cifre)

Nell'output, questa è la serie di caratteri del valore restituito se il parametro **Type** della chiamata MQINQMP è TYPSTR.

Se l'opzione IPCVAL viene specificata e la conversione ha avuto esito positivo, il campo *ReturnedCCSID* , al ritorno, è lo stesso valore del valore passato.

Il valore iniziale di questo campo è zero.

IPRETENC (numero intero con segno a 10 cifre)

Nell'output, questa è la codifica del valore restituito.

Se l'opzione IPCVAL viene specificata e la conversione ha avuto esito positivo, il campo *ReturnedEncoding* , al ritorno, è lo stesso valore del valore passato.

Il valore iniziale di questo campo è ENNAT.

IPRETNAMCHRP (numero intero con segno a 10 cifre)

Il nome effettivo della proprietà interrogata.

In fase di input è possibile passare un buffer di stringa utilizzando il campo *VSPtr* o *VSOffset* della struttura MQCHARV. La lunghezza del buffer della stringa viene specificato utilizzando il campo *VSBufsize* della struttura MQCHARV.

Al ritorno dalla chiamata MQINQMP, il buffer di stringa viene completato con il nome della proprietà che è stata interrogata, a condizione che il buffer di stringa sia stato sufficientemente lungo per contenere completamente il nome. Il campo *VSLength* della struttura MQCHARV viene riempito con la lunghezza del nome della proprietà. Il campo *VSCCSID* della struttura MQCHARV viene compilato per indicare la serie di caratteri del nome restituito, se la conversione del nome non è riuscita o meno.

Questo è un campo di immissione / emissione. Il valore iniziale di questo campo è MQCHARV_DEFAULT.

IPSID (numero intero con segno a 10 cifre)

Questo è l'identificativo della struttura. Il valore deve essere:

IPSIDV
Identificativo per la struttura delle opzioni della proprietà del messaggio di interrogazione.

Questo è sempre un campo di input. Il valore iniziale di questo campo è IPSIDV.

IPTYP (numero intero con segno a 10 cifre)

Una rappresentazione stringa del tipo di dati della proprietà.

Se la proprietà è stata specificata in un'intestazione MQRFH2 e l'attributo MQRFH2 `dt` non è riconosciuto, questo campo può essere utilizzato per determinare il tipo di dati della proprietà. *`TypeString`* viene restituito nella serie di caratteri codificata 1208 (UTF-8) ed è i primi otto byte del valore dell'attributo `dt` della proprietà che non è stato possibile riconoscere

Questo è sempre un campo di output. Il valore iniziale di questo campo è la stringa nulla nel linguaggio di programmazione C e 8 caratteri vuoti in altri linguaggi di programmazione.

IPVER (numero intero con segno a 10 cifre)

Questo è il numero di versione della struttura. Il valore deve essere:

IPVER1
Numero di versione per la struttura di opzioni della proprietà del messaggio di richiesta.

La seguente costante specifica il numero di versione della versione corrente:

IPVERC
La versione corrente della struttura di opzioni della proprietà del messaggio di interrogazione.

Questo è sempre un campo di input. Il valore iniziale di questo campo è IPVER1.

Valori iniziali

Tabella 707. Campi in MQIPMO

Nome campo	Nome della costante	Valore della costante
IPSID	IPSIDV	'IMPO'
IPVER	IPVER1	1
IPOPT	IPINQF	
IPREQENC	ENNAT	
IPREQCSI	CSAPL	
IPRETENC	ENNAT	
IPRETCSI	0	
IPRE1	0	
IPRETNAMCHRP		
IPTYP		spazi vuoti

Dichiarazione RPG

```
D* MQIMPO Structure
D*
D*
D* Structure identifier
```

```
D IPSID         1   4  INZ('IMPO')
D*
D* Structure version number
D IPVER         5   8I 0 INZ(1)
D*
** Options that control the action of
D* MQINQMP
D IPOPT         9   12I 0 INZ(0)
D*
D* Requested encoding of Value
D IPREQENC       13   16I 0 INZ(273)
D*
** Requested character set identifier
D* of Value
D IPREQCSI       17   20I 0 INZ(-3)
D*
D* Returned encoding of Value
D IPRETENC       21   24I 0 INZ(273)
D*
** Returned character set identifier of
D* Value
D IPRETCSI       25   28I 0 INZ(0)
D*
D* Reserved
D IPRE1         29   32I 0 INZ(0)
D*
D* Returned property name
D* Address of variable length string
D IPRETNAMCHRP     33   48*  INZ(*NULL)
D* Offset of variable length string
D IPRETNAMCHRO     49   52I 0 INZ(0)
D* Size of buffer
D IPRETNAMVSBS     53   56I 0 INZ(-1)
D* Length of variable length string
D IPRETNAMCHRL     57   60I 0 INZ(0)
D* CCSID of variable length string
D IPRETNAMCHRC     61   64I 0 INZ(-3)
D*
D* Property data type as a string
D IPTYP         65   72  INZ
```

IBM i MQMD (Message Descriptor) su IBM i

Panoramica

Scopo: La struttura MQMD contiene le informazioni di controllo che accompagnano i dati dell'applicazione quando un messaggio viaggia tra le applicazioni di invio e di ricezione. La struttura è un parametro di input/output nelle chiamate MQGET, MQPUT e MQPUT1 .

Versione: la versione attuale di MQMD Š MDVER2. I campi che esistono solo nelle versioni più recenti della struttura sono identificati come tali nelle descrizioni che seguono.

Il file COPY fornito contiene la versione più recente di MQMD supportata dall'ambiente, ma con il valore iniziale del campo MDVER impostato su MDVER1. Per utilizzare i campi che non sono presenti nella struttura version-1 , l'applicazione deve impostare il campo MDVER sul numero di versione della versione richiesta.

È disponibile una dichiarazione per la struttura version-1 con nome MQMD1.

Serie di caratteri e codifica: i dati in MQMD devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e nella codifica del gestore code locale fornito da ENNAT. Tuttavia, se l'applicazione è in esecuzione come IBM MQ MQI client, la struttura deve essere nella serie di caratteri e nella codifica del client.

Se i gestori code di invio e di ricezione utilizzano diverse serie di caratteri o codifiche, i dati in MQMD vengono convertiti automaticamente. Non è necessario che l'applicazione converta MQMD.

- “Utilizzo di diverse versioni di MQMD” a pagina 1141
- “Contesto messaggio” a pagina 1141
- “Scadenza messaggio” a pagina 1142

- “Campi” a pagina 1142
- “Valori iniziali” a pagina 1184
- “Dichiarazione RPG” a pagina 1185

Utilizzo di diverse versioni di MQMD

Un MQMD version-2 è in genere equivalente all'utilizzo di un MQMD version-1 e al prefisso dei dati del messaggio con una struttura MQMDE. Tuttavia, se tutti i campi nella struttura MQMDE hanno i valori predefiniti, MQMDE può essere omesso. Viene utilizzato un MQMD version-1 più MQMDE, come descritto più avanti in questa sezione.

- Nelle chiamate MQPUT e MQPUT1 , se l'applicazione fornisce un MQMD version-1 , l'applicazione può facoltativamente aggiungere un prefisso ai dati del messaggio con un MQMDE, impostando il campo MDFMT in MQMD su FMMDE per indicare che è presente un MQMDE. Se l'applicazione non fornisce un MQMDE, il gestore code assume i valori predefiniti per i campi in MQMDE.

  Nota: Molti dei campi presenti in MQMD version-2 ma non in MQMD version-1 sono campi di input / output nelle chiamate MQPUT e MQPUT1 . Tuttavia, il gestore code non restituisce alcun valore nei campi equivalenti in MQMDE sull'output delle chiamate MQPUT e MQPUT1 ; se l'applicazione richiede tali valori di output, deve utilizzare un MQMD version-2 .
- Nella chiamata MQGET, se l'applicazione fornisce un MQMD version-1 , il gestore code prefissa il messaggio restituito con un MQMDE, ma solo se uno o più campi in MQMDE hanno un valore non predefinito. Il campo MDFMT in MQMD avrà il valore FMMDE per indicare che è presente un MQMDE.

I valori predefiniti che il gestore code utilizza per i campi in MQMDE sono uguali ai valori iniziali di questi campi, mostrati in Tabella 709 a pagina 1184.

Quando un messaggio si trova su una coda di trasmissione, alcuni dei campi in MQMD sono impostati su valori particolari; consultare “MQXQH (Transmission - queue header) su IBM i” a pagina 1281 per i dettagli.

Contesto messaggio

Alcuni campi in MQMD contengono il contesto del messaggio. In genere:

- *Contesto identità* correlato all'applicazione che ha inserito originariamente il messaggio
- *Contesto di origine* relativo all'applicazione che ha inserito il messaggio più di recente
- *Contesto utente* fa riferimento all'applicazione che ha originariamente inserito il messaggio.

Queste due applicazioni possono essere la stessa applicazione, ma possono anche essere applicazioni diverse (ad esempio, quando un messaggio viene inoltrato da un'applicazione a un'altra).

Sebbene l'identità e il contesto di origine in genere abbiano i significati descritti in precedenza, il contenuto di entrambi i tipi di campi di contesto in MQMD in realtà dipende dalle opzioni PM* specificate quando viene inserito il messaggio. Di conseguenza, il contesto di identità non è necessariamente correlato all'applicazione che ha originariamente inserito il messaggio e il contesto di origine non è necessariamente correlato all'applicazione che ha inserito il messaggio più di recente, ma dipende dalla progettazione della suite di applicazioni.

Esiste una classe di applicazione che non modifica mai il contesto del messaggio, ovvero l'MCA (message channel agent). Gli MCA che ricevono i messaggi dai gestori code remoti utilizzano l'opzione di contesto PMSETA nella chiamata MQPUT o MQPUT1 . Ciò consente all'MCA ricevente di conservare esattamente il contesto del messaggio che ha viaggiato con il messaggio dall'MCA mittente. Tuttavia, il risultato è che il contesto di origine non si riferisce all'applicazione che ha inserito più di recente il messaggio (l'MCA ricevente), ma si riferisce invece a un'applicazione precedente che ha inserito il messaggio (probabilmente l'applicazione di origine stessa).

Per ulteriori informazioni, consultare Contesto del messaggio.

Scadenza messaggio

I messaggi scaduti su una coda caricata (una coda che è stata aperta) vengono automaticamente rimossi dalla coda entro un periodo di tempo ragionevole dopo la loro scadenza. Alcune altre nuove funzioni di questa release di IBM MQ possono far sì che le code caricate vengano sottoposte a scansione meno frequentemente rispetto alla versione precedente del prodotto, tuttavia i messaggi scaduti sulle code caricate vengono sempre rimossi entro un periodo di tempo ragionevole dalla loro scadenza.

Campi

La struttura MQMD contiene i seguenti campi; i campi sono descritti in ordine alfabetico:

MDACC (stringa bit a 32 byte)

Token di account.

Fa parte del *contesto di identità* del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

MDACC consente a una applicazione di far sì che il lavoro eseguito come risultato del messaggio venga addebitato in maniera appropriata. Il gestore code tratta queste informazioni come una stringa di bit e non ne controlla il contenuto.

Quando il gestore code genera queste informazioni, viene impostato come segue:

- Il primo byte del campo è impostato sulla lunghezza delle informazioni di account presenti nei byte che seguono; questa lunghezza è compresa nell'intervallo tra zero e 30 e viene memorizzata nel primo byte come un numero intero binario.
- Il secondo byte e i byte successivi (come specificato dal campo della lunghezza) vengono impostati sulle informazioni di account appropriate per l'ambiente.
  - z/OS Su z/OS le informazioni di account sono impostate su:
    - Per il batch z/OS , le informazioni di account dalla scheda JES JOB o da un'istruzione JES ACCT nella scheda EXEC (i separatori virgola vengono modificati in X'FF '). Queste informazioni vengono troncate, se necessario, a 31 byte.
    - Per TSO, il numero di account dell'utente.
    - Per CICS, l'identificativo dell'unità di lavoro LU 6.2 (UEPUOWDS) (26 byte).
    - Per IMS, il nome PSB di 8 caratteri concatenato con il token di recupero IMS di 16 caratteri.
  - IBM i Su IBM i, le informazioni di account sono impostate sul codice di account per il lavoro.
  - Linux AIX Su AIX and Linux, le informazioni di account sono impostate sull'identificativo utente numerico, in caratteri ASCII.
  - Windows Su Windows, le informazioni di account sono impostate su un SID (security identifier) Windows NT in un formato compresso. Il SID identifica in modo univoco l'identificativo utente memorizzato nel campo *MDUID* . Quando il SID viene memorizzato nel campo *MDACC* , l'autorità di identificazione a 6 byte (ubicata nel terzo byte e nei byte successivi del SID) viene omessa. Ad esempio, se il SID Windows NT ha una lunghezza di 28 byte, nel campo *MDACC* vengono memorizzati 22 byte di informazioni SID.
- L'ultimo byte è impostato sul tipo di token di account, uno dei seguenti valori:

  ATTCIC
  Identificativo LUOW CICS .

  ATTDOS
  Token di account predefinito PC DOS.

  ATTWNT
  Identificativo di sicurezza Windows .

ATT400
Token di account IBM i .

ATTUNX
Identificativo numerico AIX and Linux .

ATTUSR
Token di account definito dall'utente.

ATTUNK
Tipo di token di conteggio sconosciuto.

Il tipo di token di account è impostato su un valore esplicito solo nei seguenti ambienti:

– 

AIX

– 

IBM i

– 

Windows

e per IBM MQ MQI clients collegati a questi sistemi.

In altri ambienti, il tipo di token di account è impostato sul valore ATTUNK. In questi ambienti il campo MDPAT può essere utilizzato per dedurre il tipo di token di account ricevuto.

- Tutti gli altri byte sono impostati su zero binario.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se PMSETI o PMSETA viene specificato nel parametro **PMO** . Se non viene specificato né PMSETI né PMSETA, questo campo viene ignorato all'immissione ed è un campo di sola emissione. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Dopo il corretto completamento di una chiamata MQPUT o MQPUT1 , questo campo contiene il MDACC che è stato trasmesso con il messaggio se è stato inserito in una coda. Questo sarà il valore di MDACC che viene conservato con il messaggio se viene conservato (consultare la descrizione di PMRET in “MQPMO (Put - message options) su IBM i” a pagina 1206 per ulteriori informazioni sulle pubblicazioni conservate) ma non viene utilizzato come MDACC quando il messaggio viene inviato come una pubblicazione ai sottoscrittori poiché forniscono un valore per sovrascrivere MDACC in tutte le pubblicazioni ad essi inviate. Se il messaggio non ha contesto, il campo è completamente binario zero.

Questo è un campo di output per la chiamata MQGET.

Questo campo non è soggetto ad alcuna conversione basata sulla serie di caratteri del gestore code - il campo viene trattato come una stringa di bit e non come una stringa di caratteri.

Il gestore code non esegue alcuna operazione con le informazioni in questo campo. L'applicazione deve interpretare le informazioni se desidera utilizzarle per scopi contabili.

È possibile utilizzare il seguente valore speciale per il campo *MDACC* :

ACNONE
Nessun token di account specificato.

Il valore è zero binario per la lunghezza del campo.

La lunghezza di questo campo è fornita da LNACCT. Il valore iniziale di questo campo è ACNONE.

MDAID (stringa di caratteri a 32 byte)

Dati dell'applicazione relativi all'identità.

Fa parte del *contesto di identità* del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

MDAID sono informazioni definite dalla suite di applicazioni e possono essere utilizzate per fornire ulteriori informazioni sul messaggio o sul suo creatore. Il gestore code considera queste informazioni

come dati carattere, ma non ne definisce il formato. Quando il gestore code genera queste informazioni, è completamente vuoto.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se PMSETI o PMSETA viene specificato nel parametro **PMO** . Se è presente un carattere null, il valore null e i seguenti caratteri vengono convertiti in spazi vuoti dal gestore code. Se non viene specificato né PMSETI né PMSETA, questo campo viene ignorato all'immissione ed è un campo di sola emissione. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Dopo il corretto completamento di una chiamata MQPUT o MQPUT1 , questo campo contiene il MDAID che è stato trasmesso con il messaggio se è stato inserito in una coda. Questo sarà il valore di MDAID che viene conservato con il messaggio se viene conservato (vedere la descrizione di PMRET per ulteriori dettagli sulle pubblicazioni conservate) ma non viene utilizzato come MDAID quando il messaggio viene inviato come pubblicazione ai sottoscrittori poiché forniscono un valore da sovrascrivere MDAID in tutte le pubblicazioni a loro inviate. Se il messaggio non ha contesto, il campo è completamente vuoto.

Questo è un campo di output per la chiamata MQGET. La lunghezza di questo campo è fornita da LNAIDD. Il valore iniziale di questo campo è di 32 caratteri vuoti.

MDAOD (stringa di caratteri a 4 byte)

Dati di applicazione relativi all'origine.

Questo fa parte del *contesto di origine* del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

MDAOD è un'informazione definita dalla suite di applicazioni che può essere utilizzata per fornire ulteriori informazioni sull'origine del messaggio. Ad esempio, potrebbe essere impostato dalle applicazioni in esecuzione con l'autorizzazione utente appropriata per indicare se i dati di identità sono attendibili.

Il gestore code considera queste informazioni come dati carattere, ma non ne definisce il formato. Quando il gestore code genera queste informazioni, è completamente vuoto.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se PMSETA è specificato nel parametro **PMO** . Tutte le informazioni che seguono un carattere null all'interno del campo vengono scartate. Il carattere null e i seguenti caratteri vengono convertiti in spazi dal gestore code. Se PMSETA non viene specificato, questo campo viene ignorato all'immissione ed è un campo di sola emissione.

Dopo il corretto completamento di una chiamata MQPUT o MQPUT1 , questo campo contiene il MDAOD che è stato trasmesso con il messaggio se è stato inserito in una coda. Questo sarà il valore di MDAOD che viene conservato con il messaggio se viene conservato (vedere la descrizione di PMRET per ulteriori dettagli sulle pubblicazioni conservate) ma non viene utilizzato come MDAOD quando il messaggio viene inviato come pubblicazione ai sottoscrittori poiché forniscono un valore da sovrascrivere MDAOD in tutte le pubblicazioni a loro inviate. Se il messaggio non ha contesto, il campo è completamente vuoto.

Questo è un campo di output per la chiamata MQGET. La lunghezza di questo campo è fornita da LNAORD. Il valore iniziale di questo campo è di 4 caratteri vuoti.

MDBOC (numero intero con segno a 10 cifre)

Contatore backout.

Si tratta di un conteggio del numero di volte in cui il messaggio è stato precedentemente restituito dalla chiamata MQGET come parte di un'unità di lavoro e successivamente ne è stato eseguito il backout. Viene fornito come supporto all'applicazione nel rilevamento degli errori di elaborazione basati sul contenuto del messaggio. Il conteggio esclude le chiamate MQGET che specificano una delle opzioni GMBRW*.

La precisione di questo conteggio è influenzata dall'attributo della coda **HardenGetBackout** ; consultare “Attributi per le code” a pagina 1406.

Questo è un campo di output per la chiamata MQGET. Viene ignorato per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è 0.

MDCID (stringa bit a 24 byte)

Identificativo di correlazione.

Si tratta di una stringa di byte che l'applicazione può utilizzare per correlare un messaggio ad un altro o per correlare il messaggio ad un altro lavoro che l'applicazione sta eseguendo. L'identificativo di correlazione è una proprietà permanente del messaggio e persiste nei riavvii del gestore code. Poiché l'identificativo di correlazione è una stringa di byte e non una stringa di caratteri, l'identificativo di correlazione non viene convertito tra serie di caratteri quando il messaggio passa da un gestore code all'altro.

Per le chiamate MQPUT e MQPUT1 , l'applicazione può specificare qualsiasi valore. Il gestore code trasmette questo valore con il messaggio e lo consegna all'applicazione che emette la richiesta get per il messaggio.

Se l'applicazione specifica PMNCID, il gestore code genera un identificativo di correlazione univoco che viene inviato con il messaggio e restituito anche all'applicazione di invio all'output dalla chiamata MQPUT o MQPUT1 .

Questo identificativo di correlazione generato viene conservato con il messaggio se viene conservato e viene utilizzato come identificativo di correlazione quando il messaggio viene inviato come pubblicazione ai sottoscrittori che specificano CINONE nel campo SDCID nell'MQSD passato alla chiamata MQSUB.

Consultare “MQPMO (Put - message options) su IBM i” a pagina 1206 per ulteriori dettagli sulle pubblicazioni conservate

Quando un gestore code o un agent del canale dei messaggi genera un messaggio di report, imposta il campo MDCID nel modo specificato dal campo MDREP del messaggio originale, ROCMTC o ROPCI. Anche le applicazioni che generano messaggi di report devono eseguire questa operazione.

Per la chiamata MQGET, MDCID è uno dei cinque campi che è possibile utilizzare per selezionare un determinato messaggio da richiamare dalla coda. Consultare la descrizione del campo MDMID per dettagli su come specificare i valori per questo campo.

Specificare CINONE come identificativo di correlazione ha lo stesso effetto di non specificare MOCORI, ovvero, qualsiasi identificativo di correlazione corrisponderà.

Se l'opzione GMMUC viene specificata nel parametro **GMO** nella chiamata MQGET, questo campo viene ignorato.

Al ritorno da una chiamata MQGET, il campo MDCID è impostato sull'identificativo di correlazione del messaggio restituito (se presente).

È possibile utilizzare i seguenti valori speciali:

CINONE
Non è stato specificato alcun identificatore di correlazione.

Il valore è zero binario per la lunghezza del campo.

CINEWS
Il messaggio è l'inizio di una nuova sessione.

Questo valore viene riconosciuto da CICS bridge come indica l'avvio di una nuova sessione, ossia l'avvio di una nuova sequenza di messaggi.

Per la chiamata MQGET, questo è un campo di input/output. Per le chiamate MQPUT e MQPUT1 , questo è un campo di input se PMNCID non è specificato e un campo di output se PMNCID è specificato. La lunghezza di questo campo è fornita da LNCID. Il valore iniziale di questo campo è CINONE.

MDCSI (numero intero con segno a 10 cifre)

Specifica l'identificativo della serie di caratteri dei dati carattere nel messaggio.

Nota: I dati carattere in MQMD e le altre strutture dati IBM MQ che sono parametri sulle chiamate devono trovarsi nella serie di caratteri del gestore code. Questo è definito dall'attributo **CodedCharSetId** del gestore code; consultare "Attributi per il gestore code su IBM i" a pagina 1439 per i dettagli di questo attributo.

È possibile utilizzare i seguenti valori speciali:

CSQM
L'identificativo della serie di caratteri del gestore code.

I dati carattere nel messaggio si trovano nella serie di caratteri del gestore code.

Nelle chiamate MQPUT e MQPUT1 , il gestore code modifica questo valore in MQMD inviato con il messaggio nell'identificativo della serie di caratteri true del gestore code. Di conseguenza, il valore CSQM non viene mai restituito dalla chiamata MQGET.

CINHT
Eredita l'identificativo della serie di caratteri di questa struttura.

I dati carattere nel messaggio si trovano nella stessa serie di caratteri di questa struttura; questa è la serie di caratteri del gestore code. Solo per MQMD, CSINHT ha lo stesso significato di CSQM.

Il gestore code modifica questo valore nell'MQMD inviato con il messaggio all'effettivo CCSID (character set identifier) di MQMD. Se non si verifica alcun errore, il valore CSINHT non viene restituito dalla chiamata MQGET.

CSINHT non può essere utilizzato se il valore del campo MDPAT in MQMD è ATBRKR.

CSEMBD
Identificativo della serie di caratteri incorporata.

I dati carattere nel messaggio si trovano in una serie di caratteri con l'identificativo contenuto nei dati messaggio stessi. È possibile che vi sia un numero qualsiasi di identificativi della serie di caratteri incorporati nei dati del messaggio, che si applicano a parti differenti dei dati. Questo valore deve essere utilizzato per i messaggi PCF che contengono dati in una combinazione di serie di caratteri. I messaggi PCF hanno un nome formato FMPCF.

Specificare questo valore solo sulle chiamate MQPUT e MQPUT1 . Se viene specificato nella chiamata MQGET, impedisce la conversione del messaggio.

Nelle chiamate MQPUT e MQPUT1 , il gestore code modifica i valori CSQM e CSINHT nell'MQMD inviato con il messaggio come descritto in precedenza, ma non modifica l'MQMD specificato nella chiamata MQPUT o MQPUT1 . Nessun altro controllo viene eseguito sul valore specificato.

Le applicazioni che richiamano i messaggi devono confrontare questo campo con il valore previsto dall'applicazione; se i valori differiscono, l'applicazione potrebbe dover convertire i dati carattere nel messaggio.

Se l'opzione GMCONV è specificata nella chiamata MQGET, questo campo è un campo di input/output. Il valore specificato dall'applicazione è il CCSID (coded character set identifier) in cui devono essere convertiti i dati del messaggio, se necessario. Se la conversione ha esito positivo o non è necessaria, il valore non viene modificato (ad eccezione del fatto che il valore CSQM o CSINHT viene convertito nel valore effettivo). Se la conversione ha esito negativo, il valore dopo la chiamata MQGET rappresenta il CCSID (coded character set identifier) del messaggio non convertito restituito all'applicazione.

Altrimenti, questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è CSQM.

MDENC (numero intero con segno a 10 cifre)

Codifica numerica dei dati del messaggio.

Specifica la codifica numerica dei dati numerici nel messaggio; non si applica ai dati numerici nella stessa struttura MQMD. La codifica numerica definisce la rappresentazione utilizzata per numeri interi binari, interi decimali compressi e numeri a virgola mobile.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Il gestore code non verifica la validità del campo. Viene definito il seguente valore speciale:

ENNAT
Codifica macchina nativa.

La codifica è il valore predefinito per il linguaggio di programmazione e la macchina su cui è in esecuzione l'applicazione.

Nota: Il valore di questa costante dipende dal linguaggio di programmazione e dall'ambiente. Per questo motivo, le applicazioni devono essere compilate utilizzando i file di intestazione, macro, COPY o INCLUDE appropriati all'ambiente in cui verrà eseguita l'applicazione.

Le applicazioni che inserano messaggi normalmente devono specificare ENNAT. Le applicazioni che richiamano i messaggi devono confrontare questo campo con il valore ENNAT; se i valori differiscono, l'applicazione potrebbe dover convertire i dati numerici nel messaggio. L'opzione GMCONV può essere utilizzata per richiedere al gestore code di convertire il messaggio come parte dell'elaborazione della chiamata MQGET.

Se l'opzione GMCONV è specificata nella chiamata MQGET, questo campo è un campo di input/output. Il valore specificato dall'applicazione è la codifica in cui i dati del messaggio devono essere convertiti, se necessario. Se la conversione ha esito positivo o non è necessario, il valore non viene modificato. Se la conversione ha esito negativo, il valore dopo la chiamata MQGET rappresenta la codifica del messaggio non convertito restituito all'applicazione.

In altri casi, si tratta di un campo di output per la chiamata MQGET e di un campo di input per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è ENNAT.

MDEXP (numero intero con segno a 10 cifre)

Durata del messaggio.

Si tratta di un periodo di tempo espresso in decimi di secondo, impostato dall'applicazione che inserisce il messaggio. Il messaggio diventa idoneo per l'eliminazione se viene rimosso dalla coda di destinazione prima che trascorra questo periodo di tempo.

Il valore viene ridotto per riflettere il tempo che il messaggio trascorre sulla coda di destinazione e anche su eventuali code di trasmissione intermedie se l'inserimento avviene su una coda remota. Può anche essere ridotto dagli agent del canale dei messaggi per riflettere i tempi di trasmissione, se questi sono significativi. Allo stesso modo, un'applicazione che inoltra questo messaggio a un'altra coda potrebbe diminuire il valore, se necessario, se ha conservato il messaggio per un periodo di tempo significativo. Tuttavia, il tempo di scadenza viene trattato come approssimativo e il valore non deve essere ridotto per riflettere piccoli intervalli di tempo.

Quando il messaggio viene richiamato da un'applicazione che utilizza la chiamata MQGET, il campo MDEXP rappresenta la quantità di tempo di scadenza originale che rimane.

Una volta trascorso il tempo di scadenza di un messaggio, diventa idoneo per essere eliminato dal gestore code. Nelle implementazioni correnti, il messaggio viene eliminato quando si verifica una chiamata MQGET browse o non browse che avrebbe restituito il messaggio se non fosse già scaduto. Ad esempio, una chiamata MQGET non sfogliare con il campo GMMO in MQGMO impostato su MONONE che legge da una coda ordinata FIFO causerà l'eliminazione di tutti i messaggi scaduti fino al primo messaggio non scaduto. Con una coda ordinata con priorità, la stessa chiamata eliminerà i messaggi scaduti di priorità superiore e i messaggi di priorità uguale che sono arrivati sulla coda prima del primo messaggio non scaduto.

Un messaggio scaduto non viene mai restituito a un'applicazione (da una chiamata MQGET di ricerca o non di ricerca), quindi il valore nel campo MDEXP del descrittore del messaggio dopo una chiamata MQGET riuscita è maggiore di zero o il valore speciale EIULIM.

Se un messaggio viene inserito su una coda remota, il messaggio potrebbe scadere (ed essere eliminato) mentre si trova su una coda di trasmissione intermedia, prima che il messaggio raggiunga la coda di destinazione.

Un report viene generato quando un messaggio scaduto viene eliminato, se il messaggio ha specificato una delle opzioni del report ROEXP*. Se nessuna di queste opzioni viene specificata, non viene generato alcun report di questo tipo; si presume che il messaggio non sia più rilevante dopo questo periodo di tempo (forse perché un messaggio successivo lo ha sostituito).

Qualsiasi altro programma che elimina i messaggi in base all'ora di scadenza deve anche inviare un messaggio di report appropriato, se ne è stato richiesto uno.

Nota:

1. Se un messaggio viene inserito con un'ora MDEXP pari a zero, la chiamata MQPUT o MQPUT1 ha esito negativo con codice di errore RC2013; in questo caso non viene generato alcun messaggio di report.
2. Poiché un messaggio con un tempo di scadenza trascorso non può essere effettivamente eliminato fino a un momento successivo, è possibile che vi siano messaggi su una coda che hanno superato il tempo di scadenza e che non sono quindi idonei per il richiamo. Tuttavia, questi messaggi vengono conteggiati per il numero di messaggi sulla coda per tutti gli scopi, incluso il trigger di profondità.
3. Un report di scadenza viene generato, se richiesto, quando il messaggio viene effettivamente eliminato, non quando diventa idoneo per l'eliminazione.
4. L'eliminazione di un messaggio scaduto e la creazione di un report di scadenza, se richiesto, non fanno mai parte dell'unità di lavoro dell'applicazione, anche se il messaggio è stato pianificato per l'eliminazione come risultato di una chiamata MQGET che opera all'interno di un'unità di lavoro.
5. Se un messaggio quasi scaduto viene richiamato da una chiamata MQGET all'interno di un'unità di lavoro e l'unità di lavoro viene successivamente ripristinata, è possibile che il messaggio diventi idoneo per essere eliminato prima che possa essere richiamato di nuovo.
6. Se un messaggio quasi scaduto è bloccato da una chiamata MQGET con GMLK, il messaggio potrebbe diventare idoneo per essere eliminato prima di poter essere richiamato da una chiamata MQGET con GMMUC; il codice motivo RC2034 viene restituito in questa chiamata MQGET successiva, se ciò si verifica.
7. Quando viene richiamato un messaggio di richiesta con una scadenza maggiore di zero, l'applicazione può intraprendere una delle seguenti azioni quando invia il messaggio di risposta:
   - Copiare il tempo di scadenza rimanente dal messaggio di richiesta al messaggio di risposta.
   - Impostare la scadenza nel messaggio di risposta su un valore esplicito maggiore di zero.
   - Impostare la scadenza nel messaggio di replica su EIULIM.

   L'azione da intraprendere dipende dalla progettazione della suite di applicazioni. Tuttavia, l'azione predefinita per inserire i messaggi in una coda di messaggi non recapitabili (messaggi non recapitati) deve essere quella di conservare il tempo di scadenza rimanente del messaggio e di continuare a diminuirlo.
8. I messaggi trigger vengono sempre generati con EIULIM.
9. Un messaggio (normalmente su una coda di trasmissione) che ha un nome MDFMT FMXQH ha un secondo descrittore di messaggi all'interno di MQXQH. Pertanto, dispone di due campi MDEXP associati. In questo caso vanno evidenziati i seguenti punti aggiuntivi:
   - Quando un'applicazione inserisce un messaggio su una coda remota, il gestore code inserisce il messaggio inizialmente su una coda di trasmissione locale e prefissa i dati del messaggio dell'applicazione con una struttura MQXQH. Il gestore code imposta i valori dei due campi MDEXP in modo che siano uguali a quelli specificati dall'applicazione.

     Se un'applicazione inserisce un messaggio direttamente su una coda di trasmissione locale, i dati del messaggio devono già iniziare con una struttura MQXQH e il nome del formato deve essere FMXQH (ma il gestore code non lo applica). In questo caso, l'applicazione non deve impostare i valori di questi due campi MDEXP in modo che siano uguali. Il gestore code non controlla che il campo MDEXP all'interno di MQXQH contenga un valore valido o che i dati del messaggio siano sufficientemente lunghi per includerlo.

- Quando un messaggio con un nome MDFMT di FMXQH viene richiamato da una coda (normale o di trasmissione), il gestore code decresce entrambi questi campi MDEXP con il tempo trascorso in attesa sulla coda. Non viene generato alcun errore se i dati del messaggio non sono abbastanza lunghi da includere il campo MDEXP in MQXQH.
- Il gestore code utilizza il campo MDEXP nel descrittore del messaggio separato (ossia, non quello nel descrittore del messaggio integrato nella struttura MQXQH) per verificare se il messaggio è idoneo per l'eliminazione.
- Se i valori iniziali dei due campi MDEXP erano diversi, è quindi possibile che il tempo MDEXP nel descrittore del messaggio separato quando il messaggio viene richiamato sia maggiore di zero (in modo che il messaggio non sia idoneo per l'eliminazione), mentre il tempo in base al campo MDEXP in MQXQH è trascorso. In questo caso, il campo MDEXP in MQXQH è impostato su zero.

Viene riconosciuto il seguente valore speciale:

EIULIM
Durata illimitata.

Il messaggio ha una scadenza illimitata.

Questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è EIULIM.

MDFB (numero intero con segno a 10 cifre)

Feedback o codice di errore.

Questo viene utilizzato con un messaggio di tipo MTRPRT per indicare la natura del prospetto ed è significativo solo con quel tipo di messaggio. Il campo può contenere uno dei valori FB* o uno dei valori RC*. I codici di feedback sono raggruppati come segue:

FBNONE
Nessun feedback fornito.

FBSFST
Valore più basso per il feedback generato dal sistema.

SSLST FBST
Il valore più alto per il feedback generato dal sistema.

L'intervallo di codici di feedback generati dal sistema da FBSFST a FBSLST include i codici di feedback generali elencati più avanti in questa sezione (FB*) e anche i codici di errore (RC*) che possono verificarsi quando non è possibile inserire il messaggio nella coda di destinazione.

FBAFST
Valore più basso per il feedback generato dall'applicazione.

FBALST
Valore massimo per il feedback generato dall'applicazione.

Le applicazioni che generano messaggi di report non devono utilizzare codici di feedback nell'intervallo di sistema (diverso da FBQUIT), a meno che non vogliano simulare i messaggi di report generati dal gestore code o dall'agent del canale dei messaggi.

Nelle chiamate MQPUT o MQPUT1 , il valore specificato deve essere FBNONE o compreso nell'intervallo di sistema o nell'intervallo dell'applicazione. Questa opzione è selezionata indipendentemente dal valore di MDMT.

Codici di feedback generali

FBCOA
Conferma di arrivo sulla coda di destinazione (vedi ROCOA).

FBCOD
Conferma della consegna alla domanda di ricezione (vedi ROCOD).

FBEXP
Messaggio scaduto.

Il messaggio è stato eliminato perché non è stato rimosso dalla coda di destinazione prima della scadenza.

FBPAN
Notifica di azione positiva (vedere ROPAN).

FBNAN
Notifica di azione negativa (vedere RONAN).

FBQUIT
L'applicazione deve terminare.

Questo può essere utilizzato da un programma di pianificazione del workload per controllare il numero di istanze di un programma applicativo in esecuzione. L'invio di un messaggio MTRPRT con questo codice di feedback a un'istanza del programma applicativo indica a tale istanza che l'elaborazione deve essere arrestata. Tuttavia, l'aderenza a questa convenzione è una questione per l'applicazione; non viene applicata dal gestore code.

IMS-bridge feedback codes: Quando il bridge IMS riceve un codice di rilevamento diverso da zero IMS-OTMA, il bridge IMS converte il codice di rilevamento da esadecimale a decimale, aggiunge il valore FBIERR (300) e inserisce il risultato nel campo MDFB del messaggio di risposta. Ciò comporta che il codice di feedback abbia un valore compreso tra FBIFST (301) e FBILST (399) quando si è verificato un errore IMS-OTMA.

I seguenti codici di feedback possono essere generati dal bridge IMS :

FBDLZ
Lunghezza dati zero.

La lunghezza del segmento era zero nei dati dell'applicazione del messaggio.

FBDLN
Lunghezza dati negativa.

Una lunghezza del segmento era negativa nei dati dell'applicazione del messaggio.

FBDLTB
Lunghezza dati troppo grande.

La lunghezza di un segmento era troppo grande nei dati dell'applicazione del messaggio.

FBBUFO
Overflow del buffer.

Il valore di uno dei campi di lunghezza potrebbe causare l'overflow dei dati nel buffer dei messaggi.

FBLOB1
Lunghezza in errore di uno.

Il valore di uno dei campi di lunghezza era un byte troppo breve.

FBIIH
Struttura MQIIH non valida o mancante.

Il campo MDFMT in MQMD specifica FMIMS, ma il messaggio non inizia con una struttura MQIIH valida.

FBNAFI
ID utente non autorizzato per l'utilizzo in IMS.

L'ID utente contenuto nel descrittore del messaggio MQMD o la parola d'ordine contenuta nel campo IIAUT nella struttura MQIIH, non ha superato la convalida eseguita dal bridge IMS . Di conseguenza, il messaggio non è stato trasmesso a IMS.

FBIERR
Errore non previsto restituito da IMS.

IMSha restituito un errore non previsto. Per ulteriori informazioni sull'errore, consultare il log degli errori IBM MQ sul sistema su cui si trova il bridge IMS .

FBIFST
Valore più basso per il feedback generato da IMS.

I codici di feedback generati da IMSoccupano l'intervallo da FBIFST (300) a FBILST (399). Il codice di rilevamento IMS-OTMA è MDFB meno FBIERR.

FBILST
Valore massimo per il feedback generato da IMS.

CICS-bridge feedback codes: i seguenti codici di feedback possono essere generati da CICS bridge:

FBCAAB
Applicazione terminata in modo anomalo.

Il programma applicativo specificato nel messaggio è terminato in modo anomalo. Questo codice di feedback si verifica solo nel campo DLREA della struttura MQDLH.

FBCAN
Impossibile avviare l'applicazione.

EXEC CICS LINK per il programma applicativo specificato nel messaggio non è riuscito. Questo codice di feedback si verifica solo nel campo DLREA della struttura MQDLH.

FBCBRF
CICS bridge è terminato in modo anomalo senza completare la normale elaborazione degli errori.

FBCCSE
Identificativo della serie di caratteri non valido.

FBCIHE
Struttura dell'intestazione delle informazioni CICS mancante o non valida.

FBCCAE
Lunghezza di CICS commarea non valida.

FBCCIE
Identificativo di correlazione non valido.

FBCDLQ
Coda di messaggi non instradabili non disponibile.

L'attività CICS bridge non è stata in grado di copiare una risposta a questa richiesta nella coda di messaggi non instradabili. La richiesta è stata ripristinata.

FBCENE
Codifica non valida.

FBCINA
CICS bridge ha rilevato un errore non previsto.

Questo codice di feedback si verifica solo nel campo DLREA della struttura MQDLH.

FBCNTA
Identificativo utente non autorizzato o parola d'ordine non valida.

Questo codice di feedback si verifica solo nel campo DLREA della struttura MQDLH.

FBCUBO
Unità di lavoro ripristinata.

L'unità di lavoro è stata ripristinata per uno dei seguenti motivi:

- È stato rilevato un errore durante l'elaborazione di un'altra richiesta all'interno della stessa unità di lavoro.
- Si è verificata una fine anomala CICS mentre l'unità di lavoro era in corso.

FBCUWE
Campo di controllo dell'unità di lavoro CIUOW non valido.

MQ codici motivo: per i messaggi di report di eccezione, MDFB contiene un codice motivo MQ . Tra i codici di errore possibili sono:

RC2051
(2051, X'803 ') Chiamate Put inibite per la coda.

RC2053
(2053, X'805 ') La coda contiene già il numero massimo di messaggi.

RC2035
(2035, X'7F3') Non autorizzato per l'accesso.

RC2056
(2056, X'808 ') Nessuno spazio disponibile sul disco per la coda.

RC2048
(2048, X'800 ') La coda non supporta i messaggi persistenti.

RC2031
(2031, X'7EF') La lunghezza del messaggio è maggiore del massimo consentito per il gestore code.

RC2030
(2030, X'7EE') La lunghezza del messaggio è maggiore del massimo consentito per la coda.

Questo è un campo di output per la chiamata MQGET e un campo di immissione per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è FBNONE.

MDFMT (stringa di caratteri a 8 byte)

Nome formato dei dati del messaggio.

Questo è un nome che il mittente del messaggio può utilizzare per indicare al destinatario la natura dei dati nel messaggio. Tutti i caratteri che si trovano nella serie di caratteri del gestore code possono essere specificati per il nome, ma si consiglia di limitare il nome ai seguenti:

- Maiuscolo da A a Z
- Cifre numeriche da 0 a 9

Se vengono utilizzati altri caratteri, potrebbe non essere possibile tradurre il nome tra le serie di caratteri dei gestori code di invio e di ricezione.

Il nome deve essere riempito con spazi vuoti fino alla lunghezza del campo o con un carattere null utilizzato per terminare il nome prima della fine del campo; il valore null e i caratteri successivi vengono trattati come spazi vuoti. Non specificare un nome con spazi vuoti iniziali o intermedi. Per la chiamata MQGET, il gestore code restituisce il nome riempito con spazi vuoti alla lunghezza del campo.

Il gestore code non controlla che il nome sia conforme ai suggerimenti precedentemente descritti.

I nomi che iniziano con "MQ" in maiuscolo, minuscolo e con caratteri misti hanno significati definiti dal gestore code; non utilizzare i nomi che iniziano con queste lettere per i propri formati. I formati integrati del gestore code sono:

FMNONE
Nessun nome formato.

La natura dei dati non è definita. Ciò significa che i dati non possono essere convertiti quando il messaggio viene richiamato da una coda utilizzando l'opzione GMCONV.

Se GMCONV viene specificato nella chiamata MQGET e la serie di caratteri o la codificazione dei dati nel messaggio differisce da quella specificata nel parametro **MSGDSC** , il messaggio viene restituito con i seguenti codici di completamento e motivo (supponendo che non vi siano altri errori):

- Codice di completamento CCWARN e codice motivo RC2110 se i dati FMNONE si trova all'inizio del messaggio.

- Codice di completamento CCOK e codice motivo RCNONE se i dati FMNONE sono alla fine del messaggio (ovvero, preceduti da una o più strutture di intestazione MQ). Le strutture dell'intestazione MQ vengono convertite nella serie di caratteri richiesta e codificate in questo caso.

MN FMADM
Messaggio del server di comando di richiesta/risposta.

Il messaggio è una richiesta del server dei comandi o un messaggio di risposta in formato PCF (programmable command format). I messaggi di questo formato possono essere convertiti se l'opzione GMCONV è specificata sulla chiamata MQGET. Per ulteriori informazioni sull'utilizzo dei messaggi di formato dei comandi programmabili, consultare Utilizzo dei formati dei comandi programmabili.

FMCICS
Intestazione delle informazioni CICS .

I dati del messaggio iniziano con l'intestazione delle informazioni CICS MQCIH, seguita dai dati dell'applicazione. Il nome del formato dei dati dell'applicazione è fornito dal campo CIFMT nella struttura MQCIH.

FMCMD1
Messaggio di replica di comando tipo 1.

Il messaggio è un messaggio di risposta del server di comandi MQSC contenente il conteggio oggetti, il codice di completamento e il codice motivo. I messaggi di questo formato possono essere convertiti se l'opzione GMCONV è specificata sulla chiamata MQGET.

FMCMD2
Messaggio di risposta del comando di tipo 2.

Il messaggio è un messaggio di risposta del server di comandi MQSC contenente informazioni sugli oggetti richiesti. I messaggi di questo formato possono essere convertiti se l'opzione GMCONV è specificata sulla chiamata MQGET.

FMDLH
Intestazione non instradabile.

I dati del messaggio iniziano con l'intestazione non instradabile MQDLH. I dati del messaggio originale seguono immediatamente la struttura MQDLH. Il nome del formato dei dati del messaggio originale viene fornito dal campo DLFMT nella struttura MQDLH; consultare “MQDLH (Intestazione non instradabile) su IBM i” a pagina 1093 per i dettagli di questa struttura. I messaggi di questo formato possono essere convertiti se l'opzione GMCONV è specificata sulla chiamata MQGET.

I report COA e COD non vengono creati per i messaggi che hanno un MDFMT di FMDLH.

FMDH
Intestazione elenco di distribuzione.

I dati del messaggio iniziano con l'intestazione dell'elenco di distribuzione MQDH; ciò include gli array dei record MQOR e MQPMR. L'intestazione dell'elenco di distribuzione può essere seguita da ulteriori dati. Il formato dei dati aggiuntivi (se presenti) viene fornito dal campo DHFMT nella struttura MQDH; consultare “MQDH (Distribution header) su IBM i” a pagina 1088 per i dettagli di questa struttura. I messaggi con formato FMDH possono essere convertiti se l'opzione GMCONV è specificata nella chiamata MQGET.

FMEVNT
Messaggio evento.

Il messaggio è un messaggio di eventi di MQ che riporta un evento che si è verificato. I messaggi di evento hanno la stessa struttura dei comandi programmabili; per ulteriori informazioni su questa struttura, consultare Strutture per comandi e risposte. Per informazioni sugli eventi, vedere Monitoraggio eventi.

I messaggi di evento Version-1 possono essere convertiti se l'opzione GMCONV viene specificata nella chiamata MQGET.

FMIMS
Intestazione delle informazioni IMS .

I dati del messaggio iniziano con l'intestazione delle informazioni IMS MQIIH, seguita dai dati applicazione. Il nome del formato dei dati dell'applicazione viene fornito dal campo *IIFMT* nella struttura MQIIH. I messaggi di questo formato possono essere convertiti se l'opzione GMCONV è specificata sulla chiamata MQGET.

FMIMVS
Stringa variabile IMS .

Il messaggio è una stringa variabile IMS , che è una stringa nel formato `llzzccc`, dove:

ll
è un campo di lunghezza di 2 byte che specifica la lunghezza totale dell'elemento stringa della variabile IMS . Questa lunghezza è uguale alla lunghezza di `ll` (2 byte), più la lunghezza di `zz` (2 byte), più la lunghezza della stringa di caratteri stessa. `ll` è un numero intero binario a 2 byte nella codifica specificata dal campo `MDENC` .

zz
è un campo a 2 byte contenente indicatori significativi per IMS. `zz` è una stringa di byte costituita da due campi stringa di bit a 1 byte e viene trasmessa senza modifiche dal mittente al ricevente (ovvero, `zz` non è soggetto ad alcuna conversione).

ccc
è una stringa di caratteri a lunghezza variabile contenente `ll-4` caratteri. `ccc` si trova nella serie di caratteri specificata dal campo `MDCSI` .

I messaggi di questo formato possono essere convertiti se l'opzione GMCONV è specificata sulla chiamata MQGET.

FMMDE
Estensione del descrittore messaggi.

I dati del messaggio iniziano con l'estensione MQMDE del descrittore del messaggio e sono facoltativamente seguiti da altri dati (di solito i dati del messaggio dell'applicazione). Il nome formato, la serie di caratteri e la codifica dei dati che seguono MQMDE sono forniti dai campi `MEFMT`, `MECSI`e `MEENC` in MQMDE. Consultare “MQMDE (estensione descrittore messaggi) su IBM i” a pagina 1186 per i dettagli di questa struttura. I messaggi di questo formato possono essere convertiti se l'opzione GMCONV è specificata sulla chiamata MQGET.

FMPCF
Messaggio definito dall'utente in formato PCF (programmable command format).

Il messaggio è un messaggio definito dall'utente che è conforme alla struttura di un messaggio PCF (programmable command format). I messaggi di questo formato possono essere convertiti se l'opzione GMCONV è specificata sulla chiamata MQGET. Consultare Utilizzo dei formati di comando programmabili per ulteriori informazioni sull'utilizzo dei messaggi del formato di comandi programmabili.

MMRM
Intestazione del messaggio di riferimento.

I dati del messaggio iniziano con l'intestazione del messaggio di riferimento MQRMH e sono facoltativamente seguiti da altri dati. Il nome formato, la serie di caratteri e la codifica dei dati sono forniti dai campi `RMFMT`, `RMCSI`e `RMENC` in MQRMH. Consultare “MQRMH (Reference message header) su IBM i” a pagina 1234 per i dettagli di questa struttura. I messaggi di questo formato possono essere convertiti se l'opzione GMCONV è specificata sulla chiamata MQGET.

FMRF
Regole e intestazione di formattazione.

I dati del messaggio iniziano con le regole e l'intestazione di formattazione MQRFH ed è facoltativamente seguita da altri dati. Il nome del formato, la serie di caratteri e la codifica dei dati (se presenti) vengono forniti dai campi RFFMT, RFCSIe RFENC in MQRFH. I messaggi di questo formato possono essere convertiti se l'opzione GMCONV è specificata sulla chiamata MQGET.

FMRFH2
Intestazione regola e formattazione versione 2.

I dati del messaggio iniziano con le regole version-2 e l'intestazione di formattazione MQRFH2ed è facoltativamente seguita da altri dati. Il nome del formato, la serie di caratteri e la codifica dei dati facoltativi (se presenti) vengono forniti dai campi RF2FMT, RF2CSIe RF2ENC in MQRFH2. I messaggi di questo formato possono essere convertiti se l'opzione GMCONV è specificata sulla chiamata MQGET.

FMSTR
Messaggio composto interamente da caratteri.

I dati del messaggio dell'applicazione possono essere una stringa SBCS (single - byte character set) o una stringa DBCS (double - byte character set). I messaggi di questo formato possono essere convertiti se l'opzione GMCONV è specificata sulla chiamata MQGET.

FMTM
Messaggio di attivazione.

Il messaggio è un messaggio di trigger, descritto dalla struttura MQTM; consultare "MQTM - Messaggio trigger" a pagina 1271 per dettagli su questa struttura. I messaggi di questo formato possono essere convertiti se l'opzione GMCONV è specificata sulla chiamata MQGET.

FMWIH
Intestazione delle informazioni sul lavoro.

I dati del messaggio iniziano con l'intestazione delle informazioni di lavoro MQWIH, seguita dai dati dell'applicazione. Il nome del formato dei dati dell'applicazione viene fornito dal campo WIFMT nella struttura MQWIH.

FMXQH
Intestazione della coda di trasmissione.

I dati del messaggio iniziano con l'intestazione della coda di trasmissione MQXQH. I dati provenienti dal messaggio originale seguono immediatamente la struttura MQXQH. Il nome formato dei dati del messaggio originale viene fornito dal campo MDFMT della struttura MQMD che fa parte dell'intestazione della coda di trasmissione MQXQH. Consultare "MQXQH (Transmission - queue header) su IBM i" a pagina 1281 per i dettagli di questa struttura.

I report COA e COD non vengono generati per i messaggi che hanno un MDFMT di FMXQH.

Questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . La lunghezza di questo campo è fornita da LNFMT. Il valore iniziale di questo campo è FMNONE.

MDGID (stringa bit a 24 byte)

Identificativo gruppo.

Si tratta di una stringa di byte utilizzata per identificare il particolare gruppo di messaggi o messaggio logico a cui appartiene il messaggio fisico. MDGID viene utilizzato anche se la segmentazione è consentita per il messaggio. In tutti questi casi, MDGID ha un valore non null e uno o più dei seguenti indicatori è impostato nel campo MDMFL :

- MMIG
- MFLMIG
- MFSEG
- MFLSEG
- MFSEGA

Se nessuno di questi indicatori è impostato, MDGID ha il valore null speciale GINONE.

Questo campo non deve essere impostato dall'applicazione nella chiamata MQPUT o MQGET se:

- Nella chiamata MQPUT, viene specificato PMLOGO.
- Sulla chiamata MQGET, MOGRPI non viene specificato.

Utilizzare queste chiamate per i messaggi che non sono messaggi di report. Tuttavia, se l'applicazione richiede un maggiore controllo o la chiamata è MQPUT1, l'applicazione deve verificare che MDGID sia impostata su un valore appropriato.

I gruppi di messaggi e i segmenti possono essere elaborati correttamente solo se l'identificativo del gruppo è univoco. Per questo motivo, le applicazioni non dovrebbero generare i propri identificatori di gruppo; invece, le applicazioni devono effettuare una delle seguenti operazioni:

- Se viene specificato PMLOGO, il gestore code genera automaticamente un identificativo gruppo univoco per il primo messaggio nel gruppo o segmento del messaggio logico e utilizza tale identificativo per i restanti messaggi nel gruppo o nei segmenti del messaggio logico, in modo che l'applicazione non debba eseguire alcuna azione speciale. Utilizzare questa procedura.
- Se PMLOGO non è specificato, l'applicazione deve richiedere al gestore code di generare l'identificativo del gruppo, impostando MDGID su GINONE sulla prima chiamata MQPUT o MQPUT1 per un messaggio nel gruppo o segmento del messaggio logico. L'identificativo del gruppo restituito dal gestore code sull'output di tale chiamata deve essere utilizzato per i restanti messaggi nel gruppo o nei segmenti del messaggio logico. Se un gruppo di messaggi contiene messaggi segmentati, è necessario utilizzare lo stesso identificativo di gruppo per tutti i segmenti e i messaggi nel gruppo.

  Quando PMLOGO non è specificato, i messaggi in gruppi e segmenti di messaggi logici possono essere inseriti in qualsiasi ordine (ad esempio, in ordine inverso), ma l'identificativo del gruppo deve essere allocato dalla prima chiamata MQPUT o MQPUT1 emessa per uno qualsiasi di questi messaggi.

All'input delle chiamate MQPUT e MQPUT1 , il gestore code utilizza il valore descritto in dettaglio in PMOPT. Nell'output delle chiamate MQPUT e MQPUT1 , il gestore code imposta questo campo sul valore che è stato inviato con il messaggio se l'oggetto aperto è una coda singola e non un elenco di distribuzione, ma lo lascia invariato se l'oggetto aperto è un elenco di distribuzione. In quest' ultimo caso, se l'applicazione deve conoscere gli identificativi di gruppo generati, l'applicazione deve fornire i record MQPMR contenenti il campo PRGID .

In fase di input della chiamata MQGET, il gestore code utilizza il valore descritto nella Tabella 1. Nell'output della chiamata MQGET, il gestore code imposta questo campo sul valore per il messaggio richiamato.

Viene definito il seguente valore speciale:

GINONE
Nessun identificativo di gruppo specificato.

Il valore è zero binario per la lunghezza del campo. Questo è il valore utilizzato per i messaggi che non sono in gruppi, non in segmenti di messaggi logici e per cui la segmentazione non è consentita.

La lunghezza di questo campo è data da LNGID. Il valore iniziale di questo campo è GINONE. Questo campo viene ignorato se MDVER è inferiore a MDVER2.

MDMFL (numero intero con segno a 10 cifre)

Gli indicatori del messaggio.

Si tratta di indicatori che specificano gli attributi del messaggio o ne controllano l'elaborazione. Le bandiere sono divise nelle seguenti categorie:

- Indicatore di segmentazione
- Indicatori di stato

Questi sono descritti a loro volta.

Indicatori di segmentazione: quando un messaggio è troppo grande per una coda, un tentativo di inserire il messaggio nella coda di solito non riesce. La segmentazione è una tecnica con cui il gestore code o l'applicazione suddivide il messaggio in parti più piccole denominate segmenti e colloca ciascun segmento nella coda come un messaggio fisico separato. L'applicazione che richiama il messaggio può richiamare i segmenti uno per uno oppure richiedere al gestore code di riassemblare i segmenti in un singolo messaggio restituito dalla chiamata MQGET. Quest' ultimo si ottiene specificando l'opzione GMCMPM nella chiamata MQGET e fornendo un buffer abbastanza grande da accogliere il messaggio completo. (Consultare “MQGMO (opzioni Get - message) su IBM i” a pagina 1105 per dettagli sull'opzione GMCMPM.) La segmentazione di un messaggio può verificarsi sul gestore code di invio, su un gestore code intermedio o sul gestore code di destinazione.

È possibile specificare una delle seguenti opzioni per controllare la segmentazione di un messaggio:

MFSEGI
Segmentazione inibita.

Questa opzione impedisce che il messaggio venga suddiviso in segmenti dal gestore code. Se specificata per un messaggio che è già un segmento, questa opzione impedisce che il segmento venga suddiviso in segmenti più piccoli.

Il valore di questo indicatore è zero binario. Questa è l'opzione predefinita.

MFSEGA
Segmentazione consentita.

Questa opzione consente al gestore code di suddividere il messaggio in segmenti. Se specificata per un messaggio che è già un segmento, questa opzione consente di suddividere il segmento in segmenti più piccoli. MFSEGA può essere impostato senza MFSEG o MFLSEG.

Quando il gestore code segmenta un messaggio, il gestore code attiva l'indicatore MFSEG nella copia di MQMD che viene inviato con ciascun segmento, ma non modifica le impostazioni di tali indicatori nell'MQMD fornito dall'applicazione nella chiamata MQPUT o MQPUT1 . Per l'ultima parte del messaggio logico, il gestore code attiva anche l'indicatore MFLSEG nell'MQMD inviato con il segmento.

Nota: È necessario prestare attenzione quando i messaggi vengono inseriti con MFSEGA ma senza PMLOGO. Se il messaggio è:

- Non è un segmento
- Non in un gruppo, e
- Non inoltrato,

l'applicazione deve ricordarsi di reimpostare il campo MDGID su GINONE prima di ogni chiamata MQPUT o MQPUT1 , in modo da generare un identificativo gruppo univoco dal gestore code per ogni messaggio. Se questa operazione non viene eseguita, i messaggi non correlati potrebbero inavvertitamente finire con lo stesso identificativo del gruppo, il che potrebbe portare a un'elaborazione non corretta in seguito. Consultare le descrizioni del campo MDGID e dell'opzione PMLOGO per ulteriori informazioni su quando il campo MDGID deve essere reimpostato.

Il gestore code suddivide i messaggi in segmenti, in base alle necessità, per garantire che i segmenti (più i dati di intestazione che possono essere richiesti) rientrino nella coda. Tuttavia, è presente un limite inferiore per la dimensione di un segmento generato dal gestore code e solo l'ultimo segmento creato da un messaggio può essere inferiore a questo limite. (Il limite inferiore per la dimensione di un segmento generato dall'applicazione è un byte.) I segmenti generati dal gestore code possono essere di lunghezza diversa. Il gestore code elabora il messaggio nel modo seguente:

- I formati definiti dall'utente sono suddivisi su limiti che sono multipli di 16 byte. Ciò significa che il gestore code non genererà segmenti inferiori a 16 byte (diversi dall'ultimo segmento).
- I formati incorporati diversi da FMSTR sono suddivisi in punti appropriati alla natura dei dati presenti. Tuttavia, il gestore code non suddivide mai un messaggio nel mezzo di una struttura di

intestazione MQ . Ciò significa che un segmento contenente una singola struttura di intestazione MQ non può essere ulteriormente suddiviso dal gestore code e, di conseguenza, la dimensione minima del segmento possibile per quel messaggio è maggiore di 16 byte.

Il secondo segmento o quello successivo generato dal gestore code inizierà con uno dei seguenti:

- Una struttura di intestazione MQ
- L'inizio dei dati del messaggio dell'applicazione
- Part - way attraverso i dati del messaggio dell'applicazione

- FMSTR viene suddiviso senza considerare la natura dei dati presenti (SBCS, DBCS o SBCS/DBCS misti). Quando la stringa è DBCS o SBCS/DBCS misto, ciò può risultare in segmenti che non possono essere convertiti da una serie di caratteri ad un'altra. Il gestore code non suddivide mai i messaggi FMSTR in segmenti inferiori a 16 byte (diversi dall'ultimo segmento).
- I campi `MDFMT`, `MDCSIe` `MDENC` in MQMD di ciascun segmento sono impostati dal gestore code per descrivere correttamente i dati presenti all'inizio del segmento; il nome del formato sarà il nome di un formato integrato o il nome di un formato definito dall'utente.
- Il campo `MDREP` nell'MQMD dei segmenti con `MDOFF` maggiore di zero viene modificato come segue:
  - Per ogni tipo di report, se l'opzione di report è RO* D, ma il segmento non può contenere nessuno dei primi 100 byte di dati utente (ovvero, i dati che seguono le strutture di intestazione MQ che possono essere presenti), l'opzione di report viene modificata in RO*.

Il gestore code segue le regole precedenti, ma altrimenti suddivide i messaggi in modo imprevedibile; non fare supposizioni su dove viene suddiviso un messaggio

Per i messaggi persistenti, il gestore code può effettuare la segmentazione solo all'interno di un'unità di lavoro:

- Se la chiamata MQPUT o MQPUT1 è in funzione all'interno di un'unità di lavoro definita dall'utente, viene utilizzata tale unità di lavoro. Se la chiamata ha esito negativo durante il processo di segmentazione, il gestore code rimuove tutti i segmenti che sono stati inseriti nella coda come risultato della chiamata non riuscita. Tuttavia, l'errore non impedisce il corretto commit dell'unità di lavoro.
- Se la chiamata opera al di fuori di un'unità di lavoro definita dall'utente e non esiste alcuna unità di lavoro definita dall'utente, il gestore code crea un'unità di lavoro solo per la durata della chiamata. Se la chiamata ha esito positivo, il gestore code esegue automaticamente il commit dell'unità di lavoro (non è necessario che l'applicazione esegua questa operazione). Se la chiamata ha esito negativo, il gestore code esegue il backout dell'unità di lavoro.
- Se la chiamata opera all'esterno di un'unità di lavoro definita dall'utente, ma esiste un'unità di lavoro definita dall'utente, il gestore code non è in grado di eseguire la segmentazione. Se il messaggio non richiede la segmentazione, la chiamata può ancora avere esito positivo. Ma se il messaggio richiede la segmentazione, la chiamata ha esito negativo con codice di errore RC2255.

Per i messaggi non persistenti, il gestore code non richiede che sia disponibile un'unità di lavoro per eseguire la segmentazione.

Si deve prestare particolare attenzione alla conversione dei dati dei messaggi che possono essere segmentati:

- Se la conversione dei dati viene eseguita solo dall'applicazione ricevente sulla chiamata MQGET e l'applicazione specifica l'opzione GMCMPM, all'uscita di conversione dei dati verrà trasmesso il messaggio completo per l'uscita da convertire e il fatto che il messaggio è stato segmentato non sarà evidente all'uscita.
- Se l'applicazione ricevente richiama un segmento alla volta, verrà richiamata l'uscita di conversione dati per convertire un segmento alla volta. L'uscita deve quindi essere in grado di convertire i dati in un segmento indipendentemente dai dati in uno qualsiasi degli altri segmenti.

Se la natura dei dati nel messaggio è tale che la segmentazione arbitraria dei dati sui limiti di 16 byte può risultare in segmenti che non possono essere convertiti dall'exit oppure il formato è FMSTR e la serie di caratteri è DBCS o SBCS/DBCS misti, l'applicazione di invio deve creare e inserire i segmenti, specificando MFSEGI per eliminare ulteriore segmentazione. In questo modo, l'applicazione mittente può garantire che ogni segmento contenga informazioni sufficienti per consentire all'exit di conversione dati di convertire correttamente il segmento.

- Se la conversione del mittente viene specificata per un MCA (message channel agent) di invio, l'MCA converte solo i messaggi che non sono segmenti di messaggi logici; l'MCA non tenta mai di convertire i messaggi che sono segmenti.

Questo indicatore è un indicatore di input nelle chiamate MQPUT e MQPUT1 e un indicatore di output nella chiamata MQGET. Nell'ultima chiamata, il gestore code ripete anche il valore dell'indicatore al campo GMSEG in MQGMO.

Il valore iniziale di questo indicatore è MFSEGI.

Indicatori di stato: sono indicatori che indicano se il messaggio fisico appartiene a un gruppo di messaggi, è un segmento di un messaggio logico, entrambi o nessuno dei due. Uno o più dei seguenti valori possono essere specificati nella chiamata MQPUT o MQPUT1 o restituiti dalla chiamata MQGET:

MMIG
Il messaggio è un membro di un gruppo.

MFLMIG
Il messaggio è l'ultimo messaggio logico in un gruppo.

Se questo indicatore è impostato, il Gestore code attiva MFMIG nella copia di MQMD che viene inviata con il messaggio, ma non modifica le impostazioni di tali indicatori nell'MQMD fornito dall'applicazione nella chiamata MQPUT o MQPUT1 .

È valido per un gruppo composto da un solo messaggio logico. In questo caso, MFLMIG è impostato, ma il campo MDSEQ ha il valore uno.

MFSEG
Il messaggio è un segmento di un messaggio logico.

Quando MFSEG viene specificato senza MFLSEG, la lunghezza dei dati del messaggio dell'applicazione nel segmento (escludendo le lunghezze di tutte le strutture di intestazione MQ che possono essere presenti) deve essere almeno una. Se la lunghezza è zero, la chiamata MQPUT o MQPUT1 ha esito negativo con codice di errore RC2253.

MFLSEG
Il messaggio è l'ultimo segmento di un messaggio logico.

Se questo indicatore è impostato, il gestore code attiva MFSEG nella copia di MQMD inviata con il messaggio, ma non modifica le impostazioni di tali indicatori nell'MQMD fornito dall'applicazione sulla chiamata MQPUT o MQPUT1 .

È valido per un messaggio logico composto da un solo segmento. In questo caso, MFLSEG è impostato, ma il campo MDOFF ha il valore zero.

Quando viene specificato MFLSEG, è consentito che la lunghezza dei dati del messaggio dell'applicazione nel segmento (escluse le lunghezze delle strutture di intestazione che possono essere presenti) sia zero.

L'applicazione deve assicurarsi che questi indicatori siano impostati correttamente quando si inseriscono i messaggi. Se PMLOGO è specificato o è stato specificato nella precedente chiamata MQPUT per l'handle della coda, le impostazioni degli indicatori devono essere congruenti con le informazioni sul gruppo e sul segmento conservate dal gestore code per l'handle della coda. Le seguenti condizioni si applicano alle successive chiamate MQPUT per l'handle di coda quando viene specificato PMLOGO:

- Se non è presente alcun gruppo corrente o messaggio logico, tutti questi indicatori (e le relative combinazioni) sono validi.

- Una volta specificato MFMIG, deve rimanere attivo fino a quando non viene specificato MFLMIG. La chiamata ha esito negativo con codice di errore RC2241 se questa condizione non viene soddisfatta.
- Una volta specificato MFSEG, deve rimanere attivo fino a quando non viene specificato MFLSEG. La chiamata ha esito negativo con codice di errore RC2242 se questa condizione non viene soddisfatta.
- Una volta che MFSEG è stato specificato senza MFMIG, MFMIG deve rimanere spento fino a quando non è stato specificato MFLSEG. La chiamata ha esito negativo con codice di errore RC2242 se questa condizione non viene soddisfatta.

La Tabella 1 mostra le combinazioni valide degli indicatori e i valori utilizzati per i vari campi.

Questi indicatori sono indicatori di input per le chiamate MQPUT e MQPUT1 e indicatori di output per la chiamata MQGET. In quest' ultima chiamata, il gestore code ripete anche i valori degli indicatori ai campi GMGST e GMSST in MQGMO.

Indicatori predefiniti: è possibile specificare quanto segue per indicare che il messaggio ha attributi predefiniti:

MFNONE
Nessun indicatore di messaggio (attributi di messaggio predefiniti).

Ciò impedisce la segmentazione e indica che il messaggio non è in un gruppo e non è un segmento di un messaggio logico. MFNONE è definito per aiutare la documentazione del programma. Non si intende utilizzare questo indicatore con altri, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Il campo MDMFL è suddiviso in sottocampi; per i dettagli, vedere "Opzioni di report e indicatori di messaggi su IBM i" a pagina 1473.

Il valore iniziale di questo campo è MFNONE. Questo campo viene ignorato se MDVER è inferiore a MDVER2.

MDMID (stringa bit a 24 byte)

L'identificativo del messaggio.

È una stringa di byte utilizzata per distinguere un messaggio da un altro. Generalmente, due messaggi non devono avere lo stesso identificativo di messaggio, sebbene ciò non sia consentito dal gestore code. L'identificativo del messaggio è una proprietà permanente del messaggio e persiste durante i riavvii del gestore code. Poiché l'identificativo del messaggio è una stringa di byte e non una stringa di caratteri, l'identificativo del messaggio non viene convertito tra le serie di caratteri quando il messaggio passa da un gestore code all'altro.

Per le chiamate MQPUT e MQPUT1 , se MINONE o PMNMID è specificato dall'applicazione, il gestore code genera un identificativo del messaggio univoco quando il messaggio viene inserito e lo inserisce nel descrittore del messaggio inviato con il messaggio. Il gestore code restituisce questo identificativo del messaggio anche nel descrittore del messaggio appartenente all'applicazione mittente. L'applicazione può utilizzare questo valore per registrare informazioni su messaggi particolari e per rispondere alle query provenienti da altre parti dell'applicazione.

Un MDMID generato dal gestore code è costituito da un identificativo del prodotto a 4 byte (AMQ¬ o CSQ¬ in ASCII o EBCDIC, dove ¬ rappresenta un singolo carattere vuoto), seguito da un'implementazione specifica del prodotto di una stringa univoca. In IBM MQ contiene i primi 12 caratteri del nome del gestore code e un valore derivato dall'orologio di sistema. Tutti i gestori code che possono intercomunicare devono quindi avere nomi che differiscono nei primi 12 caratteri, per garantire che gli identificatori dei messaggi siano univoci. La capacità di generare una stringa univoca dipende anche dal fatto che l'orologio di sistema non venga modificato all'indietro. Per eliminare la possibilità che un identificativo di messaggio generato dal gestore code ne duplici uno generato dall'applicazione, l'applicazione dovrebbe evitare di creare identificatori con caratteri iniziali compresi nell'intervallo da A a I in ASCII o EBCDIC (da X'41 'a X'49' e da X'C1'a X'C9'). Tuttavia, all'applicazione non viene impedito di generare identificatori con caratteri iniziali in questi intervalli.

Se il messaggio viene inserito in un argomento, il gestore code genera identificatori di messaggi univoci come necessario per ogni messaggio pubblicato. Se PMNMID viene specificato

dall'applicazione, il gestore code genera un identificativo di messaggio univoco da restituire all'output. Se MINONE viene specificato dall'applicazione, il valore del campo MDMID in MQMD non viene modificato al ritorno dalla chiamata.

Consultare la descrizione di PMRET in PMOPT per ulteriori informazioni sulle pubblicazioni conservate.

Se il messaggio viene inserito in un elenco di distribuzione, il gestore code genera identificativi di messaggi univoci come necessario, ma il valore del campo MDMID in MQMD non viene modificato al ritorno dalla chiamata, anche se è stato specificato MINONE o PMNMID. Se l'applicazione deve conoscere gli identificatori dei messaggi generati dal gestore code, deve fornire i record MQPMR contenenti il campo PRMID .

L'applicazione di invio può anche specificare un valore particolare per l'identificativo del messaggio, diverso da MINONE; questo arresta il gestore code che genera un identificativo del messaggio univoco. Un'applicazione che inoltra un messaggio può utilizzare questa funzionalità per propagare l'identificativo del messaggio originale.

Il gestore code non utilizza questo campo se non per:

- Genera un valore univoco se richiesto, come descritto in precedenza
- Consegna il valore all'applicazione che emette la richiesta get per il messaggio
- Copiare il valore nel campo MDCID di qualsiasi messaggio di report generato su questo messaggio (in base alle opzioni MDREP)

Quando il gestore code o un agent del canale messaggi genera un messaggio di report, imposta il campo MDMID nel modo specificato dal campo MDREP del messaggio originale, RONMI o ROPMI. Anche le applicazioni che generano messaggi di report devono eseguire questa operazione.

Per la chiamata MQGET, MDMID è uno dei cinque campi che è possibile utilizzare per selezionare un determinato messaggio da richiamare dalla coda. Di solito, la chiamata MQGET restituisce il messaggio successivo sulla coda, ma se è richiesto un determinato messaggio, è possibile ottenere tale risultato specificando uno o più dei cinque criteri di selezione, in qualsiasi combinazione; questi campi sono:

- MDMID
- MDCID
- MDGID
- MDSEQ
- MDOFF

L'applicazione imposta uno o più di questi campi sui valori richiesti, quindi imposta le opzioni di corrispondenza MO* corrispondenti nel campo GMMO in MQGMO per indicare che tali campi devono essere utilizzati come criteri di selezione. Solo i messaggi con i valori specificati in tali campi sono candidati per il richiamo. Il valore predefinito per il campo GMMO (se non modificato dall'applicazione) corrisponde sia all'identificativo del messaggio che all'identificativo di correlazione.

Di solito, il messaggio restituito è il primo nella coda che soddisfa i criteri di selezione. Ma se è specificato GMBRWN, il messaggio restituito è il messaggio successivo che soddisfa i criteri di selezione; la scansione di questo messaggio inizia con il messaggio che segue la posizione corrente del cursore.

Nota: La coda viene sottoposta a scansione in modo sequenziale per un messaggio che soddisfa i criteri di selezione, quindi i tempi di richiamo saranno più lenti rispetto a quando non viene specificato alcun criterio di selezione, soprattutto se è necessario eseguire la scansione di molti messaggi prima che ne venga trovato uno adatto.

Consultare la Tabella 1 per ulteriori informazioni su come vengono utilizzati i criteri di selezione in varie situazioni.

Specificare MINONE come identificativo del messaggio ha lo stesso effetto di non specificare MOMSGI, ovvero, qualsiasi identificativo del messaggio corrisponderà.

Questo campo viene ignorato se l'opzione GMMUC viene specificata nel parametro **GMO** sulla chiamata MQGET.

Alla restituzione da una chiamata MQGET, il campo MDMID viene impostato sull'identificativo del messaggio restituito (se presente).

È possibile utilizzare il seguente valore speciale:

MINONE
Nessun identificativo di messaggio specificato.

Il valore è zero binario per la lunghezza del campo.

Questo è un campo di input / output per le chiamate MQGET, MQPUT e MQPUT1 . La lunghezza di questo campo è data da LNMID. Il valore iniziale di questo campo è MINONE.

MDMT (numero intero con segno a 10 cifre)

Il tipo di messaggio.

Indica il tipo di messaggio. I tipi di messaggio sono raggruppati come segue:

MTSFST
Il valore più basso per i tipi di messaggio definiti dal sistema.

MTSLST
Il valore più alto per i tipi di messaggi definiti dal sistema.

I seguenti valori sono attualmente definiti nell'intervallo di sistema:

MTDGRM
Il messaggio non richiede una risposta.

Il messaggio non richiede una risposta.

MTRQST
Messaggio che richiede una risposta.

Il messaggio richiede una risposta.

Il nome della coda a cui inviare la risposta deve essere specificato nel campo MDRQ . Il campo MDREP indica come devono essere impostati MDMID e MDCID della risposta.

MTRPLA
Rispondere a un precedente messaggio di richiesta.

Il messaggio è la risposta ad un messaggio di richiesta precedente (MTRQST). Il messaggio deve essere inviato alla coda indicata dal campo MDRQ del messaggio di richiesta. Il campo MDREP della richiesta deve essere utilizzato per controllare la modalità di impostazione di MDMID e MDCID della risposta.

Nota: Il gestore code non applica la relazione richiesta - risposta; questa è una responsabilità dell'applicazione.

MTRPRT
Messaggio di prospetto.

Il messaggio riporta alcune ricorrenze previste o impreviste, di solito correlate ad altri messaggi (ad esempio, è stato ricevuto un messaggio di richiesta che conteneva dati non validi). Il messaggio deve essere inviato alla coda indicata dal campo MDRQ del descrittore del messaggio originale. Il campo MDFB deve essere impostato per indicare la natura del prospetto. Il campo MDREP del messaggio originale può essere utilizzato per controllare la modalità di impostazione di MDMID e MDCID del messaggio di report.

I messaggi di report generati dal gestore code o dall'agent del canale dei messaggi vengono sempre inviati alla coda MDRQ , con i campi MDFB e MDCID impostati come descritto in precedenza.

Altri valori all'interno dell'intervallo di sistema possono essere definiti nelle versioni future di MQI e sono accettati senza errori dalle chiamate MQPUT e MQPUT1 .

Possono essere utilizzati anche valori definiti dall'applicazione. Devono essere compresi nel seguente intervallo:

MTAFST
Valore più basso per i tipi di messaggio definiti dall'applicazione.

MTALST
Il valore più alto per i tipi di messaggio definiti dall'applicazione.

Per le chiamate MQPUT e MQPUT1 , il valore MDMT deve essere compreso nell'intervallo definito dal sistema o nell'intervallo definito dall'applicazione; in caso contrario, la chiamata ha esito negativo con codice di errore RC2029.

Questo è un campo di output per la chiamata MQGET e un campo di immissione per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è MTDGRM.

MDOFF (numero intero con segno a 10 cifre)

L'offset dei dati nel messaggio fisico dall'inizio del messaggio logico.

Questo è l'offset in byte dei dati nel messaggio fisico dall'inizio del messaggio logico di cui fanno parte i dati. Questi dati sono denominati *segmento*. L'offset è compreso tra 0 e 999 999 999. Un messaggio fisico che non è un segmento di un messaggio logico ha uno scostamento di zero.

Questo campo non deve essere impostato dall'applicazione nella chiamata MQPUT o MQGET se:

- Nella chiamata MQPUT, viene specificato PMLOGO.
- Nella chiamata MQGET, MOOFFS non è specificato.

Questi sono i modi consigliati per utilizzare queste chiamate per i messaggi che non sono messaggi di report. Tuttavia, se l'applicazione non è conforme a queste condizioni o se la chiamata è MQPUT1, l'applicazione deve garantire che MDOFF sia impostato su un valore appropriato.

All'input delle chiamate MQPUT e MQPUT1 , il gestore code utilizza il valore dettagliato nella Tabella 1. All'output delle chiamate MQPUT e MQPUT1 , il gestore code imposta questo campo sul valore che è stato inviato con il messaggio.

Per un messaggio di report che riporta un segmento di un messaggio logico, il campo MDOLN (purché non sia OLUNDF) viene utilizzato per aggiornare l'offset nelle informazioni del segmento conservate dal gestore code.

In fase di input della chiamata MQGET, il gestore code utilizza il valore descritto nella Tabella 1. Nell'output della chiamata MQGET, il gestore code imposta questo campo sul valore per il messaggio richiamato.

Il valore iniziale di questo campo è zero. Questo campo viene ignorato se MDVER è inferiore a MDVER2.

MDOLN (numero intero con segno a 10 cifre)

Lunghezza del messaggio originale.

Questo campo è rilevante solo per i messaggi di prospetto che sono segmenti. Specifica la lunghezza del segmento del messaggio a cui si riferisce il messaggio di report; non specifica la lunghezza del messaggio logico di cui fa parte il segmento, né la lunghezza dei dati nel messaggio di report.

Nota: Quando si genera un messaggio di report per un messaggio che è un segmento, il gestore code e l'agent del canale dei messaggi copiano in MQMD per il messaggio di report i campi MDGID, MDSEQ, MDOFFe *MDMFL*, dal messaggio originale. Di conseguenza, il messaggio del report è anche un segmento. Le applicazioni che generano messaggi di report sono consigliate per fare lo stesso e per assicurarsi che il campo MDOLN sia impostato correttamente.

Viene definito il seguente valore speciale:

OLUNDF
Lunghezza originale del messaggio non definita.

MDOLN è un campo di input nelle chiamate MQPUT e MQPUT1 , ma il valore fornito dall'applicazione è accettato solo in particolari circostanze:

- Se il messaggio inserito è un segmento ed è anche un messaggio di report, il gestore code accetta il valore specificato. Il valore deve essere:
  - Maggiore di zero se il segmento non è l'ultimo segmento
  - Non inferiore a zero se il segmento è l'ultimo segmento
  - Non inferiore alla lunghezza dei dati presenti nel messaggio

  Se queste condizioni non vengono soddisfatte, la chiamata ha esito negativo con codice di errore RC2252.
- Se il messaggio inserito è un segmento ma non un messaggio di report, il gestore code ignora il campo e utilizza la lunghezza dei dati del messaggio dell'applicazione.
- In tutti gli altri casi, il gestore code ignora il campo e utilizza invece il valore OLUNDF.

Questo è un campo di output sulla chiamata MQGET.

Il valore iniziale di questo campo è OLUNDF. Questo campo viene ignorato se MDVER è inferiore a MDVER2.

MDPAN (stringa di caratteri a 28 byte)

Nome dell'applicazione che inserisce il messaggio.

Questo fa parte del *contesto di origine* del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Il formato di MDPAN dipende dal valore di MDPAT.

Quando questo campo viene impostato dal gestore code (ossia, per tutte le opzioni tranne PMSETA), viene impostato sul valore determinato dall'ambiente:

- z/OS Su z/OS, il gestore code utilizza:
  - Per il batch z/OS , il nome lavoro di 8 caratteri dalla scheda JES JOB
  - Per TSO, l'identificativo utente TSO di 7 caratteri
  - Per CICS, l'applid di 8 caratteri, seguito dal tranid di 4 caratteri
  - Per IMS, l'identificativo di sistema IMS di 8 caratteri, seguito dal nome PSB di 8 caratteri
  - Per XCF, il nome del gruppo XCF di 8 caratteri, seguito dal nome del membro XCF di 16 caratteri
  - Per un messaggio generato da un gestore code, i primi 28 caratteri del nome del gestore code
  - Per l'accodamento distribuito senza CICS, il nome lavoro di 8 caratteri dell'iniziatore di canali, seguito dal nome di 8 caratteri del modulo che viene inserito nella coda di messaggi non recapitabili, seguito dall'identificativo di un'attività di 8 caratteri.
  - Per l'elaborazione dei binding di lingua MQSeries Java con IBM MQ for z/OS il nome lavoro di 8 caratteri dello spazio di indirizzo creato per l'ambiente z/OS UNIX System Services . In genere, si tratta di un identificativo utente TSO con un singolo carattere numerico aggiunto.

  Il nome o i nomi vengono riempiti a destra con spazi vuoti, come qualsiasi spazio nel resto del campo. Se è presente più di un nome, non vi è alcun separatore tra di essi.
- Windows Su sistemi PC DOS e Windows , il gestore code utilizza:
  - Per un'applicazione CICS , il nome della transazione CICS
  - Per un'applicazione nonCICS , i 28 caratteri più a destra del nome completo dell'eseguibile
- IBM i Su IBM i, il gestore code utilizza il nome lavoro completo.
- Linux AIX Su AIX and Linux, il gestore code utilizza:
  - Per un'applicazione CICS , il nome della transazione CICS

  - Per un'applicazione nonCICS , i 14 caratteri più a destra del nome completo dell'eseguibile, se questo è disponibile per il gestore code, e gli altri spazi (ad esempio, su AIX)
- Su VSE/ESA, il gestore code utilizza l'applid di 8 caratteri, seguito dal tranid di 4 caratteri.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se PMSETA è specificato nel parametro **PMO** . Tutte le informazioni che seguono un carattere null all'interno del campo vengono scartate. Il carattere null e i seguenti caratteri vengono convertiti in spazi dal gestore code. Se PMSETA non viene specificato, questo campo viene ignorato all'immissione ed è un campo di sola emissione.

Questo è un campo di output per la chiamata MQGET. La lunghezza di questo campo è data da LNPAN. Il valore iniziale di questo campo è di 28 caratteri vuoti.

MDPAT (numero intero con segno a 10 cifre)

Tipo di applicazione che inserisce il messaggio.

Questo fa parte del **contesto di origine** del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

MDPAT può avere uno dei seguenti tipi standard. Anche i tipi definiti dall'utente possono essere utilizzati, ma devono essere limitati ai valori compresi tra ATUFST e ATULST.

AAIX
AIX (stesso valore di ATUNIX).

ATBRKR
Broker.

ATCICS
Transazione CICS .

ATCICB
CICS bridge.

ATVSE
Transazione CICS/VSE .

ATDOS
Applicazione IBM MQ MQI client su PC DOS.

Gestore code ATD
Agent gestore code distribuito.

ATGUAR
Applicazione Tandem Guardian (stesso valore di ATNSK).

ATIMS
Applicazione IMS .

ATIMSB
Bridge IMS .

ATJAVA
Java.

AMVS
Applicazione MVS o TSO (stesso valore di ATZOS).

NOTA
Lotus Notes Applicazione agent.

ATNSK
Applicazione kernel NonStop tandem.

AT390
Applicazione OS/390 (stesso valore di ATZOS).

AT400
Applicazione IBM i .

ATQM
Gestore code.

ATUNIX
Applicazione UNIX .

ATVOS
Applicazione VOS Stratus.

ATWIN
Applicazione Windows a 16 bit.

ATWINT
Applicazione Windows a 32 bit.

ATXCF
XCF.

ATZOS
Applicazione z/OS .

ATDEF
Il tipo di applicazione predefinita.

Questo è il tipo di applicazione predefinito per la piattaforma su cui è in esecuzione l'applicazione.

Nota: Il valore di questa costante è specifico dell'ambiente.

ATUNK
Tipo di applicazione sconosciuto.

Questo valore può essere utilizzato per indicare che il tipo di applicazione è sconosciuto, anche se sono presenti altre informazioni di contesto.

ATUFST
Il valore più basso per il tipo di applicazione definito dall'utente.

ATULST
Il valore più alto per il tipo di applicazione definito dall'utente.

Può verificarsi anche il seguente valore speciale:

ATNCON
Non sono presenti informazioni di contesto nel messaggio.

Questo valore viene impostato dal gestore code quando un messaggio viene inserito senza contesto (ovvero, viene specificata l'opzione di contesto PMNOC).

Quando viene richiamato un messaggio, è possibile verificare MDPAT per questo valore per decidere se il messaggio ha un contesto (si consiglia che MDPAT non sia mai impostato su ATNCON, da un'applicazione che utilizza PMSETA, se uno qualsiasi degli altri campi di contesto non è vuoto).

ATSIB
Indica un messaggio originato in un altro prodotto di messaggistica IBM MQ e arrivato tramite il bridge SIB (Service Integration Bus).

Quando il gestore code genera queste informazioni come risultato di un inserimento di un'applicazione, il campo viene impostato su un valore determinato dall'ambiente.

Tenere presente che su IBM i, il campo è impostato su AT400; il gestore code non utilizza mai ATCICS su IBM i.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se PMSETA è specificato nel parametro **PMO** . Se PMSETA non viene specificato, questo campo viene ignorato all'immissione ed è un campo di sola emissione.

Dopo il corretto completamento di una chiamata MQPUT o MQPUT1 , questo campo contiene il MDPAT che è stato trasmesso con il messaggio se è stato inserito in una coda. Questo sarà il

valore di MDPAT che viene conservato con il messaggio se viene conservato (vedere la descrizione di PMRET per ulteriori dettagli sulle pubblicazioni conservate) ma non viene utilizzato come MDPAT quando il messaggio viene inviato come pubblicazione ai sottoscrittori poiché forniscono un valore da sovrascrivere MDPAT in tutte le pubblicazioni a loro inviate. Se il messaggio non ha alcun contesto, il campo è impostato su ATNCON.

Questo è un campo di output per la chiamata MQGET. Il valore iniziale di questo campo è ATNCON.

MDPD (stringa di caratteri a 8 byte)

Data in cui è stato inserito il messaggio.

Questo fa parte del *contesto di origine* del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Il formato utilizzato per la data in cui questo campo viene generato dal gestore code è:

- AAAAMMGG

dove i caratteri rappresentano:

AAAA
anno (quattro cifre)

MI
mese dell'anno (da 01 a 12)

GG
giorno del mese (da 01 a 31)

GMT (Greenwich Mean Time) viene utilizzato per i campi MDPD e MDPT , in base all'orologio di sistema impostato in modo accurato su GMT.

Se il messaggio è stato inserito come parte di un'unità di lavoro, la data è quella in cui è stato immesso il messaggio e non la data in cui è stato eseguito il commit dell'unità di lavoro.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se PMSETA è specificato nel parametro **PMO** . Il contenuto del campo non viene controllato dal gestore code, ad eccezione del fatto che tutte le informazioni che seguono un carattere null all'interno del campo vengono eliminate. Il carattere null e i seguenti caratteri vengono convertiti in spazi dal gestore code. Se PMSETA non viene specificato, questo campo viene ignorato all'immissione ed è un campo di sola emissione.

Dopo il corretto completamento di una chiamata MQPUT o MQPUT1 , questo campo contiene il MDPD che è stato trasmesso con il messaggio se è stato inserito in una coda. Questo sarà il valore di MDPD che viene conservato con il messaggio se viene conservato (vedere la descrizione di PMRET per ulteriori dettagli sulle pubblicazioni conservate) ma non viene utilizzato come MDPD quando il messaggio viene inviato come pubblicazione ai sottoscrittori poiché forniscono un valore da sovrascrivere MDPD in tutte le pubblicazioni a loro inviate. Se il messaggio non ha contesto, il campo è completamente vuoto.

Questo è un campo di output per la chiamata MQGET. La lunghezza di questo campo è fornita da LNPDAT. Il valore iniziale di questo campo è di 8 caratteri vuoti.

MDPER (numero intero con segno a 10 cifre)

Durata del messaggio.

Indica se il messaggio sopravvive agli errori di sistema e riavvia il gestore code. Per le chiamate MQPUT e MQPUT1 , il valore deve essere uno dei seguenti:

PEPER
Il messaggio è persistente.

Ciò significa che il messaggio sopravvive agli errori di sistema e ai riavvii del gestore code. Una volta che il messaggio è stato inserito e l'unità di lavoro del putter è stata sottoposta a commit (se il messaggio è inserito come parte di un'unità di lavoro), il messaggio viene conservato nella memoria ausiliaria. Rimane lì fino a quando il messaggio non viene rimosso dalla coda e viene

eseguito il commit dell'unità di lavoro del getter (se il messaggio viene richiamato come parte di un'unità di lavoro).

Quando un messaggio persistente viene inviato ad una coda remota, viene utilizzato un meccanismo di memorizzazione e inoltro per conservare il messaggio su ciascun gestore code lungo l'instradamento alla destinazione, fino a quando non si sa che il messaggio è arrivato al gestore code successivo.

I messaggi persistenti non possono essere posizionati su:

- Code dinamiche temporanee
- Code condivise in cui il livello della struttura CFS è inferiore a tre o la struttura CFS non è recuperabile.

I messaggi persistenti possono essere inseriti in code dinamiche permanenti, code predefinite e code condivise in cui il livello della struttura CFS è 3 e la CFS è recuperabile.

PENPER
Il messaggio non è persistente.

Ciò significa che il messaggio di solito non sopravvive agli errori di sistema o ai riavvii del gestore code. Ciò si applica anche se viene trovata una copia intatta del messaggio nella memoria ausiliaria durante il riavvio del gestore code.

Nel caso speciale di code condivise, i messaggi non persistenti *sopravvivono* ai riavvii dei gestori code nel gruppo di condivisione code, ma non sopravvivono agli errori della CF utilizzata per memorizzare i messaggi nelle code condivise.

PEQDEF
Il messaggio ha la persistenza predefinita.

- Se la coda è una coda cluster, la persistenza del messaggio viene presa dall'attributo **DefPersistence** definito sul gestore code di destinazione che possiede la particolare istanza della coda su cui è collocato il messaggio. Di solito, tutte le istanze di una coda cluster hanno lo stesso valore per l'attributo **DefPersistence** , anche se non è obbligatorio.

  Il valore di **DefPersistence** viene copiato nel campo *MDPER* quando il messaggio viene inserito nella coda di destinazione. Se **DefPersistence** viene modificato successivamente, i messaggi che sono già stati inseriti nella coda non vengono influenzati.
- Se la coda non è una coda cluster, la persistenza del messaggio viene ricavata dall'attributo **DefPersistence** definito nel gestore code locale, anche se il gestore code di destinazione è remoto.

  Se è presente più di una definizione nel percorso di risoluzione del nome della coda, la persistenza predefinita viene presa dal valore di questo attributo nella prima definizione nel percorso. È possibile che si tratti di:

  – Una coda alias
  – Una coda locale
  – Una definizione locale di una coda remota
  – Un alias del gestore code
  – Una coda di trasmissione (ad esempio, la coda `DefXmitQName`)

  Il valore **DefPersistence** viene copiato nel campo `MDPER` quando il messaggio viene inserito. Se **DefPersistence** viene modificato successivamente, i messaggi già inseriti non vengono influenzati.

Sia i messaggi persistenti che quelli non persistenti possono esistere nella stessa coda.

Quando si risponde a un messaggio, le applicazioni normalmente utilizzano per il messaggio di risposta la persistenza del messaggio di richiesta.

Per una chiamata MQGET, il valore restituito è PEPER o PENPER.

Questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è PEQDEF.

MDPRI (numero intero con segno a 10 cifre)

Priorità del messaggio.

Per le chiamate MQPUT e MQPUT1 , il valore deve essere maggiore o uguale a zero; zero è la priorità più bassa. È possibile utilizzare anche il valore speciale seguente:

PRQDEF
Priorità predefinita per la coda.

- Se la coda è una coda cluster, la priorità per il messaggio viene presa dall'attributo **DefPriority** come definito nel gestore code di destinazione che possiede la particolare istanza della coda in cui è posizionato il messaggio. Di solito, tutte le istanze di una coda cluster hanno lo stesso valore per l'attributo **DefPriority** , anche se non è obbligatorio.

  Il valore di **DefPriority** viene copiato nel campo MDPRI quando il messaggio viene inserito nella coda di destinazione. Se **DefPriority** viene modificato successivamente, i messaggi che sono già stati inseriti nella coda non vengono influenzati.
- Se la coda non è una coda cluster, la priorità per il messaggio viene presa dall'attributo **DefPriority** come definito nel gestore code locale, anche se il gestore code di destinazione è remoto.

  Se esiste più di una definizione nel percorso di risoluzione del nome della coda, la priorità predefinita viene presa dal valore di questo attributo nella prima definizione nel percorso. È possibile che si tratti di:

  – Una coda alias
  – Una coda locale
  – Una definizione locale di una coda remota
  – Un alias del gestore code
  – Una coda di trasmissione (ad esempio, la coda DefXmitQName)

  Il valore **DefPriority** viene copiato nel campo MDPRI quando il messaggio viene inserito. Se **DefPriority** viene modificato successivamente, i messaggi già inseriti non vengono influenzati.

Il valore restituito dalla chiamata MQGET è sempre maggiore o uguale a zero; il valore PRQDEF non viene mai restituito.

Se un messaggio viene inserito con una priorità superiore a quella massima supportata dal gestore code locale (questo valore massimo viene fornito dall'attributo del gestore code **MaxPriority**), il messaggio viene accettato dal gestore code, ma viene inserito nella coda con la priorità massima del gestore code; la chiamata MQPUT o MQPUT1 viene completata con CCWARN e codice motivo RC2049. Tuttavia, il campo MDPRI conserva il valore specificato dall'applicazione che ha inserito il messaggio.

Quando si risponde ad un messaggio, le applicazioni normalmente utilizzano per il messaggio di risposta la priorità del messaggio di richiesta. In altre situazioni, specificando PRQDEF è possibile eseguire l'ottimizzazione della priorità senza modificare l'applicazione.

Questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è PRQDEF.

MDPT (stringa di caratteri a 8 byte)

L'ora in cui è stato inserito il messaggio.

Questo fa parte del **contesto di origine** del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Il formato utilizzato per l'ora in cui questo campo viene generato dal gestore code è:

- HHMMSSTH

dove i caratteri rappresentano (in ordine):

OO
ore (da 00 a 23)

MI
minuti (da 00 a 59)

SS
secondi (da 00 a 59; consultare note)

T
decimi di secondo (da 0 a 9)

H
centesimi di secondo (da 0 a 9)

Nota: Se l'orologio di sistema è sincronizzato con uno standard di tempo molto accurato, è possibile in rare occasioni che vengano restituiti 60 o 61 per i secondi in MDPT. Ciò si verifica quando i secondi bisestili vengono inseriti nello standard temporale globale.

GMT (Greenwich Mean Time) viene utilizzato per i campi MDPD e MDPT , in base all'orologio di sistema impostato in modo accurato su GMT.

Se il messaggio è stato inserito come parte di un'unità di lavoro, l'ora è quella in cui è stato immesso il messaggio e non quella in cui è stato eseguito il commit dell'unità di lavoro.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se PMSETA è specificato nel parametro **PMO** . Il contenuto del campo non viene controllato dal gestore code, ad eccezione del fatto che tutte le informazioni che seguono un carattere null all'interno del campo vengono eliminate. Il carattere null e i seguenti caratteri vengono convertiti in spazi dal gestore code. Se PMSETA non viene specificato, questo campo viene ignorato all'immissione ed è un campo di sola emissione.

Dopo il corretto completamento di una chiamata MQPUT o MQPUT1 , questo campo contiene il valore MDPT che è stato trasmesso con il messaggio se è stato inserito in una coda. Questo sarà il valore di MDPT che viene conservato con il messaggio se viene conservato (vedere la descrizione di PMRET per ulteriori dettagli sulle pubblicazioni conservate) ma non viene utilizzato come MDPT quando il messaggio viene inviato come pubblicazione ai sottoscrittori poiché forniscono un valore da sovrascrivere MDPT in tutte le pubblicazioni a loro inviate. Se il messaggio non ha contesto, il campo è completamente vuoto.

Questo è un campo di output per la chiamata MQGET. La lunghezza di questo campo è data da LNPTIM. Il valore iniziale di questo campo è di 8 caratteri vuoti.

MDREP (numero intero con segno a 10 cifre)

Opzioni per i messaggi di report.

Un messaggio di report è un messaggio relativo ad un altro messaggio, utilizzato per informare un'applicazione di eventi previsti o imprevisti correlati al messaggio originale. Il campo MDREP consente all'applicazione che invia il messaggio originale di specificare quali messaggi di report sono richiesti, se i dati del messaggio dell'applicazione devono essere inclusi in essi e (sia per i report che per le risposte) come devono essere impostati gli identificativi del messaggio e della correlazione nel report o nel messaggio di risposta. È possibile richiedere uno o tutti (o nessuno) dei seguenti tipi di messaggi di report:

- Eccezione
- Scadenza
- Conferma all'arrivo (COA)
- Conferma alla consegna (COD)
- PAN (positive action notification)
- Notifica azione negativa (NAN)

Se è richiesto più di un tipo di messaggio di prospetto o sono necessarie altre opzioni di prospetto, i valori possono essere aggiunti insieme (non aggiungere la stessa costante più di una volta).

L'applicazione che riceve il messaggio di report può determinare il motivo per cui il report è stato generato esaminando il campo MDFB in MQMD; consultare il campo MDFB per ulteriori dettagli.

L'utilizzo delle opzioni del report quando si mette un messaggio in un argomento può causare la generazione e l'invio all'applicazione di zero, uno o più messaggi di report. Ciò è dovuto al fatto che il messaggio di pubblicazione può essere inviato a zero, una o più applicazioni di sottoscrizione.

Opzioni di eccezione: è possibile specificare una delle seguenti opzioni per richiedere un messaggio di report di eccezione.

ROATTIVITA '

Report di attività richiesti

Questa opzione del report consente di generare un report di attività, ogni volta che un messaggio con questa serie di opzioni del report viene elaborato dalle applicazioni di supporto.

I messaggi con questa serie di opzioni di report devono essere accettati da qualsiasi gestore code, anche se non 'comprendono' l'opzione. Ciò consente di impostare l'opzione di report su qualsiasi messaggio utente, anche se sono elaborati da gestori code precedenti. Per ottenere ciò, l'opzione di prospetto viene inserita nel sottocampo ROAUM.

Se un processo (un gestore code o un processo utente) esegue un'attività su un messaggio con ROACT impostato, può scegliere di generare e inserire un report di attività.

L'opzione del report di attività consente di tracciare l'instradamento di qualsiasi messaggio in una rete di gestori code. L'opzione di report può essere specificata su qualsiasi messaggio utente corrente e immediatamente può iniziare a calcolare l'instradamento del messaggio attraverso la rete. Se l'applicazione che genera il messaggio non può abilitare la creazione del report di attività, può essere abilitata utilizzando un'uscita incrociata API fornita dagli amministratori del gestore code.

Ai report di attività sono applicabili diverse condizioni:

1. L'instradamento sarà meno dettagliato se nella rete sono presenti meno gestori code in grado di generare report di attività.
2. I report di attività potrebbero non essere facilmente "ordinabili" al fine di determinare il percorso intrapreso.
3. I report di attività potrebbero non essere in grado di trovare un instradamento alla destinazione richiesta.

ROEXC

Report di eccezione richiesti.

Questo tipo di report può essere generato da un agent del canale dei messaggi quando un messaggio viene inviato a un altro gestore code e il messaggio non può essere consegnato alla coda di destinazione specificata. Ad esempio, la coda di destinazione o una coda di trasmissione intermedia potrebbe essere piena oppure il messaggio potrebbe essere troppo grande per la coda.

La creazione del messaggio di report di eccezioni dipende dalla persistenza del messaggio originale e dalla velocità del canale del messaggio (normale o veloce) attraverso cui il messaggio originale viaggia:

- Per tutti i messaggi persistenti e per i messaggi non persistenti che viaggiano attraverso i normali canali di messaggi, il report di eccezione viene generato solo se l'azione specificata dall'applicazione mittente per la condizione di errore può essere completata correttamente. L'applicazione mittente può specificare una delle seguenti azioni per controllare la disposizione del messaggio originale quando si verifica la condizione di errore:
  - RODLQ (ciò fa sì che il messaggio originale venga inserito nella coda di messaggi non recapitabili).
  - RODISC (questo causa l'eliminazione del messaggio originale).

Se l'azione specificata dall'applicazione mittente non può essere completata correttamente, il messaggio originale viene lasciato nella coda di trasmissione e non viene generato alcun messaggio di report di eccezione.

- Per i messaggi non persistenti che viaggiano attraverso i canali di messaggi veloci, il messaggio originale viene rimosso dalla coda di trasmissione e il report di eccezione viene generato anche se l'azione specificata per la condizione di errore non può essere completata correttamente. Ad esempio, se viene specificato RODLQ, ma il messaggio originale non può essere inserito nella coda di messaggi non recapitabili poiché (ad esempio) la coda è piena, viene generato il messaggio di report dell'eccezione e il messaggio originale viene eliminato.

  Consultare Persistenza dei messaggi per ulteriori informazioni sui canali di messaggi normali e veloci.

Non viene generato un report di eccezione se l'applicazione che ha inserito il messaggio originale può ricevere una notifica sincrona del problema mediante il codice motivo restituito dalla chiamata MQPUT o MQPUT1 .

Le applicazioni possono anche inviare report di eccezione, per indicare che un messaggio che ha ricevuto non può essere elaborato (ad esempio, perché si tratta di una transazione di addebito che causerebbe il superamento del limite di credito del conto).

I dati del messaggio originale non sono inclusi nel messaggio di report.

Non specificare più di uno tra ROEXC, ROEXCD e ROEXCF.

ROEXCD
Report di eccezioni con dati richiesti.

È uguale a ROEXC, tranne che i primi 100 byte dei dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio del report. Se il messaggio originale contiene una o più strutture di intestazione MQ , vengono incluse nel messaggio di report, in aggiunta ai 100 byte dei dati dell'applicazione.

Non specificare più di uno tra ROEXC, ROEXCD e ROEXCF.

ROEXCF
Sono richiesti report di eccezione con dati completi.

È uguale a ROEXC, tranne per il fatto che tutti i dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio del prospetto.

Non specificare più di uno tra ROEXC, ROEXCD e ROEXCF.

Opzioni di scadenza: è possibile specificare una delle seguenti opzioni per richiedere un messaggio di report di scadenza.

ROEXP
Report di scadenza richiesti.

Questo tipo di report viene generato dal gestore code se il messaggio viene eliminato prima del recapito a un'applicazione perché è trascorso il tempo di scadenza (consultare il campo MDEXP). Se questa opzione non è impostata, non viene generato alcun messaggio di report se un messaggio viene eliminato per questo motivo (anche se viene specificata una delle opzioni ROEXC*).

I dati del messaggio originale non sono inclusi nel messaggio di report.

Non specificare più di uno tra ROEXP, ROEXPD e ROEXPF.

ROEXPD
Report di scadenza con dati richiesti.

È uguale a ROEXP, ma i primi 100 byte dei dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio del report. Se il messaggio originale contiene una o più strutture di intestazione MQ , vengono incluse nel messaggio di report, in aggiunta ai 100 byte dei dati dell'applicazione.

Non specificare più di uno tra ROEXP, ROEXPD e ROEXPF.

ROEXPF
Sono richiesti report di scadenza con dati completi.

È uguale a ROEXP, tranne che tutti i dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio di report.

Non specificare più di uno tra ROEXP, ROEXPD e ROEXPF.

Opzioni di conferma all'arrivo: è possibile specificare una delle seguenti opzioni per richiedere un messaggio di report di conferma all'arrivo.

ROCOA
Sono richiesti report di conferma all'arrivo.

Questo tipo di report viene generato dal gestore code proprietario della coda di destinazione, quando il messaggio viene inserito nella coda di destinazione. I dati del messaggio originale non sono inclusi nel messaggio di report.

Se il messaggio è inserito come parte di un'unità di lavoro e la coda di destinazione è una coda locale, il messaggio di report COA generato dal gestore code diventa disponibile per il richiamo solo se e quando viene eseguito il commit dell'unità di lavoro.

Un report COA non viene generato se il campo MDFMT nel descrizione del messaggio è FMXQH o FMDLH. Ciò impedisce la creazione di un report COA se il messaggio viene inserito in una coda di trasmissione o non è distribuibile e viene inserito in una coda di messaggi non recapitabili.

Non specificare più di uno tra ROCOA, ROCOAD e ROCOAF.

ROCOAD
Report di conferma all'arrivo con dati richiesti.

Questo è lo stesso di ROCOA, tranne che i primi 100 byte dei dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio del report. Se il messaggio originale contiene una o più strutture di intestazione MQ , vengono incluse nel messaggio di report, in aggiunta ai 100 byte dei dati dell'applicazione.

Non specificare più di uno tra ROCOA, ROCOAD e ROCOAF.

ROCOAF
Report di conferma all'arrivo con dati completi richiesti.

È lo stesso di ROCOA, tranne per il fatto che tutti i dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio del prospetto.

Non specificare più di uno tra ROCOA, ROCOAD e ROCOAF.

Opzioni di eliminazione e scadenza: è possibile specificare la seguente opzione per impostare l'ora di scadenza e l'indicatore di eliminazione per i messaggi di report.

ROPDA
Impostare la scadenza del messaggio di report e l'indicatore di eliminazione.

Questa opzione garantisce che i messaggi di report e i messaggi di risposta ereditino l'ora di scadenza e l'indicatore di eliminazione (da eliminare o meno) dai messaggi originali. Con questa serie di opzioni, notificare e rispondere ai messaggi:

1. Eredita l'indicatore RODISC (se è stato impostato).
2. Eredita il tempo di scadenza rimanente del messaggio, se il messaggio non è un report di scadenza. Se il messaggio è un report di scadenza, il tempo di scadenza è impostato su 60 secondi.

Con questa opzione impostata, si applica quanto segue:

Nota:

1. I messaggi di report e di risposta vengono generati con un indicatore di eliminazione e un valore di scadenza e non possono rimanere nel sistema.
2. I messaggi di instradamento della traccia non possono raggiungere le code di destinazione sui gestori code non abilitati per l'instradamento della traccia.
3. Le code non vengono riempite con report che non possono essere consegnati, se i collegamenti di comunicazione sono interrotti.
4. Le risposte del server dei comandi ereditano la restante scadenza della richiesta.

Opzioni di conferma della consegna: è possibile specificare una delle seguenti opzioni per richiedere un messaggio di report di conferma della consegna.

ROCOD
Sono richiesti i report di conferma della consegna.

Questo tipo di report viene generato da un gestore code quando un'applicazione richiama il messaggio dalla coda di destinazione in modo che il messaggio venga eliminato dalla coda. I dati del messaggio originale non sono inclusi nel messaggio di report.

Se il messaggio viene richiamato come parte di un'unità di lavoro, il messaggio di report viene generato all'interno della stessa unità di lavoro, in modo che il report non sia disponibile fino a quando non viene eseguito il commit dell'unità di lavoro. Se viene eseguito il backout dell'unità di lavoro, il report non viene inviato.

Un report COD non viene generato se il campo MDFMT nel descrittore del messaggio è FMDLH. Ciò impedisce la creazione di un report COD se il messaggio non è distribuibile e viene inserito in una coda di messaggi non recapitabili.

ROCOD non è valido se la coda di destinazione è una coda XCF.

Non specificare più di uno tra ROCOD, ROCODD e ROCODF.

ARROTONDA
I report di conferma della consegna con i dati richiesti.

È uguale a ROCOD, ma i primi 100 byte dei dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio del prospetto. Se il messaggio originale contiene una o più strutture di intestazione MQ , vengono incluse nel messaggio di report, in aggiunta ai 100 byte dei dati dell'applicazione.

Se GMATM viene specificato nella chiamata MQGET per il messaggio originale e il messaggio richiamato viene troncato, la quantità di dati del messaggio dell'applicazione inseriti nel messaggio di report è il minimo di:

- La lunghezza del messaggio originale
- 100 byte.

ROCODD non è valido se la coda di destinazione è una coda XCF.

Non specificare più di uno tra ROCOD, ROCODD e ROCODF.

ROCODF
Sono richiesti report di conferma della consegna con dati completi.

È uguale a ROCOD, tranne per il fatto che tutti i dati del messaggio dell'applicazione dal messaggio originale sono inclusi nel messaggio del prospetto.

ROCODF non è valido se la coda di destinazione è una coda XCF.

Non specificare più di uno tra ROCOD, ROCODD e ROCODF.

Opzioni di notifica azione: è possibile specificare una o entrambe le seguenti opzioni per richiedere che l'applicazione ricevente invii un messaggio di report di azione positiva o negativa.

ROPAN
Sono richiesti i report di notifica delle azioni positive.

Questo tipo di report viene generato dall'applicazione che richiama il messaggio e agisce su di esso. Indica che l'azione richiesta nel messaggio è stata eseguita correttamente. L'applicazione che genera il report determina se i dati devono essere inclusi nel report.

Oltre a trasmettere questa richiesta all'applicazione che richiama il messaggio, il gestore code non esegue alcuna azione basata su questa opzione. È responsabilità dell'applicazione di richiamo generare il report, se appropriato.

RONAN
Sono richiesti report di notifica azione negativa.

Questo tipo di report viene generato dall'applicazione che richiama il messaggio e agisce su di esso. Indica che l'azione richiesta nel messaggio non è stata eseguita correttamente. L'applicazione che genera il report determina se i dati devono essere inclusi nel report. Ad esempio, potrebbe essere opportuno includere alcuni dati che indicano il motivo per cui non è stato possibile eseguire la richiesta.

Oltre a trasmettere questa richiesta all'applicazione che richiama il messaggio, il gestore code non esegue alcuna azione basata su questa opzione. È responsabilità dell'applicazione di richiamo generare il report, se appropriato.

La determinazione delle condizioni che corrispondono ad un'azione positiva e che corrispondono ad un'azione negativa è di competenza della domanda. Tuttavia, se la richiesta è stata eseguita solo parzialmente, si consiglia di generare un report NAN piuttosto che un report PAN, se richiesto. Si raccomanda inoltre che ogni possibile condizione corrisponda a un'azione positiva o a un'azione negativa, ma non a entrambe.

Opzioni identificativo messaggio: è possibile specificare una delle seguenti opzioni per controllare il modo in cui deve essere impostato il `MDMID` del messaggio di report (o del messaggio di risposta).

RONMI
Nuovo identificativo messaggio.

Questa è l'azione predefinita e indica che se viene generato un report o una risposta come risultato di questo messaggio, deve essere generato un nuovo `MDMID` per il report o il messaggio di risposta.

RAMMI
Inoltrare l'identificativo del messaggio.

Se viene generato un report o una risposta come risultato di questo messaggio, il `MDMID` di questo messaggio deve essere copiato nel `MDMID` del report o del messaggio di risposta.

Il `MsgId` di un messaggio di pubblicazione sarà diverso per ogni sottoscrittore che riceve una copia della pubblicazione e quindi il `MsgId` copiato nel report o nel messaggio di risposta sarà diverso per ogni sottoscrittore.

Se questa opzione non viene specificata, viene assunto RONMI.

Opzioni identificativo di correlazione: è possibile specificare una delle seguenti opzioni per controllare la modalità di impostazione del `MDCID` del messaggio di report (o del messaggio di risposta).

ROCMTC
Copiare l'identificativo del messaggio nell'identificativo di correlazione.

Questa è l'azione predefinita e indica che se un report o una risposta viene generato come risultato di questo messaggio, il `MDMID` di questo messaggio deve essere copiato nel `MDCID` del report o del messaggio di risposta.

Il `MsgId` di un messaggio di pubblicazione sarà diverso per ogni sottoscrittore che riceve una copia della pubblicazione e quindi il `MsgId` copiato in `CorrelId` del messaggio di report o di risposta sarà diverso per ciascuno di essi.

ROPCI
Passare l'identificativo di correlazione.

Se viene generato un report o una risposta come risultato di questo messaggio, il MDCID di questo messaggio deve essere copiato nel MDCID del report o del messaggio di risposta.

Il MDCID di un messaggio di pubblicazione sarà specifico per un sottoscrittore a meno che non utilizzi l'opzione SOSCID e imposti il campo SCDIC in MQSD su CINONE. Pertanto, è possibile che il MDCID copiato nel MDCID del messaggio di report o di risposta sia diverso per ciascuno di essi.

Se questa opzione non viene specificata, viene utilizzato ROCMTC.

I server che rispondevano alle richieste o che generavano messaggi di report sono consigliati per controllare se le opzioni ROPMI o ROPCI erano impostate nel messaggio originale. Se lo fossero, i server dovrebbero intraprendere l'azione descritta per tali opzioni. Se nessuno dei due è impostato, i server devono eseguire l'azione predefinita corrispondente.

: È possibile specificare una delle seguenti opzioni per controllare la disposizione del messaggio originale quando non può essere consegnato alla coda di destinazione. Queste opzioni si applicano solo a quelle situazioni che risulterebbero nella generazione di un messaggio di report di eccezione se ne fosse stato richiesto uno dall'applicazione mittente. L'applicazione può impostare le opzioni di disposizione indipendentemente dalla richiesta di report di eccezioni.

RODLQ
Inserimento messaggio nella coda dei messaggi non instradabili.

Questa è l'azione predefinita e indica che il messaggio deve essere collocato nella coda di messaggi non recapitabili, se il messaggio non può essere consegnato alla coda di destinazione. Ciò si verifica nelle situazioni seguenti:

- Quando l'applicazione che ha inserito il messaggio originale non può essere notificata in modo sincrono del problema mediante il codice motivo restituito dalla chiamata MQPUT o MQPUT1 . Viene generato un messaggio di report di eccezione, se richiesto dal mittente.
- Quando l'applicazione che ha inserito il messaggio originale stava inserendo un argomento

Verrà generato un messaggio di report di eccezione, se richiesto dal mittente.

RODISC
Eliminazione messaggio.

Ciò indica che il messaggio deve essere eliminato se non può essere consegnato alla coda di destinazione. Ciò si verifica nelle situazioni seguenti:

- Quando l'applicazione che ha inserito il messaggio originale non può essere notificata in modo sincrono del problema mediante il codice motivo restituito dalla chiamata MQPUT o MQPUT1 . Viene generato un messaggio di report di eccezione, se richiesto dal mittente.
- Quando l'applicazione che ha inserito il messaggio originale stava inserendo un argomento

Verrà generato un messaggio di report di eccezione, se richiesto dal mittente.

Se è necessario restituire il messaggio originale al mittente, senza che il messaggio originale venga inserito nella coda di messaggi non recapitabili, il mittente deve specificare RODISC con ROEXCF.

Opzione predefinita: è possibile specificare quanto segue se non è richiesta alcuna opzione di report:

RONONE
Nessun report richiesto.

Questo valore può essere utilizzato per indicare che non sono state specificate altre opzioni. RONONE è definito per aiutare la documentazione del programma. Non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Informazioni generali:

1. Tutti i tipi di report richiesti devono essere specificamente richiesti dall'applicazione che invia il messaggio originale. Ad esempio, se viene richiesto un report COA ma non un report di eccezione, viene generato un report COA quando il messaggio viene inserito nella coda di destinazione, ma

non viene generato alcun report di eccezione se la coda di destinazione è piena quando arriva il messaggio. Se non è impostata alcuna opzione MDREP , non viene generato alcun messaggio di report dal gestore code o dall'MCA (message channel agent).

Alcune opzioni di report possono essere specificate anche se il gestore code locale non le riconosce; ciò è utile quando l'opzione deve essere elaborata dal gestore code di destinazione. Consultare “Opzioni di report e indicatori di messaggi su IBM i” a pagina 1473 per maggiori dettagli.

Se viene richiesto un messaggio di report, il nome della coda a cui il report deve essere inviato deve essere specificato nel campo MDRQ . Quando viene ricevuto un messaggio di report, è possibile determinare la natura del report esaminando il campo MDFB nel descrittore del messaggio.

2. Se il gestore code o l'MCA che genera un messaggio di report non è in grado di inserire il messaggio di report nella coda di risposta (ad esempio, perché la coda di risposta o la coda di trasmissione è piena), il messaggio di report viene posizionato nella coda di messaggi non recapitabili. Se anche questo non riesce, o se non c'è una coda di messaggi non recapitabili, l'azione intrapresa dipende dal tipo di messaggio di report:

   - Se il messaggio di report è un report di eccezione, il messaggio che ha causato la creazione del report di eccezione viene lasciato nella relativa coda di trasmissione; ciò garantisce che il messaggio non venga perso.
   - Per tutti gli altri tipi di report, il messaggio di report viene eliminato e l'elaborazione continua normalmente. Questa operazione viene eseguita perché il messaggio originale è già stato consegnato in modo sicuro (per i messaggi di report COA o COD) o non è più di alcun interesse (per un messaggio di report di scadenza).

   Una volta che un messaggio di report è stato posizionato correttamente su una coda (la coda di destinazione o una coda di trasmissione intermedia), il messaggio non è più soggetto a un'elaborazione speciale; viene trattato come qualsiasi altro messaggio.
3. Quando il report viene generato, la coda MDRQ viene aperta e il messaggio di report viene inserito utilizzando l'autorizzazione di MDUID nell'MQMD del messaggio che causa il report, ad eccezione dei seguenti casi:

   - I report di eccezioni generati da un MCA ricevente vengono inseriti con qualsiasi autorizzazione utilizzata da MCA quando tenta di inserire il messaggio che causa il report. L'attributo del canale CDPA determina l'identificativo utente utilizzato.
   - I report COA generati dal gestore code vengono inseriti con qualsiasi autorizzazione utilizzata quando il messaggio che causa il report è stato inserito sul gestore code che genera il report. Ad esempio, se il messaggio è stato inserito da un MCA di ricezione utilizzando l'identificativo utente di MCA, il gestore code inserisce il report COA utilizzando l'identificativo utente di MCA.

   Le applicazioni che generano i report normalmente devono utilizzare la stessa autorizzazione che avrebbero utilizzato per generare una risposta; questa dovrebbe normalmente essere l'autorizzazione dell'identificativo utente nel messaggio originale.

   Se il report deve viaggiare verso una destinazione remota, i mittenti e i destinatari possono decidere se accettarlo o meno, allo stesso modo in cui lo fanno per altri messaggi.
4. Se è richiesto un messaggio di report con dati:

   - Il messaggio di report viene sempre generato con la quantità di dati richiesti dal mittente del messaggio originale. Se il messaggio di report è troppo grande per la coda di risposta, l'elaborazione descritta in precedenza si verifica; il messaggio di report non viene mai troncato per adattarsi alla coda di risposta.
   - Se il MDFMT del messaggio originale è FMXQH, i dati inclusi nel report non includono MQXQH. I dati del report iniziano con il primo byte dei dati oltre MQXQH nel messaggio originale. Ciò si verifica se la coda è una coda di trasmissione.
5. Se un messaggio di report COA, COD o di scadenza viene ricevuto nella coda di risposta, è garantito che il messaggio originale è arrivato, è stato consegnato o è scaduto, come appropriato. Tuttavia,

se uno o più di questi messaggi di prospetto viene richiesto e non viene ricevuto, non è possibile presumere il contrario, poiché si è verificato uno dei seguenti casi:

a. Il messaggio di report viene trattenuto perché un collegamento è inattivo.
b. Il messaggio di report viene congelato perché esiste una condizione di blocco in una coda di trasmissione intermedia o nella coda di risposta (ad esempio, la coda è piena o inibita per le inserzioni).
c. Il messaggio di report si trova su una coda di messaggi non recapitabili.
d. Quando il gestore code ha tentato di generare il messaggio di report, non è stato in grado di inserirlo nella coda appropriata e non è stato in grado di inserirlo nella coda di messaggi non recapitabili, quindi non è stato possibile generare il messaggio di report.
e. Si è verificato un errore del gestore code tra l'azione riportata (arrivo, consegna o scadenza) e la generazione del corrispondente messaggio di report. Ciò non si verifica per i messaggi di report COD se l'applicazione richiama il messaggio originale all'interno di un'unità di lavoro, poiché il messaggio di report COD viene generato all'interno della stessa unità di lavoro.

I messaggi di report di eccezione possono essere trattenuti allo stesso modo per i precedenti motivi 1, 2 e 3. Tuttavia, quando un MCA non è in grado di generare un messaggio di report di eccezione (il messaggio di report non può essere inserito nella coda di risposta o nella coda di messaggi non instradabili), il messaggio originale rimane nella coda di trasmissione del mittente e il canale viene chiuso. Ciò si verifica indipendentemente dal fatto che il messaggio di report debba essere generato all'estremità di invio o di ricezione del canale.

6. Se il messaggio originale è temporaneamente bloccato (causando la generazione di un messaggio di report di eccezione e l'inserimento del messaggio originale in una coda di messaggi non recapitabili), ma il blocco viene cancellato e un'applicazione legge il messaggio originale dalla coda di messaggi non recapitabili e lo inserisce di nuovo nella sua destinazione, potrebbe verificarsi quanto segue:
   - Anche se è stato generato un messaggio di report di eccezioni, il messaggio originale alla fine arriva correttamente alla sua destinazione.
   - Viene generato più di un messaggio di report di eccezione rispetto a un singolo messaggio originale, poiché il messaggio originale potrebbe rilevare un altro blocco in un secondo momento.

Messaggi di report durante l'inserimento in un argomento:

1. I report possono essere generati durante l'inserimento di un messaggio in un argomento. Questo messaggio verrà inviato a tutti i sottoscrittori dell'argomento, che potrebbe essere zero, uno o molti. È necessario tenerne conto quando si sceglie di utilizzare le opzioni di report, poiché di conseguenza potrebbero essere generati molti messaggi di report.
2. Quando si inserisce un messaggio in un argomento, è possibile che vi siano molte code di destinazione a cui deve essere fornita una copia del messaggio. Se alcune di queste code di destinazione presentano un problema, ad esempio la coda piena, il corretto completamento di MQPUT dipende dall'impostazione di NPMSGDLV o PMSGDLV (a seconda della persistenza del messaggio). Se l'impostazione è tale che la consegna del messaggio alla coda di destinazione deve essere eseguita correttamente (ad esempio, è un messaggio persistente per un sottoscrittore durevole e PMSGDLV è impostato su ALL o ALLDUR), l'esito positivo viene definito come uno dei seguenti criteri:
   - Inserimento riuscito nella coda del sottoscrittore
   - Utilizzo di RODLQ e di un inserimento riuscito nella DLQ (Dead - letter queue) se la coda del sottoscrittore (subscriber) non può accettare il messaggio
   - Utilizzo di RODISC se la coda del sottoscrittore non può accettare il messaggio.

Messaggi di report per segmenti di messaggi:

1. I messaggi di report possono essere richiesti per i messaggi per cui è consentita la segmentazione (vedere la descrizione dell'indicatore MFSEGA). Se il gestore code ritiene necessario segmentare il messaggio, è possibile generare un messaggio di report per ciascuno dei segmenti che successivamente rilevano la condizione pertinente. Le applicazioni devono pertanto essere

preparate a ricevere più messaggi di report per ogni tipo di messaggio di report richiesto. Il campo MDGID nel messaggio di report può essere utilizzato per correlare più report con l'identificativo del gruppo del messaggio originale e il campo MDFB utilizzato per identificare il tipo di ciascun messaggio di report.

2. Se GMLOGO viene utilizzato per richiamare i messaggi di report per i segmenti, tenere presente che i report di tipi differenti possono essere restituiti dalle successive chiamate MQGET. Ad esempio, se sono richiesti entrambi i report COA e COD per un messaggio segmentato dal gestore code, le chiamate MQGET per i messaggi di report potrebbero restituire i messaggi di report COA e COD intercalati in modo imprevedibile. Ciò può essere evitato utilizzando l'opzione GMCMPM (facoltativamente con GMATM). GMCMPM fa sì che il gestore code riassembli i messaggi di report che hanno lo stesso tipo di report. Ad esempio, la prima chiamata MQGET potrebbe riassemblare tutti i messaggi COA relativi al messaggio originale e la seconda chiamata MQGET potrebbe riassemblare tutti i messaggi COD. Il primo riassemblato dipende dal tipo di messaggio di report che si verifica per primo sulla coda.
3. Le applicazioni che inseriscono i segmenti possono specificare diverse opzioni di report per ciascun segmento. Si segnalano tuttavia i seguenti punti:
   - Se i segmenti vengono richiamati utilizzando l'opzione GMCMPM, solo le opzioni di report nel primo segmento vengono rispettate dal gestore code.
   - Se i segmenti vengono richiamati uno per uno e la maggior parte di essi dispone di una delle opzioni ROCOD*, ma almeno un segmento non lo è, non sarà possibile utilizzare l'opzione GMCMPM per richiamare i messaggi di report con una singola chiamata MQGET o utilizzare l'opzione GMASGA per rilevare quando sono arrivati tutti i messaggi di report.
4. In una rete MQ , è possibile che i gestori code abbiano funzionalità differenti. Se un messaggio di report per un segmento viene generato da un gestore code o da un MCA che non supporta la segmentazione, per impostazione predefinita il gestore code o l'MCA non includeranno le informazioni del segmento necessarie nel messaggio di report e ciò potrebbe rendere difficile identificare il messaggio originale che ha causato la generazione del report. Questa difficoltà può essere evitata richiedendo i dati con il messaggio di report, ovvero specificando le opzioni RO* D o RO* F appropriate. Tuttavia, tenere presente che se viene specificato RO* D, è possibile che vengano restituiti meno di 100 byte di dati del messaggio dell'applicazione all'applicazione che richiama il messaggio di report, se il messaggio di report è generato da un gestore code o da un MCA che non supporta la segmentazione.

Contenuto del descrittore del messaggio per un messaggio di report: quando il gestore code o MCA (message channel agent) genera un messaggio di report, imposta i campi nel descrittore del messaggio sui seguenti valori e inserisce il messaggio nel modo normale.

Tabella 708. I valori utilizzati per i campi MQMD quando un messaggio di report viene generato dal sistema

Campo in MQMD	Valore utilizzato
MDSID	MSIDV
MDVER	MDVER2
MDREP	RONONE
MDMT	MTRPRT
MDEXP	EIULIM
MDFB	Come appropriato per la natura del report (FBCOA, FBCOD, FBEXP o un valore RC*)
MDENC	Copiato dal descrittore del messaggio originale
MDCSI	Copiato dal descrittore del messaggio originale
MDFMT	Copiato dal descrittore del messaggio originale
MDPRI	Copiato dal descrittore del messaggio originale

Tabella 708. I valori utilizzati per i campi MQMD quando un messaggio di report viene generato dal sistema (Continua)

Campo in MQMD	Valore utilizzato
MDPER	Copiato dal descrittore del messaggio originale
MDMID	Come specificato dalle opzioni del prospetto nel descrittore del messaggio originale
MDCID	Come specificato dalle opzioni del prospetto nel descrittore del messaggio originale
MDBOC	0
MDRQ	Spazi
MDRM	Nome del gestore code
MDUID	Come impostato dall'opzione PMPASI
MDACC	Come impostato dall'opzione PMPASI
MDAID	Come impostato dall'opzione PMPASI
MDPAT	ATQM o come appropriato per l'agent del canale dei messaggi
MDPAN	Primi 28 byte del nome del gestore code o del nome dell'agent del canale dei messaggi. Per i messaggi di report generati dal bridge IMS , questo campo contiene i nomi del gruppo XCF e del membro XCF del sistema IMS a cui è correlato il messaggio.
MDPD	Data di invio del messaggio di report
MDPT	Ora di invio del messaggio di report
MDAOD	Spazi
MDGID	Copiato dal descrittore del messaggio originale
MDSEQ	Copiato dal descrittore del messaggio originale
MDOFF	Copiato dal descrittore del messaggio originale
MDMFL	Copiato dal descrittore del messaggio originale
MDOLN	Copiato dal descrittore del messaggio originale se non OLUNDF e impostato sulla lunghezza dei dati del messaggio originale in caso contrario

Un'applicazione che genera un report è consigliata per impostare valori simili, ad eccezione dei seguenti:

- Il campo MDRM può essere impostato su spazi vuoti (il gestore code lo modificherà con il nome del gestore code locale quando il messaggio viene inserito).
- I campi di contesto devono essere impostati utilizzando l'opzione che sarebbe stata utilizzata per una risposta, normalmente PMPASI.

Analisi del campo del report: il campo MDREP contiene campi secondari; per questo motivo, le applicazioni che devono controllare se il mittente del messaggio ha richiesto un particolare report devono utilizzare una delle tecniche descritte in "Analisi del campo del prospetto su IBM i" a pagina 1475.

Questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . Il valore iniziale di questo campo è RONONE.

MDRM (stringa di caratteri a 48 byte)

Nome del gestore code di risposte.

Questo è il nome del gestore code a cui deve essere inviato il messaggio di risposta o il messaggio di report. MDRQ è il nome locale di una coda definita su questo gestore code.

Se il campo MDRM è vuoto, il gestore code locale ricerca il nome **MDRQ** nelle relative definizioni di coda. Se esiste una definizione locale di una coda remota con questo nome, il valore **MDRM** nel messaggio trasmesso viene sostituito dal valore dell'attributo **RemoteQMgrName** dalla definizione della coda remota e questo valore verrà restituito nel descrittore del messaggio quando l'applicazione ricevente emette una chiamata MQGET per il messaggio. Se una definizione locale di una coda remota non esiste, il MDRM che viene trasmesso con il messaggio è il nome del gestore code locale.

Se il nome viene specificato, può contenere spazi vuoti finali; il primo carattere null e i caratteri che lo seguono vengono trattati come spazi vuoti. Altrimenti, tuttavia, non viene effettuato alcun controllo che il nome soddisfi le regole di denominazione per i gestori code o che questo nome sia noto al gestore code di invio; ciò è valido anche per il nome trasmesso, se il **MDRM** viene sostituito nel messaggio trasmesso.

Se una coda di risposta non è richiesta, si consiglia (anche se non è selezionata) che il campo MDRM sia impostato su spazi vuoti; il campo non deve essere lasciato non inizializzato.

Per la chiamata MQGET, il gestore code restituisce sempre il nome completato con spazi vuoti alla lunghezza del campo.

Questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . La lunghezza di questo campo è fornita da LNQMN. Il valore iniziale di questo campo è di 48 caratteri vuoti.

MDRQ (stringa di caratteri a 48 byte)

Nome della coda di risposte.

Questo è il nome della coda messaggi a cui l'applicazione che ha emesso la richiesta di richiamo per il messaggio deve inviare messaggi MTRPLY e MTRPRT. Il nome è il nome locale di una coda definita nel gestore code identificato da MDRM. Questa coda non deve essere una coda modello, sebbene il gestore code di invio non lo verifichi quando viene inserito il messaggio.

Per le chiamate MQPUT e MQPUT1 , questo campo non deve essere vuoto se il campo MDMT ha il valore MTRQST o se eventuali messaggi di report sono richiesti dal campo MDREP . Tuttavia, il valore specificato (o sostituito) viene passato all'applicazione che emette la richiesta get per il messaggio, indipendentemente dal tipo di messaggio.

Se il campo MDRM è vuoto, il gestore code locale ricerca il nome MDRQ nelle proprie definizioni di coda. Se esiste una definizione locale di una coda remota con questo nome, il valore MDRQ nel messaggio trasmesso viene sostituito dal valore dell'attributo **RemoteQName** dalla definizione della coda remota e questo valore verrà restituito nel descrittore del messaggio quando l'applicazione ricevente emette una chiamata MQGET per il messaggio. Se non esiste una definizione locale di una coda remota, MDRQ non viene modificato.

Se il nome viene specificato, può contenere spazi vuoti finali; il primo carattere null e i caratteri che lo seguono vengono trattati come spazi vuoti. Altrimenti, tuttavia, non viene effettuato alcun controllo che il nome soddisfi le regole di denominazione per le code; ciò è vero anche per il nome trasmesso, se il MDRQ viene sostituito nel messaggio trasmesso. L'unico controllo effettuato è che è stato specificato un nome, se le circostanze lo richiedono.

Se una coda di risposta non è richiesta, si consiglia (anche se non è selezionata) che il campo MDRQ sia impostato su spazi vuoti; il campo non deve essere lasciato non inizializzato.

Per la chiamata MQGET, il gestore code restituisce sempre il nome completato con spazi vuoti alla lunghezza del campo.

Se un messaggio che richiede un messaggio di report non può essere consegnato e anche il messaggio di report non può essere consegnato alla coda specificata, sia il messaggio originale che il messaggio di report vengono inviati alla coda di messaggi non recapitabili (messaggio non consegnato). Consultare l'attributo **DeadLetterQName** descritto in “Attributi per il gestore code su IBM i” a pagina 1439.

Questo è un campo di output per la chiamata MQGET e un campo di input per le chiamate MQPUT e MQPUT1 . La lunghezza di questo campo è fornita da LNQN. Il valore iniziale di questo campo è di 48 caratteri vuoti.

MDSEQ (numero intero con segno a 10 cifre)

Numero di sequenza del messaggio logico all'interno del gruppo.

I numeri della sequenza cominciano con il valore 1 e aumentano di 1 per ogni nuovo messaggio logico nel gruppo, fino a un valore massimo pari a 999 999 999. Un messaggio fisico che non si trova in un gruppo ha un numero di sequenza di 1.

Questo campo non deve essere impostato dall'applicazione nella chiamata MQPUT o MQGET se:

- Nella chiamata MQPUT, viene specificato PMLOGO.
- Nella chiamata MQGET, MOSEQN non è specificato.

Questi sono i modi consigliati per utilizzare queste chiamate per i messaggi che non sono messaggi di report. Tuttavia, se l'applicazione richiede un maggiore controllo o la chiamata è MQPUT1, l'applicazione deve verificare che MDSEQ sia impostata su un valore appropriato.

All'input delle chiamate MQPUT e MQPUT1 , il gestore code utilizza il valore dettagliato nella Tabella 1. All'output delle chiamate MQPUT e MQPUT1 , il gestore code imposta questo campo sul valore che è stato inviato con il messaggio.

In fase di input della chiamata MQGET, il gestore code utilizza il valore descritto nella Tabella 1. Nell'output della chiamata MQGET, il gestore code imposta questo campo sul valore per il messaggio richiamato.

Il valore iniziale di questo campo è uno. Questo campo viene ignorato se MDVER è inferiore a MDVER2.

MDSID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

MSIDV
Identificativo per la struttura del descrittore del messaggio.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MDSIDV.

MDUID (stringa di caratteri a 12 byte)

Identificativo utente.

Fa parte del *contesto di identità* del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

MDUID specifica l'ID utente dell'applicazione che ha originato il messaggio. Il gestore code considera queste informazioni come dati carattere, ma non ne definisce il formato.

Una volta ricevuto un messaggio, è possibile utilizzare MDUID nel campo ODAU del parametro **OBJDSC** di una chiamata MQOPEN o MQPUT1 successiva, in modo che il controllo dell'autorizzazione venga eseguito per l'utente MDUID invece che per l'applicazione che esegue l'apertura.

Quando il gestore code genera queste informazioni per una chiamata MQPUT o MQPUT1 , il gestore code utilizza un identificatore utente determinato dall'ambiente.

Quando l'identificativo utente viene determinato dall'ambiente:

- z/OS Su z/OS, il gestore code utilizza:
  - Per il batch, l'identificativo utente dalla scheda JES JOB o dall'attività avviata
  - Per TSO, l'identificativo utente di collegamento
  - Per CICS, l'identificativo utente associato all'attività
  - Per IMS, l'identificativo utente dipende dal tipo di applicazione:

- Per:
  - Regioni BMP non messaggio
  - Regioni IFP non messaggi
  - BMP messaggio e regioni IFP messaggio che non hanno emesso una chiamata GU con esito positivo

  il gestore code utilizza l'identificativo utente dalla scheda JES JOB della regione o l'identificativo utente TSO. Se questi sono vuoti o nulli, utilizza il nome del blocco di specifica del programma (PSB).
- Per:
  - BMP messaggio e regioni IFP messaggio che hanno emesso una chiamata GU con esito positivo
  - Regioni MPP

  il gestore code utilizza uno dei seguenti:
  - L'identificativo utente collegato associato al messaggio
  - Il nome del terminale logico (LTERM)
  - L'identificativo utente della scheda JES JOB della regione
  - L'identificativo utente TSO
  - Il nome PSB

- IBM i Su IBM i, il gestore code utilizza il nome del profilo utente associato con il lavoro dell'applicazione.
- Linux AIX Su AIX and Linux, il gestore code utilizza:
  - Il nome di accesso dell'applicazione
  - L'identificativo utente effettivo del processo se non è disponibile alcun accesso
  - L'identificativo utente associato alla transazione, se l'applicazione è una transazione CICS
- In VSE/ESA, questo è un campo riservato.
- Windows Su Windows, il gestore code utilizza i primi 12 caratteri del nome utente collegato.

Per le chiamate MQPUT e MQPUT1 , questo è un campo di input / output se PMSETI o PMSETA viene specificato nel parametro **PMO** . Tutte le informazioni che seguono un carattere null all'interno del campo vengono scartate. Il carattere null e i seguenti caratteri vengono convertiti in spazi dal gestore code. Se PMSETI o PMSETA non vengono specificati, questo campo viene ignorato all'immissione ed è un campo di sola emissione.

Dopo il corretto completamento di una chiamata MQPUT o MQPUT1 , questo campo contiene il `MDUID` che è stato trasmesso con il messaggio se è stato inserito in una coda. Questo sarà il valore di `MDUID` che viene conservato con il messaggio se viene conservato (vedere la descrizione di PMRET per ulteriori dettagli sulle pubblicazioni conservate) ma non viene utilizzato come `MDUID` quando il messaggio viene inviato come pubblicazione ai sottoscrittori poiché forniscono un valore da sovrascrivere `MDUID` in tutte le pubblicazioni a loro inviate. Se il messaggio non ha contesto, il campo è completamente vuoto.

Questo è un campo di output per la chiamata MQGET. La lunghezza di questo campo è fornita da LNUID. Il valore iniziale di questo campo è di 12 caratteri vuoti.

MDVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere uno dei seguenti.

MDVER1
Struttura descrittore del messaggio Version-1 .

MDVER2
Struttura del descrittore del messaggio Version-2 .

Nota: Quando viene utilizzato un MQMD version-2 , il gestore code esegue ulteriori controlli su qualsiasi struttura di intestazione MQ che può essere presente all'inizio dei dati del messaggio dell'applicazione; per ulteriori dettagli, consultare le note di utilizzo per la chiamata MQPUT.

I campi esistenti solo nella versione più recente della struttura vengono identificati come tali nelle descrizioni dei campi. La seguente costante specifica il numero di versione della versione corrente:

MDVERC
Versione corrente della struttura del descrittore del messaggio.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MDVER1.

Valori iniziali

Tabella 709. Campi in MQMD

Nome campo	Nome della costante	Valore della costante
MDSID	MSIDV	'MD¬¬'
MDVER	MDVER1	1
MDREP	RONONE	0
MDMT	MTDGRM	8
MDEXP	EIULIM	-1
MDFB	FBNONE	0
MDENC	ENNAT	Dipende dall'ambiente
MDCSI	CSQM	0
MDFMT	FMNONE	Spazi
MDPRI	PRQDEF	-1
MDPER	PEQDEF	2
MDMID	MINONE	Valori null
MDCID	CINONE	Valori null
MDBOC	Nessuna	0
MDRQ	Nessuna	Spazi
MDRM	Nessuna	Spazi
MDUID	Nessuna	Spazi
MDACC	ACNONE	Valori null
MDAID	Nessuna	Spazi
MDPAT	ATNCON	0
MDPAN	Nessuna	Spazi
MDPD	Nessuna	Spazi
MDPT	Nessuna	Spazi
MDAOD	Nessuna	Spazi

Tabella 709. Campi in MQMD (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
MDGID	GINONE	Valori null
MDSEQ	Nessuna	1
MDOFF	Nessuna	0
MDMFL	MFNONE	0
MDOLN	OLUNDF	-1
Note: 1. Il simbolo ¬ rappresenta un singolo carattere vuoto.		

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQMD Structure
D*
D* Structure identifier
D  MDSID                  1      4    INZ('MD  ')
D* Structure version number
D  MDVER                  5      8I 0 INZ(1)
D* Options for report messages
D  MDREP                  9     12I 0 INZ(0)
D* Message type
D  MDMT                  13     16I 0 INZ(8)
D* Message lifetime
D  MDEXP                 17     20I 0 INZ(-1)
D* Feedback or reason code
D  MDFB                  21     24I 0 INZ(0)
D* Numeric encoding of message data
D  MDENC                 25     28I 0 INZ(273)
D* Character set identifier of messagedata
D  MDCSI                 29     32I 0 INZ(0)
D* Format name of message data
D  MDFMT                 33     40    INZ('        ')
D* Message priority
D  MDPRI                 41     44I 0 INZ(-1)
D* Message persistence
D  MDPER                 45     48I 0 INZ(2)
D* Message identifier
D  MDMID                 49     72    INZ(X'00000000000000-
D                                     000000000000000000000-
D                                     000000000000')
D* Correlation identifier
D  MDCID                 73     96    INZ(X'00000000000000-
D                                     000000000000000000000-
D                                     000000000000')
D* Backout counter
D  MDBOC                 97    100I 0 INZ(0)
D* Name of reply queue
D  MDRQ                 101    148    INZ
D* Name of reply queue manager
D  MDRM                 149    196    INZ
D* User identifier
D  MDUID                197    208    INZ
D* Accounting token
D  MDACC                209    240    INZ(X'00000000000000-
D                                     000000000000000000000-
D                                     000000000000000000000-
D                                     000000')
D* Application data relating toidentity
D  MDAID                241    272    INZ
D* Type of application that put themessage
D  MDPAT                273    276I 0 INZ(0)
D* Name of application that put themessage
D  MDPAN                277    304    INZ
D* Date when message was put
D  MDPD                 305    312    INZ
```

```
     D* Time when message was put
     D  MDPT                 313    320    INZ
     D* Application data relating toorigin
     D  MDAOD                321    324    INZ
     D* Group identifier
     D  MDGID                325    348    INZ(X'00000000000000-
     D                                     00000000000000000000-
     D                                     000000000000')
     D* Sequence number of logical messagewithin group
     D  MDSEQ                349    352I 0 INZ(1)
     D* Offset of data in physical messagefrom start of logical message
     D  MDOFF                353    356I 0 INZ(0)
     D* Message flags
     D  MDMFL                357    360I 0 INZ(0)
     D* Length of original message
     D  MDOLN                361    364I 0 INZ(-1)
```

IBM i MQMDE (estensione descrittore messaggi) su IBM i

Panoramica

Scopo: La struttura MQMDE descrive i dati che a volte si verificano prima dei dati del messaggio dell'applicazione. La struttura contiene i campi MQMD presenti in MQMD version-2 , ma non in MQMD version-1 .

Nome formato: FMMDE.

Serie di caratteri e codifica: i dati in MQMDE devono essere nel set di caratteri fornito dall'attributo del gestore code **CodedCharSetId** e nella codifica del gestore code locale fornito da ENNAT per il linguaggio di programmazione C.

La serie di caratteri e la codifica di MQMDE devono essere impostate nei campi *MDCSI* e *MDENC* in:

- MQMD (se la struttura MQMDE si trova all'inizio dei dati del messaggio) oppure
- La struttura dell'intestazione che precede la struttura MQMDE (tutti gli altri casi).

Se MQMDE non si trova nella serie di caratteri e nella codifica del gestore code, MQMDE viene accettato ma non rispettato, vale a dire, MQMDE viene considerato come dati del messaggio.

Utilizzo: le applicazioni normali devono utilizzare un MQMD version-2 , nel qual caso non incontreranno una struttura MQMDE. Tuttavia, le applicazioni specializzate e le applicazioni che continuano a utilizzare un MQMD version-1 , possono rilevare un MQMDE in alcune situazioni. La struttura MQMDE può verificarsi nelle seguenti circostanze:

- Specificato nelle chiamate MQPUT e MQPUT1
- Restituito dalla chiamata MQGET
- Nei messaggi sulle code di trasmissione

MQMDE specificato sulle chiamate MQPUT e MQPUT1

Nelle chiamate MQPUT e MQPUT1 , se l'applicazione fornisce un MQMD version-1 , l'applicazione può facoltativamente aggiungere un prefisso ai dati del messaggio con un MQMDE, impostando il campo *MDFMT* in MQMD su FMMDE per indicare che è presente un MQMDE. Se l'applicazione non fornisce un MQMDE, il gestore code assume i valori predefiniti per i campi in MQMDE. I valori predefiniti utilizzati dal gestore code sono gli stessi dei valori iniziali per la struttura - consultare Tabella 711 a pagina 1190.

Se l'applicazione fornisce un prefisso version-2 MQMD *e* ai dati del messaggio dell'applicazione con un MQMDE, le strutture vengono elaborate come mostrato nella Tabella 710 a pagina 1187.

Tabella 710. Azione del gestore code quando MQMDE è specificato su MQPUT o MQPUT1

Versione MQMD	Valori dei campi version-2	Valori dei corrispondenti campi in MQMDE	Azione eseguita dal gestore code
1	-	Valido	MQMDE è rispettato
2	Valore predefinito	Valido	MQMDE è rispettato
2	Non predefinito	Valido	MQMDE viene trattato come dati del messaggio
1 o 2	Qualsiasi	Non valido	La chiamata ha esito negativo con un codice di errore appropriato
1 o 2	Qualsiasi	MQMDE si trova nella serie di caratteri o nella codifica non corretta oppure è una versione non supportata	MQMDE viene trattato come dati del messaggio

C'è un caso speciale. Se l'applicazione utilizza un MQMD version-2 per inserire un messaggio che è un segmento (ovvero, è impostato l'indicatore MFSEG o MFLSEG) e il nome del formato in MQMD è FMDLH, il gestore code genera una struttura MQMDE e la inserisce *tra* la struttura MQDLH e i dati che la seguono. In MQMD che il gestore code conserva con il messaggio, i campi version-2 sono impostati sui loro valori predefiniti.

Molti dei campi presenti in MQMD version-2 ma non in MQMD version-1 sono campi di input / output in MQPUT e MQPUT1. Tuttavia, il gestore code non restituisce alcun valore nei campi equivalenti in MQMDE sull'output delle chiamate MQPUT e MQPUT1 ; se l'applicazione richiede tali valori di output, deve utilizzare un MQMD version-2 .

MQMDE restituito dalla chiamata MQGET

Nella chiamata MQGET, se l'applicazione fornisce un MQMD version-1 , il gestore code antepone al messaggio restituito un MQMDE, ma solo se uno o più campi in MQMDE hanno un valore non predefinito. Il gestore code imposta il campo *MDFMT* in MQMD sul valore FMMDE per indicare che è presente un MQMDE.

Se l'applicazione fornisce un MQMDE all'inizio del parametro **BUFFER** , MQMDE viene ignorato. Al ritorno dalla chiamata MQGET, viene sostituito da MQMDE per il messaggio (se necessario) o sovrascritto dai dati del messaggio dell'applicazione (se MQMDE non è necessario).

Se MQMDE viene restituito dalla chiamata MQGET, i dati in MQMDE si trovano generalmente nella serie di caratteri e nella codifica del gestore code. Tuttavia, MQMDE potrebbe trovarsi in un'altra serie di caratteri e codificare se:

- MQMDE è stato considerato come un dato nella chiamata MQPUT o MQPUT1 (consultare Tabella 710 a pagina 1187 per le circostanze che possono causare ciò).
- Il messaggio è stato ricevuto da un gestore code remoto connesso da una connessione TCP e l'MCA (message channel agent) di ricezione non è stato impostato correttamente (per ulteriori informazioni, consultare Sicurezza degli oggetti IBM MQ for IBM i).

MQMDE nei messaggi sulle code di trasmissione

I messaggi sulle code di trasmissione hanno come prefisso la struttura MQXQH, che contiene al suo interno un MQMD version-1 . Può essere presente anche un MQMDE, posizionato tra la struttura MQXQH

e i dati del messaggio dell'applicazione, ma in genere sarà presente solo se uno o più campi in MQMDE hanno un valore non predefinito.

Tra la struttura MQXQH e i dati del messaggio dell'applicazione possono verificarsi anche altre strutture di intestazione IBM MQ . Ad esempio, quando l'intestazione MQDLH è presente e il messaggio non è un segmento, l'ordine è:

- MQXQH (contenente un MQMD version-1)
- MQMDE
- MQDLH
- Dati messaggio applicazione

Campi

La struttura MQMDE contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

MECSI (numero intero con segno a 10 cifre)

Identificativo della serie di caratteri dei dati che seguono MQMDE.

Specifica l'identificativo della serie di caratteri dei dati che seguono la struttura MQMDE; non si applica ai dati carattere nella struttura MQMDE stessa.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Il gestore code non controlla che questo campo sia valido. È possibile utilizzare il seguente valore speciale:

CINHT
Eredita l'identificativo della serie di caratteri di questa struttura.

I dati carattere nei dati *che seguono* questa struttura si trovano nella stessa serie di caratteri di questa struttura.

Il gestore code modifica questo valore nella struttura inviata nel messaggio nell'effettivo identificativo della serie di caratteri della struttura. Se non si verifica alcun errore, il valore CSINHT non viene restituito dalla chiamata MQGET.

CSINHT non può essere utilizzato se il valore del campo *MDPAT* in MQMD è ATBRKR.

Il valore iniziale di questo campo è CSUNDF.

MEENC (numero intero con segno a 10 cifre)
MEENC (numero intero con segno a 10 cifre)

Specifica la codifica numerica dei dati che seguono la struttura MQMDE; non si applica ai dati numerici nella struttura MQMDE stessa.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Il gestore code non verifica la validità del campo. Consultare il campo *MDENC* descritto in “MQMD (Message Descriptor) su IBM i” a pagina 1140 per ulteriori informazioni sulle codifiche dei dati.

Il valore iniziale di questo campo è ENNAT.

MEFLG (numero intero con segno a 10 cifre)

Indicatori generali.

È possibile specificare il seguente indicatore:

MEFNON
Nessun indicatore.

Il valore iniziale di questo campo è MEFNON.

MEFMT (stringa di caratteri a 8 byte)

Nome formato dei dati che seguono MQMDE.

Specifica il nome del formato dei dati che seguono la struttura MQMDE.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Il gestore code non controlla che questo campo sia valido. Consultare il campo *MDFMT* descritto in "MQMD (Message Descriptor) su IBM i" a pagina 1140 per ulteriori informazioni sui nomi dei formati.

Il valore iniziale di questo campo è FMNONE.

MEGID (stringa bit a 24 byte)

Identificativo gruppo.

Consultare il campo *MDGID* descritto in "MQMD (Message Descriptor) su IBM i" a pagina 1140. Il valore iniziale di questo campo è GINONE.

MELEN (numero intero con segno a 10 cifre)

Lunghezza della struttura MQMDE.

Viene definito il seguente valore:

MELEN2
Lunghezza della struttura di estensione del descrittore del messaggio version-2 .

Il valore iniziale di questo campo è MELEN2.

MEMFL (numero intero con segno a 10 cifre)

Gli indicatori del messaggio.

Consultare il campo *MDMFL* descritto in "MQMD (Message Descriptor) su IBM i" a pagina 1140. Il valore iniziale di questo campo è MFNONE.

MEOFF (numero intero con segno a 10 cifre)

L'offset dei dati nel messaggio fisico dall'inizio del messaggio logico.

Consultare il campo *MDOFF* descritto in "MQMD (Message Descriptor) su IBM i" a pagina 1140. Il valore iniziale di questo campo è 0.

MEOLN (numero intero con segno a 10 cifre)

Lunghezza del messaggio originale.

Consultare il campo *MDOLN* descritto in "MQMD (Message Descriptor) su IBM i" a pagina 1140. Il valore iniziale di questo campo è OLUNDF.

MESEQ (numero intero con segno a 10 cifre)

Numero di sequenza del messaggio logico all'interno del gruppo.

Consultare il campo *MDSEQ* descritto in "MQMD (Message Descriptor) su IBM i" a pagina 1140. Il valore iniziale di questo campo è 1.

MESID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

IDESM
Identificativo per la struttura di estensione del descrittore del messaggio.

Il valore iniziale di questo campo è MESIDV.

MEVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere:

MEVER2
Struttura di estensione del descrittore messaggi Version-2 .

La seguente costante specifica il numero di versione della versione corrente:

MEVERC
La versione corrente della struttura di estensione del descrittore del messaggio.

Il valore iniziale di questo campo è MEVER2.

Valori iniziali

Tabella 711. Campi in MQMDE

Nome campo	Nome della costante	Valore della costante
MESID	IDESM	'MDE¬'
MEVER	MEVER2	2
MELEN	MELEN2	72
MEENC	ENNAT	Dipende dall'ambiente
MECSI	SUNDF	0
MEFMT	FMNONE	Spazi
MEFLG	MEFNON	0
MEGID	GINONE	Valori null
MESEQ	Nessuna	1
MEOFF	Nessuna	0
MEMFL	MFNONE	0
MEOLN	OLUNDF	-1

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQMDE Structure
D*
D* Structure identifier
D  MESID                  1      4    INZ('MDE ')
D* Structure version number
D  MEVER                  5      8I 0 INZ(2)
D* Length of MQMDE structure
D  MELEN                  9     12I 0 INZ(72)
D* Numeric encoding of data that followsMQMDE
D  MEENC                 13     16I 0 INZ(273)
D* Character-set identifier of data thatfollows MQMDE
D  MECSI                 17     20I 0 INZ(0)
D* Format name of data that followsMQMDE
D  MEFMT                 21     28    INZ('        ')
D* General flags
D  MEFLG                 29     32I 0 INZ(0)
D* Group identifier
D  MEGID                 33     56    INZ(X'00000000000000-
D                                     00000000000000000000-
D                                     000000000000')
D* Sequence number of logical messagewithin group
D  MESEQ                 57     60I 0 INZ(1)
D* Offset of data in physical messagefrom start of logical message
```

```
D  MEOFF                 61     64I 0 INZ(0)
D* Message flags
D  MEMFL                 65     68I 0 INZ(0)
D* Length of original message
D  MEOLN                 69     72I 0 INZ(-1)
```

IBM i MQMHBO (Gestione messaggi per opzioni buffer) su IBM i

Struttura che definisce l'handle del messaggio per le opzioni di buffer

Panoramica

Scopo: la struttura MQMHBO consente alle applicazioni di specificare le opzioni che controllano il modo in cui i buffer vengono prodotti dagli handle dei messaggi. La struttura è un parametro di immissione nella chiamata MQMHBUF.

Serie di caratteri e codifica: i dati in MQMHBO devono trovarsi nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (ENNAT).

- "Campi" a pagina 1191
- "Valori iniziali" a pagina 1192
- "Dichiarazione RPG" a pagina 1192

Campi

La struttura MQMHBO contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

MBOPT (numero intero con segno a 10 cifre)

Gestore messaggi per la struttura delle opzioni del buffer - Campo MBOPT.

Queste opzioni controllano l'azione di MQMHBUF.

È possibile specificare la seguente opzione:

MBPRRF
Durante la conversione delle proprietà da un handle del messaggio in un buffer, convertirle nel formato MQRFH2 .

Facoltativamente, è anche possibile specificare la seguente opzione. Per specificare più di un'opzione, aggiungere i valori insieme (non aggiungere la stessa costante più di una volta) o combinare i valori utilizzando l'operazione OR bit per bit (se il linguaggio di programmazione supporta le operazioni bit).

MBDLPR
Le proprietà che vengono aggiunte al buffer vengono eliminate dall'handle del messaggio. Se la chiamata ha esito negativo, non viene eliminata alcuna proprietà.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MBPRRF.

MBSID (numero intero con segno a 10 cifre)

Handle del messaggio per la struttura delle opzioni del buffer - campo MBSID.

Questo è l'identificativo della struttura. Il valore deve essere:

MBSIDV
Identificativo per l'handle del messaggio nella struttura delle opzioni del buffer.

Questo è sempre un campo di input. Il valore iniziale di questo campo isMBSIDV.

MBVER (numero intero con segno a 10 cifre)

Questo è il numero di versione della struttura. Il valore deve essere:

MBVER1
Numero di versione per l'handle del messaggio nella struttura delle opzioni del buffer.

La seguente costante specifica il numero di versione della versione corrente:

MBVERC
La versione corrente dell'handle del messaggio nella struttura delle opzioni del buffer.

Questo è sempre un campo di input. Il valore iniziale di questo campo è MBVER1.

Valori iniziali

Tabella 712. Campi in MQMHBO

Nome campo	Nome della costante	Valore della costante
MVSID	MBSIDV	'MHBO'
MBVER	MBVER1	1
MBOPT	MBPRRF	

Note:

1. Il valore Stringa null o spazi vuoti indica un carattere vuoto.

Dichiarazione RPG

```
D* MQMHBO Structure
D*
D*
D* Structure identifier
D  MBSID                  1      4    INZ('MHBO')
D*
D* Structure version number
D  MBVER                  5      8I 0 INZ(1)
D*
D* Options that control the action of MQMHBUF
D  MBOPT                  9     12I 0 INZ(1)
```

IBM i MQOD (Object descriptor) su IBM i

La struttura MQOD viene utilizzata per specificare un oggetto in base al nome.

Panoramica

Scopo: I seguenti tipi di oggetto sono validi:

- Coda o elenco di distribuzione
- Elenco nomi
- Definizione di processo
- Gestore code
- Argomento

La struttura è un parametro di input / output sulle chiamate MQOPEN e MQPUT1 .

Versione: la versione corrente di MQOD è ODVER4. I campi che esistono solo nelle versioni più recenti della struttura sono identificati come tali nelle descrizioni che seguono.

Il file COPY fornito contiene la versione più recente di MQOD supportata dall'ambiente, ma con il valore iniziale del campo *ODVER* impostato su ODVER1. Per utilizzare i campi che non sono presenti nella struttura version-1 , l'applicazione deve impostare il campo *ODVER* sul numero di versione della versione richiesta.

Per aprire un elenco di distribuzione, *ODVER* deve essere ODVER2 o superiore.

Serie di caratteri e codifica: i dati in MQOD devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e la codifica del gestore code locale fornita da ENNAT. Tuttavia, se

l'applicazione è in esecuzione come un client IBM MQ , la struttura deve essere nella serie di caratteri e nella codifica del client.

- "Campi" a pagina 1193
- "Valori iniziali" a pagina 1200
- "Dichiarazione RPG" a pagina 1201

Campi

La struttura MQOD contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

ODASI (stringa bit a 40 byte)

Identificativo di sicurezza alternativo.

Si tratta di un identificativo di sicurezza che viene passato con *ODAU* al servizio di autorizzazione per consentire l'esecuzione dei controlli di autorizzazione appropriati. *ODASI* viene utilizzato solo se:

- OOALTU è specificato nella chiamata MQOPEN oppure
- PMALTU è specificato nella chiamata MQPUT1 ,

e il campo *ODAU* non è completamente vuoto fino al primo carattere null o alla fine del campo.

Il campo *ODASI* ha la seguente struttura:

- Il primo byte è un numero intero binario contenente la lunghezza dei dati significativi che seguono; il valore esclude il byte di lunghezza stesso. Se non è presente alcun identificativo di sicurezza, la lunghezza è zero.
- Il secondo byte indica il tipo di identificativo di sicurezza presente; sono possibili i seguenti valori:

  SITONT
  Identificativo di sicurezza Windows .

  NON SITO
  Nessun identificativo di sicurezza.
- Il terzo byte e i byte successivi fino alla lunghezza definita dal primo byte contengono l'identificativo di sicurezza stesso.
- I byte rimanenti nel campo sono impostati a zero binario.

È possibile utilizzare il seguente valore speciale:

SINONE
Nessun identificativo di sicurezza specificato.

Il valore è zero binario per la lunghezza del campo.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da LNSCID. Il valore iniziale di questo campo è SINONE. Questo campo viene ignorato se *ODVER* è inferiore a ODVER3.

ODAU (stringa di caratteri a 12 byte)

Identificativo utente alternativo.

Se OOALTU è specificato per la chiamata MQOPEN o PMALTU per la chiamata MQPUT1 , questo campo contiene un identificativo utente alternativo che deve essere utilizzato per controllare l'autorizzazione per l'apertura, al posto dell'identificativo utente con cui l'applicazione è attualmente in esecuzione. Alcuni controlli, tuttavia, vengono ancora eseguiti con l'identificativo utente corrente (ad esempio, controlli di contesto).

Se OOALTU e PMALTU non vengono specificati e questo campo è completamente vuoto fino al primo carattere null o alla fine del campo, l'apertura può avere esito positivo solo se non è necessaria alcuna autorizzazione utente per aprire questo oggetto con le opzioni specificate.

Se non viene specificato né OOALTU né PMALTU, questo campo viene ignorato.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da LNUID. Il valore iniziale di questo campo è di 12 caratteri vuoti.

ODDN (stringa di caratteri a 48 byte)

Nome coda dinamica.

Questo è il nome di una coda dinamica che deve essere creata dalla chiamata MQOPEN. Ciò è rilevante solo quando *ODON* specifica il nome di una coda modello; in tutti gli altri casi *ODDN* viene ignorato.

I caratteri validi nel nome sono gli stessi di quelli per *ODON*, ma è valido anche un asterisco. Un nome che è vuoto (o uno in cui vengono visualizzati solo spazi vuoti prima del primo carattere null) non è valido se *ODON* è il nome di una coda modello.

Se l'ultimo carattere non vuoto nel nome è un asterisco (`*`), il gestore code sostituisce l'asterisco con una stringa di caratteri che garantisce che il nome generato per la coda sia univoco nel gestore code locale. Per consentire un numero di caratteri sufficiente, l'asterisco è valido solo nelle posizioni da 1 a 33. Non devono essere presenti caratteri diversi dagli spazi o un carattere null dopo l'asterisco.

È valido che l'asterisco si trovi nella posizione del primo carattere, nel qual caso il nome è costituito esclusivamente dai caratteri generati dal gestore code.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da LNQN. Il valore iniziale di questo campo è `'AMQ.*'`, riempito con spazi.

ODIDC (numero intero con segno a 10 cifre)

Numero di code che non è stato possibile aprire.

Questo è il numero di code nell'elenco di distribuzione che non sono state aperte correttamente. Se presente, questo campo viene impostato anche quando si apre una singola coda che non si trova in un elenco di distribuzione.

Nota: Se presente, questo campo è impostato solo se il parametro **CMPCOD** nella chiamata MQOPEN o MQPUT1 è CCOK o CCWARN; non è impostato se il parametro **CMPCOD** è CCFAIL.

Questo è un campo di output. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *ODVER* è minore di ODVER2.

ODKDC (numero intero con segno a 10 cifre)

Numero di code locali aperte correttamente.

Questo è il numero di code nell'elenco di distribuzione che si risolvono in code locali e che sono state aperte correttamente. Il conteggio non include le code che si risolvono in code remote (anche se una coda di trasmissione locale viene utilizzata inizialmente per memorizzare il messaggio). Se presente, questo campo viene impostato anche quando si apre una singola coda che non si trova in un elenco di distribuzione.

Questo è un campo di output. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *ODVER* è minore di ODVER2.

ODMN (stringa di caratteri a 48 byte)

Nome gestore code oggetti.

Questo è il nome del gestore code su cui è definito l'oggetto *ODON* . I caratteri validi nel nome sono gli stessi di quelli per *ODON* (vedere in precedenza). Un nome che è completamente vuoto fino al primo carattere null o alla fine del campo indica il gestore code a cui è connessa l'applicazione (il gestore code locale).

I seguenti punti si applicano ai tipi di oggetto indicati:

- Se *ODOT* è OTTOP, OTNLST, OTPRO o OTQM, *ODMN* deve essere vuoto o il nome del gestore code locale.
- Se *ODON* è il nome di una coda modello, il gestore code crea una coda dinamica con gli attributi della coda modello e restituisce nel campo *ODMN* il nome del gestore code su cui è stata creata la

coda; questo è il nome del gestore code locale. Una coda modello può essere specificata solo sulla chiamata MQOPEN; una coda modello non è valida sulla chiamata MQPUT1 .

- Se *ODON* è il nome di una coda cluster e *ODMN* è vuoto, la destinazione effettiva dei messaggi inviati utilizzando l'handle della coda restituito dalla chiamata MQOPEN viene scelta dal gestore code (o dall'uscita del workload del cluster, se installato) come segue:
  - Se viene specificato OOBNDO, il gestore code seleziona un'istanza della coda del cluster durante l'elaborazione della chiamata MQOPEN e tutti i messaggi immessi utilizzando questo handle della coda vengono inviati a tale istanza.
  - Se viene specificato OOBNDN, il gestore code può scegliere un'istanza differente della coda di destinazione (che risiede su un gestore code differente nel cluster) per ogni chiamata MQPUT successiva che utilizza questo handle di coda.

  Se l'applicazione deve inviare un messaggio a un'istanza *specifica* di una coda cluster (ovvero, un'istanza della coda che si trova su un determinato gestore code nel cluster), l'applicazione deve specificare il nome di tale gestore code nel campo *ODMN* . Ciò forza il gestore code locale a inviare il messaggio al gestore code di destinazione specificato.
- Se l'oggetto che si sta aprendo è un elenco di distribuzione (ovvero, *ODREC* è maggiore di zero), *ODMN* deve essere vuoto o la stringa null. Se questa condizione non viene soddisfatta, la chiamata ha esito negativo con codice di errore RC2153.

Questo è un campo di input / output per la chiamata MQOPEN quando *ODON* è il nome di una coda modello e un campo di solo input in tutti gli altri casi. La lunghezza di questo campo è fornita da LNQMN. Il valore iniziale di questo campo è di 48 caratteri vuoti.

ODON (stringa di caratteri a 48 byte)

Il nome dell'oggetto.

Questo è il nome locale dell'oggetto come definito sul gestore code identificato da *ODMN*. Il nome può contenere i seguenti caratteri:

- Caratteri alfabetici maiuscoli (A - Z)
- Caratteri alfabetici minuscoli (a - z)
- Cifre numeriche (0 - 9)
- Punto (.), barra (/), sottolineatura (_), percentuale (%)

Il nome non deve contenere spazi iniziali o intermedi, ma può contenere spazi finali. Un carattere null può essere utilizzato per indicare la fine dei dati significativi nel nome; il valore null e i caratteri che lo seguono vengono trattati come spazi vuoti. Le seguenti limitazioni si applicano agli ambienti indicati:

- Sui sistemi che utilizzano EBCDIC Katakana, non è possibile utilizzare caratteri minuscoli.
- Su IBM i, i nomi contenenti caratteri minuscoli, barra o percentuale, devono essere racchiusi tra virgolette quando vengono specificati nei comandi. Questi apici non devono essere specificati per i nomi che si verificano come campi nelle strutture o come parametri nelle chiamate.

I seguenti punti si applicano ai tipi di oggetto indicati:

- Se *ODON* è il nome di una coda modello, il gestore code crea una coda dinamica con gli attributi della coda modello e restituisce nel campo *ODON* il nome della coda creata. Una coda modello può essere specificata solo sulla chiamata MQOPEN; una coda modello non è valida sulla chiamata MQPUT1 .
- Se l'oggetto che si sta aprendo è un elenco di distribuzione (ovvero, *ODREC* è presente e maggiore di zero), *ODON* deve essere vuoto o la stringa null. Se questa condizione non viene soddisfatta, la chiamata ha esito negativo con codice di errore RC2152.
- Se *ODOT* è OTQM, si applicano regole speciali; in questo caso, il nome deve essere completamente vuoto fino al primo carattere null o alla fine del campo.
- Se ODON è il nome di una coda alias con TARGTYPE (TOPIC), viene prima eseguito un controllo di sicurezza sulla coda alias denominata, come è normale per l'utilizzo delle code alias. Se questo controllo di sicurezza ha esito positivo, questa chiamata MQOPEN continuerà e si comporterà come

un MQOPEN di un OTTOP, inclusa l'esecuzione di un controllo di sicurezza rispetto all'oggetto argomento di gestione.

Questo è un campo di input / output per la chiamata MQOPEN quando *ODON* è il nome di una coda modello e un campo di solo input in tutti gli altri casi. La lunghezza di questo campo è fornita da LNQN. Il valore iniziale di questo campo è di 48 caratteri vuoti.

Il nome completo dell'argomento può essere creato da due campi differenti: *ODON* e *ODOS*. Per i dettagli sul modo in cui questi due campi vengono utilizzati, consultare Combinazione di stringhe argomento.

ODORO (numero intero con segno a 10 cifre)

Offset del primo record oggetto dall'inizio di MQOD.

Questo è l'offset in byte del primo record di oggetto MQOR dall'inizio della struttura MQOD. L'offset può essere positivo o negativo. *ODORO* viene utilizzato solo quando si sta aprendo un elenco di distribuzione. Il campo viene ignorato se *ODREC* è zero.

Quando viene aperto un elenco di distribuzione, è necessario fornire un array di uno o più record oggetto MQOR per specificare i nomi delle code di destinazione nell'elenco di distribuzione. Questa operazione può essere eseguita in due modi:

- Utilizzando il campo offset *ODORO*

  In tal caso, l'applicazione deve dichiarare la propria struttura contenente un MQOD seguito dall'array di record MQOR (con tutti gli elementi dell'array necessari) e impostare *ODORO* sull'offset del primo elemento dell'array dall'inizio del MQOD. È necessario assicurarsi che questo offset sia corretto.
- Utilizzando il campo puntatore *ODORP*

  In questo caso, l'applicazione può dichiarare l'array delle strutture MQOR separatamente dalla struttura MQOD e impostare *ODORP* sull'indirizzo dell'array.

Qualunque sia la tecnica scelta, è necessario utilizzare *ODORO* e *ODORP* ; la chiamata ha esito negativo con codice di errore RC2155 se entrambi sono zero o sono entrambi diversi da zero.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *ODVER* è minore di ODVER2.

ODORP (puntatore)

Indirizzo del primo record oggetto.

Questo è l'indirizzo del primo record di oggetto MQOR. *ODORP* viene utilizzato solo quando si sta aprendo un elenco di distribuzione. Il campo viene ignorato se *ODREC* è zero.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null. È possibile utilizzare *ODORP* o *ODORO* per specificare i record oggetto, ma non entrambi; consultare la descrizione del campo *ODORO* in precedenza per i dettagli. Se *ODORP* non viene utilizzato, deve essere impostato sul puntatore null o sui byte null. Questo campo viene ignorato se *ODVER* è minore di ODVER2.

ODOS (MQCHARV)

ODOS specifica il nome oggetto lungo da utilizzare.

A questo campo si fa riferimento solo per determinati valori di *ODOT*. Consultare la descrizione di *ODOT* per i dettagli sui valori che indicano che questo campo è utilizzato.

Se *ODOS* viene specificato in modo non corretto, in base alla descrizione di come utilizzare la struttura MQCHARV o se supera la lunghezza massima, la chiamata ha esito negativo con codice di errore RC2441.

Questo è un campo di immissione. I valori iniziali dei campi in questa struttura sono gli stessi della struttura MQCHARV.

Il nome completo dell'argomento può essere creato da due campi differenti: *ODON* e *ODOS*. Per i dettagli sul modo in cui questi due campi vengono utilizzati, consultare Combinazione di stringhe argomento. Questo campo viene ignorato se *ODVER* è minore di ODVER4.

ODOT (numero intero con segno a 10 cifre)

Tipo di oggetto.

Tipo di oggetto denominato in *ODON*. I possibili valori sono:

OTQ
Coda. Il nome dell'oggetto si trova in *ODON*.

OTNLST
Elenco nomi. Il nome dell'oggetto si trova in *ODON*.

OTPRO
Definizione processo. Il nome dell'oggetto si trova in *ODON*.

OTQM
Gestore code. Il nome dell'oggetto si trova in *ODON*.

OTTOP
. Il nome completo dell'argomento può essere creato da due campi differenti: *ODON* e *ODOS*.

Per i dettagli su come vengono utilizzati questi due campi, consultare Combinazione di stringhe di argomenti.

Se non è possibile trovare l'oggetto identificato dal campo *ODON* , la chiamata avrà esito negativo con il codice di errore RC2425 anche se è presente una stringa specificata in *ODOS*.

Questo è sempre un campo di input. Il valore iniziale di questo campo è OTQ.

ODREC (numero intero con segno a 10 cifre)

Numero di record oggetto presenti.

Questo è il numero di record oggetto MQOR forniti dall'applicazione. Se questo numero è maggiore di zero, indica che si sta aprendo un elenco di distribuzione, con *ODREC* che rappresenta il numero di code di destinazione nell'elenco. È valido che un elenco di distribuzione contenga una sola destinazione.

Il valore di *ODREC* non deve essere inferiore a zero e, se è maggiore di zero, *ODOT* deve essere OTQ; la chiamata ha esito negativo con codice motivo RC2154 se queste condizioni non vengono soddisfatte.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *ODVER* è minore di ODVER2.

ODRMN (stringa di caratteri a 48 byte)

Nome gestore code risolto.

Questo è il nome del gestore code di destinazione dopo che la risoluzione dei nomi è stata eseguita dal gestore code locale. Il nome restituito è il nome del gestore code proprietario della coda identificata da *ODRQN*. *ODRMN* può essere il nome del gestore code locale.

Se *ODRQN* è una coda condivisa di proprietà del gruppo di condivisione code a cui appartiene il gestore code locale, *ODRMN* è il nome del gruppo di condivisione code. Se la coda è di proprietà di un altro gruppo di condivisione code, *ODRQN* può essere il nome del gruppo di condivisione code o il nome di un gestore code che è un membro del gruppo di condivisione code (la natura del valore restituito è determinata dalle definizioni di coda che esistono nel gestore code locale).

Un valore non vuoto viene restituito solo se l'oggetto è una singola coda aperta per la ricerca, l'immissione o l'emissione (o qualsiasi combinazione). Se l'oggetto aperto è uno dei seguenti, *ODRMN* viene impostato su spazi vuoti:

- Non è una coda
- Una coda, ma non aperta per la ricerca, l'immissione o l'emissione

- Una coda cluster con OOBNDN specificato (o con OOBNDQ attivo quando l'attributo della coda **DefBind** ha il valore BNDNOT)
- Un elenco di distribuzione

Questo è un campo di output. La lunghezza di questo campo è fornita da LNQN. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione. Questo campo viene ignorato se *ODVER* è inferiore a ODVER3.

ODRO (MQCHARV)

ODRO è il nome oggetto lungo dopo che il gestore code ha risolto il nome fornito in *ODON*.

Questo campo viene restituito solo per alcuni tipi di oggetti, argomenti e alias coda che fanno riferimento a un oggetto argomento.

Se il nome oggetto lungo viene fornito in *ODOS* e non viene fornito nulla in *ODON*, il valore restituito in questo campo è uguale a quello fornito in *ODOS*.

Se questo campo viene omesso (ovvero ODRO.VSBufSize è zero), il *ODRO* non viene restituito, ma la lunghezza viene restituita in ODRO.VSLength. Se la lunghezza è inferiore al valore *ODRO* completo, viene troncato e restituisce il numero massimo di caratteri più a destra che possono rientrare nella lunghezza fornita.

Se *ODRO* non viene specificato correttamente, in base alla descrizione di come utilizzare la struttura MQCHARV o se supera la lunghezza massima, la chiamata ha esito negativo con codice di errore RC2520. Questo campo viene ignorato se *ODVER* è minore di ODVER4.

ODRQN (stringa di caratteri a 48 byte)

Nome coda risolto.

Questo è il nome della coda di destinazione dopo che la risoluzione del nome è stata eseguita dal gestore code locale. Il nome restituito è il nome di una coda che esiste sul gestore code identificato da *ODRMN*.

Un valore non vuoto viene restituito solo se l'oggetto è una singola coda aperta per la ricerca, l'immissione o l'emissione (o qualsiasi combinazione). Se l'oggetto aperto è uno dei seguenti, *ODRQN* viene impostato su spazi vuoti:

- Non è una coda
- Una coda, ma non aperta per la ricerca, l'immissione o l'emissione
- Un elenco di distribuzione
- Una coda alias che fa riferimento a un oggetto argomento (fare riferimento invece a "ODRO (MQCHARV)" a pagina 1198)

Questo è un campo di output. La lunghezza di questo campo è fornita da LNQN. Il valore iniziale di questo campo è la stringa nulla in C e 48 caratteri vuoti in altri linguaggi di programmazione. Questo campo viene ignorato se *ODVER* è inferiore a ODVER3.

ODRRO (numero intero con segno a 10 cifre)

Offset del primo record di risposta dall'inizio di MQOD.

Questo è l'offset in byte del primo record di risposta MQRR dall'inizio della struttura MQOD. L'offset può essere positivo o negativo. *ODRRO* viene utilizzato solo quando si sta aprendo un elenco di distribuzione. Il campo viene ignorato se *ODREC* è zero.

Quando un elenco di distribuzione viene aperto, è possibile fornire un array di uno o più record di risposta MQRR per identificare le code che non sono state aperte (campo*RRCC* in MQRR) e il motivo di ciascun errore (campo*RRREA* in MQRR). I dati vengono restituiti nella schiera di record di risposta nello stesso ordine in cui i nomi coda si verificano nella schiera di record oggetto. Il gestore code imposta i record di risposta solo quando il risultato della chiamata è misto (ovvero, alcune code sono state aperte correttamente mentre altre non sono riuscite o tutte non sono riuscite, ma per motivi diversi); il codice motivo RC2136 dalla chiamata indica questo caso. Se lo stesso codice motivo si applica a tutte le code, tale motivo viene restituito nel parametro **REASON** della chiamata MQOPEN

o MQPUT1 e i record di risposta non vengono impostati. I record di risposta sono facoltativi, ma se vengono forniti devono essere *ODREC* .

I record di risposta possono essere forniti nello stesso modo dei record di oggetto, specificando un offset in *ODRRO*o specificando un indirizzo in *ODRRP* ; Consultare la descrizione di *ODORO* precedentemente per dettagli su come eseguire questa operazione. Tuttavia, non è possibile utilizzare più di uno tra *ODRRO* e *ODRRP* ; la chiamata ha esito negativo con codice motivo RC2156 se entrambi sono diversi da zero.

Per la chiamata MQPUT1 , questi record di risposta vengono utilizzati per restituire informazioni sugli errori che si verificano quando il messaggio viene inviato alle code nell'elenco di distribuzione, nonché gli errori che si verificano quando le code vengono aperte. Il codice di completamento e il codice motivo dell'operazione di inserimento per una coda sostituiscono quelli dell'operazione di apertura per quella coda solo se il codice di completamento dell'ultima è CCOK o CCWARN.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *ODVER* è minore di ODVER2.

ODRRP (puntatore)

Indirizzo del primo record di risposta.

Questo è l'indirizzo del primo record di risposta MQRR. *ODRRP* viene utilizzato solo quando si sta aprendo un elenco di distribuzione. Il campo viene ignorato se *ODREC* è zero.

ODRRP o *ODRRO* possono essere utilizzati per specificare i record di risposta, ma non entrambi; consultare la descrizione precedente del campo *ODRRO* per i dettagli. Se *ODRRP* non viene utilizzato, deve essere impostato sul puntatore null o sui byte null.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null. Questo campo viene ignorato se *ODVER* è minore di ODVER2.

ODSID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

ODSIDV
Identificativo per la struttura descrittore dell'oggetto.

Questo è sempre un campo di input. Il valore iniziale di questo campo è ODSIDV.

ODSS (MQCHARV)

ODSS contiene la stringa utilizzata per fornire i criteri di selezione utilizzati durante il richiamo dei messaggi da una coda.

ODSS non deve essere fornito nei casi seguenti:

- Se *ODOT* non è OTQ
- Se la coda che si sta aprendo non viene aperta utilizzando una delle opzioni di input, OOINP*

Se *ODSS* viene fornito in questi casi, la chiamata ha esito negativo con codice di errore RC2516.

Se *ODSS* viene specificato in modo non corretto, in base alla descrizione di come utilizzare la struttura MQCHARV oppure se supera la lunghezza massima, la chiamata ha esito negativo con codice di errore RC2519. Questo campo viene ignorato se *ODVER* è minore di ODVER4.

ODUDC (numero intero con segno a 10 cifre)

Numero di code remote aperte correttamente

Questo è il numero di code nell'elenco di distribuzione che si risolvono in code remote e che sono state aperte correttamente. Se presente, questo campo viene impostato anche quando si apre una singola coda che non si trova in un elenco di distribuzione.

Questo è un campo di output. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *ODVER* è minore di ODVER2.

ODVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere uno dei seguenti.

ODVER1
Struttura del descrittore oggetto Version-1 .

ODVER2
Struttura descrittore oggetto Version-2 .

ODVER3
Struttura del descrittore oggetto Version-3 .

ODVER4
Struttura descrittore oggetto Version-4 .

I campi che esistono solo nelle versioni più recenti della struttura vengono identificati come tali nelle descrizioni dei campi. La seguente costante specifica il numero di versione della versione corrente:

ODVERC
Versione corrente della struttura descrittore dell'oggetto.

Questo è sempre un campo di input. Il valore iniziale di questo campo è ODVER1.

Valori iniziali

Tabella 713. Campi in MQOD

Nome campo	Nome della costante	Valore della costante
ODSID	ODSIDV	'OD¬¬'
ODVER	ODVER1	1
ODOT	OTQ	1
ODON	Nessuna	Spazi
ODMN	Nessuna	Spazi
ODDN	Nessuna	'AMQ.*'
ODAU	Nessuna	Spazi
ODREC	Nessuna	0
ODKDC	Nessuna	0
ODUDC	Nessuna	0
ODIDC	Nessuna	0
ODORO	Nessuna	0
ODRRO	Nessuna	0
ODORP	Nessuna	Puntatore null o byte null
ODRRP	Nessuna	Puntatore null o byte null
ODASI	SINONE	Valori null
ODRQN	Nessuna	Spazi
ODRMN	Nessuna	Spazi
ODOS	Come definito per MQCHARV	Come definito per MQCHARV
ODRO	Come fornito in *ODOS*	Come fornito in *ODOS*

Tabella 713. Campi in MQOD (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
ODSS	Nessuna	Spazi
Note: 1. Il simbolo ¬ rappresenta un singolo carattere vuoto.		

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQOD Structure
D*
D*
D* Structure identifier
D  ODSID                  1      4    INZ('OD  ')
D*
D* Structure version number
D  ODVER                  5      8I 0 INZ(1)
D*
D* Object type
D  ODOT                   9     12I 0 INZ(1)
D*
D* Object name
D  ODON                  13     60    INZ
D*
D* Object queue manager name
D  ODMN                  61    108    INZ
D*
D* Dynamic queue name
D  ODDN                 109    156    INZ('AMQ.*')
D*
D* Alternate user identifier
D  ODAU                 157    168    INZ
D*
** Number of object records
D* present
D  ODREC                169    172I 0 INZ(0)
D*
** Number of local queues opened
D* successfully
D  ODKDC                173    176I 0 INZ(0)
D*
** Number of remote queues opened
D* successfully
D  ODUDC                177    180I 0 INZ(0)
D*
** Number of queues that failed to
D* open
D  ODIDC                181    184I 0 INZ(0)
D*
** Offset of first object record
D* from start of MQOD
D  ODORO                185    188I 0 INZ(0)
D*
** Offset of first response record
D* from start of MQOD
D  ODRRO                189    192I 0 INZ(0)
D*
D* Address of first object record
D  ODORP                193    208*   INZ(*NULL)
D*
** Address of first response
D* record
D  ODRRP                209    224*   INZ(*NULL)
D*
D* Alternate security identifier
D  ODASI                225    264    INZ(X'00000000000000000-
D                                     00000000000000000000000000-
D                                     00000000000000000000000000-
D                                     0000000000000')
D*
D* Resolved queue name
```

```
D  ODRQN                265    312    INZ
D*
D* Resolved queue manager name
D  ODRMN                313    360    INZ
D*
D* reserved field
D  ODRE1                361    364I 0 INZ(0)
D*
D* reserved field
D  ODRS2                365    368I 0 INZ(0)
D*
D* Object long name
D* Address of variable length string
D  ODOSCHRP             369    384*   INZ(*NULL)
D* Offset of variable length string
D  ODOSCHRO             385    388I 0 INZ(0)
D* Size of buffer
D  ODOSVSBS             389    392I 0 INZ(-1)
D* Length of variable length string
D  ODOSCHRL             393    396I 0 INZ(0)
D* CCSID of variable length string
D  ODOSCHRC             397    400I 0 INZ(-3)
D*
D* Message Selector
D* Address of variable length string
D  ODSSCHRP             401    416*   INZ(*NULL)
D* Offset of variable length string
D  ODSSCHRO             417    420I 0 INZ(0)
D* Size of buffer
D  ODSSVSBS             421    424I 0 INZ(-1)
D* Length of variable length string
D  ODSSCHRL             425    428I 0 INZ(0)
D* CCSID of variable length string
D  ODSSCHRC             429    432I 0 INZ(-3)
D*
D* Resolved long object name
D* Address of variable length string
D  ODRSOCHRP            433    448*   INZ(*NULL)
D* Offset of variable length string
D  ODRSOCHRO            449    452I 0 INZ(0)
D* Size of buffer
D  ODRSOVSBS            453    456I 0 INZ(-1)
D* Length of variable length string
D  ODRSOCHRL            457    460I 0 INZ(0)
D* CCSID of variable length string
D  ODRSOCHRC            461    464I 0 INZ(-3)
D*
D* Alias queue resolved object type
D  ODRT                 465    468I 0 INZ(0)
```

IBM i MQOR (Object record) su IBM i

La struttura MQOR viene usata per specificare il nome della coda e il nome del gestore code di una singola coda di destinazione.

Panoramica

Scopo: MQOR è una struttura di immissione per le chiamate MQOPEN e MQPUT1 .

Serie di caratteri e codifica: i dati in MQOR devono essere nel set di caratteri fornito dall'attributo del gestore code **CodedCharSetId** e nella codifica del gestore code locale fornito da ENNAT. Tuttavia, se l'applicazione è in esecuzione come un client IBM MQ , la struttura deve essere nella serie di caratteri e nella codifica del client.

Utilizzo: fornendo un array di queste strutture sulla chiamata MQOPEN, è possibile aprire un elenco di code; questo elenco è denominato *elenco di distribuzione*. Ogni messaggio inserito utilizzando l'handle della coda restituito dalla chiamata MQOPEN viene posizionato su ciascuna delle code nell'elenco, se la coda è stata aperta correttamente.

- “Campi” a pagina 1203
- “Valori iniziali” a pagina 1203
- “Dichiarazione RPG” a pagina 1203

Campi

La struttura MQOR contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

ORMN (stringa di caratteri a 48 byte)

Nome gestore code oggetti.

È uguale al campo *ODMN* nella struttura MQOD (per i dettagli, consultare MQOD).

Questo è sempre un campo di input. Il valore iniziale di questo campo è di 48 caratteri vuoti.

ORON (stringa di caratteri a 48 byte)

Il nome dell'oggetto.

È uguale al campo *ODON* nella struttura MQOD (per i dettagli, consultare MQOD), tranne che:

- Deve essere il nome di una coda.
- Non deve essere il nome di una coda modello.

Questo è sempre un campo di input. Il valore iniziale di questo campo è di 48 caratteri vuoti.

Valori iniziali

Tabella 714. Campi in MQOR

Nome campo	Nome della costante	Valore della costante
ORON	Nessuna	Spazi
ORMN	Nessuna	Spazi

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQOR Structure
D*
D* Object name
D  ORON                   1     48    INZ
D* Object queue manager name
D  ORMN                  49     96    INZ
```

MQPD - Descrittore proprietà

MQPD viene utilizzato per definire gli attributi di una proprietà.

Panoramica

Scopo: La struttura è un parametro di input / output nella chiamata MQSETMP e un parametro di output nella chiamata MQINQMP.

Serie di caratteri e codifica: i dati in MQPD devono trovarsi nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (ENNAT).

- "Campi" a pagina 1203
- "Valori iniziali" a pagina 1206
- "Dichiarazione RPG" a pagina 1206

Campi

La struttura MQPD contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

PDCT (numero intero con segno a 10 cifre)

Descrive il contesto del messaggio a cui appartiene la proprietà.

Quando un gestore code riceve un messaggio contenente una proprietà definita da IBM MQche il gestore code riconosce come non corretta. il gestore code corregge il valore del campo *PDCT* .

È possibile specificare la seguente opzione:

PDUSC
La proprietà è associata al contesto utente.

Non è richiesta alcuna autorizzazione speciale per poter impostare una proprietà associata al contesto utente utilizzando la chiamata MQSETMP.

Se l'opzione precedentemente descritta non è richiesta, è possibile utilizzare la seguente opzione:

NODOP
La proprietà non è associata a un contesto di messaggio.

Un valore non riconosciuto viene rifiutato con un codice *PDREA* RC2482.

Si tratta di un campo di input / output per la chiamata MQSETMP e di un campo di output dalla chiamata MQINQMP. Il valore iniziale di questo campo è PDNOC.

PDCPYOPT (numero intero con segno a 10 cifre)

Descrive in quale tipo di messaggi deve essere copiata la proprietà.

Questo è un campo di sola emissione per le proprietà IBM MQ-defined riconosciute; IBM MQ imposta il valore appropriato.

Quando un gestore code riceve un messaggio contenente una proprietà definita da IBM MQche il gestore code riconosce come non corretta. il gestore code corregge il valore del campo *CopyOptions* .

È possibile specificare una o più di queste opzioni. Per specificare più di un'opzione, aggiungere i valori insieme (non aggiungere la stessa costante più di una volta) o combinare i valori utilizzando l'operazione OR bit per bit (se il linguaggio di programmazione supporta le operazioni bit).

COPFOR
Questa proprietà viene copiata in un messaggio inoltrato.

COPIPUB
Questa proprietà viene copiata nel messaggio ricevuto da un sottoscrittore quando viene pubblicato un messaggio.

COPREP
Questa proprietà viene copiata in un messaggio di risposta.

COPRP
Questa proprietà viene copiata in un messaggio di report.

COPALL
Questa proprietà viene copiata in tutti i messaggi successivi.

COPNON
Questa proprietà non viene copiata in un messaggio.

Opzione predefinita: è possibile specificare la seguente opzione per fornire la serie di opzioni di copia predefinita:

CODEF
Questa proprietà viene copiata in un messaggio inoltrato, in un messaggio di report o in un messaggio ricevuto da un sottoscrittore quando viene pubblicato un messaggio.

Ciò equivale a specificare la combinazione delle opzioni COPFOR, più COPRP, più COPPUB.

Se nessuna delle opzioni descritte in precedenza è richiesta, utilizzare la seguente opzione:

COPNON
Utilizzare questo valore per indicare che non sono state specificate altre opzioni di copia; programmaticamente non esiste alcuna relazione tra questa proprietà e i messaggi successivi. Viene sempre restituito per le proprietà del descrittore del messaggio.

Si tratta di un campo di input / output per la chiamata MQSETMP e di un campo di output dalla chiamata MQINQMP. Il valore iniziale di questo campo è COPDEF.

PDOPT (numero intero con segno a 10 cifre)

Il valore deve essere:

PDNONE
Nessuna opzione specificata

Questo è sempre un campo di input. Il valore iniziale di questo campo è PDNONE.

PDSID (numero intero con segno a 10 cifre)

Questo è l'identificatore della struttura; il valore deve essere:

PSIDV
Identificativo per la struttura del descrittore proprietà.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **PSIDV**.

PDSUP (numero intero con segno a 10 cifre)

Questo campo descrive il livello di supporto per la proprietà del messaggio richiesto dal gestore code, affinché il messaggio che contiene questa proprietà venga inserito in una coda. Ciò si applica solo alle proprietà definite da IBM MQ; il supporto per tutte le altre proprietà è facoltativo.

Il campo viene impostato automaticamente sul valore corretto quando la proprietà definita IBM MQè nota al gestore code. Se la proprietà non viene riconosciuta, viene assegnato PDSUPO. Quando un gestore code riceve un messaggio contenente una proprietà definita da IBM MQche il gestore code riconosce come non corretta. il gestore code corregge il valore del campo *PDSUP* .

Quando si imposta una proprietà definita da IBM MQutilizzando la chiamata MQSETMP su un handle del messaggio in cui è stata impostata l'opzione CMNOVA, *PDSUP* diventa un campo di input. Ciò consente a un'applicazione di inserire una proprietà definita da IBM MQ, con il valore corretto, in cui la proprietà non è supportata dal gestore code connesso, ma in cui il messaggio è destinato ad essere elaborato su un altro gestore code.

Il valore PDSUPO è sempre assegnato a proprietà che non sono IBM MQ-definite.

Uno dei seguenti valori viene restituito dalla chiamata MQINQMP o è possibile specificare uno dei seguenti valori quando si utilizza la chiamata MQSETMP su un handle del messaggio in cui è impostata l'opzione CMNOVA:

PDSUPO
La proprietà viene accettata da un gestore code anche se non è supportata. La proprietà può essere eliminata in modo che il messaggio possa fluire in un gestore code che non supporta le proprietà del messaggio. Questo valore viene assegnato anche a proprietà non definite da IBM MQ.

PDSUPR
È richiesto il supporto per la proprietà. Il messaggio è stato rifiutato da un gestore code che non supporta la proprietà definita da IBM MQ. La chiamata MQPUT o MQPUT1 ha esito negativo con codice di completamento CCFAIL e codice motivo RC2490.

PSUPL
Il messaggio viene rifiutato da un gestore code che non supporta la proprietà definita da IBM MQse il messaggio è destinato a una coda locale. La chiamata MQPUT o MQPUT1 ha esito negativo con codice di completamento CCFAIL e codice motivo RC2490.

La chiamata MQPUT o MQPUT1 ha esito positivo se il messaggio è destinato a un gestore code remoto.

Si tratta di un campo di output nella chiamata MQINQMP e di un campo di input nella chiamata MQSETMP se l'handle del messaggio è stato creato con l'opzione CMNOVA impostata. Il valore iniziale di questo campo è PDSUPO.

PDVER (numero intero con segno a 10 cifre)

Questo è il numero di versione della struttura; il valore deve essere:

PDVER1
Struttura descrittore della proprietà Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

PDVERC
La versione corrente della struttura descrittore della proprietà.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **PDVER1**.

Valori iniziali

Tabella 715. Campi in MQPD

Nome campo	Nome della costante	Valore della costante
PDSID	PDSIDV	'PD'
PDVER	PDVER1	1
PDOPT	PDNONE	0
PDSUP	PDSUPO	0
PDCT	NODOP	0
PDCPYOPT	CODEF	0

Dichiarazione RPG

```
D* MQDMHO Structure
D*
D*
D* Structure identifier
D  DMSID                  1      4    INZ('DMHO')
D*
D* Structure version number
D  DMVER                  5      8I 0 INZ(1)
D*
D* Options that control the action of MQDLTMH
D  DMOPT                  9     12I 0 INZ(0)
```

IBM i MQPMO (Put - message options) su IBM i

La struttura MQPMO consente all'applicazione di specificare le opzioni che controllano il modo in cui i messaggi vengono inseriti nelle code o pubblicati negli argomenti.

Panoramica

Finalità
La struttura è un parametro di input / output nelle chiamate MQPUT e MQPUT1 .

Versione
La versione corrente di MQPM è PMVER2. I campi che esistono solo nelle versioni più recenti della struttura sono identificati come tali nelle descrizioni che seguono.

Il file COPY fornito contiene la versione più recente di MQPMO supportata dall'ambiente, ma con il valore iniziale del campo *PMVER* impostato su PMVER1. Per utilizzare i campi che non sono presenti nella struttura version-1 , l'applicazione deve impostare il campo *PMVER* sul numero di versione della versione richiesta.

Serie di caratteri e codifica
I dati in MQPMO devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da ENNAT. Tuttavia, se l'applicazione è in esecuzione come un client IBM MQ , la struttura deve essere nella serie di caratteri e nella codifica del client.

- "Campi" a pagina 1207
- "Valori iniziali" a pagina 1221
- "Dichiarazione RPG" a pagina 1222

Campi

La struttura MQPMO contiene i seguenti campi; i campi sono descritti in ordine alfabetico:

PMCT (numero intero con segno a 10 cifre)

Gestione oggetto della coda di immissione.

Se viene specificato PMPASI o PMPASA, questo campo deve contenere l'handle della coda di immissione da cui vengono prese le informazioni di contesto da associare al messaggio da inserire.

Se PMPASI e PMPASA non vengono specificati, questo campo viene ignorato.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

PMIDC (numero intero con segno a 10 cifre)

Numero di messaggi che non è stato possibile inviare.

Questo è il numero di messaggi che non possono essere inviati alle code nell'elenco di distribuzione. Il conteggio include le code che non è stato possibile aprire e le code che sono state aperte correttamente ma per cui l'operazione di inserimento non è riuscita. Questo campo viene impostato anche quando si immette un messaggio in una singola coda che non è in un elenco di distribuzione.

Nota: Questo campo è impostato solo se il parametro **CMPCOD** nella chiamata MQPUT o MQPUT1 è CCOK o CCWARN; non è impostato se il parametro **CMPCOD** è CCFAIL.

Questo è un campo di output. Il valore iniziale di questo campo è 0. Questo campo non è impostato se *PMVER* è inferiore a PMVER2.

PMKDC (numero intero con segno a 10 cifre)

Numero di messaggi inviati correttamente alle code locali.

Questo è il numero di messaggi che la chiamata MQPUT o MQPUT1 corrente ha inviato correttamente alle code nell'elenco di distribuzione che sono code locali. Il conteggio non include i messaggi inviati alle code che si risolvono in code remote (anche se una coda di trasmissione locale viene utilizzata inizialmente per memorizzare il messaggio). Questo campo viene impostato anche quando si immette un messaggio in una singola coda che non è in un elenco di distribuzione.

Questo è un campo di output. Il valore iniziale di questo campo è 0. Questo campo non è impostato se *PMVER* è inferiore a PMVER2.

PMOPT (numero intero con segno di 10 cifre)

Opzioni che controllano l'azione di MQPUT e MQPUT1.

È possibile specificare uno o nessuno dei seguenti valori. Se ne è richiesto più di uno, è possibile aggiungere i valori (non aggiungere la stessa costante più di una volta). Le combinazioni non valide vengono annotate; qualsiasi altra combinazione è valida.

Opzioni di pubblicazione: le seguenti opzioni controllano il modo in cui i messaggi vengono pubblicati in un argomento.

PMSRTO

Tutte le informazioni inserite nei campi MDRQ e MDRM di MQMD di questa pubblicazione non vengono trasmesse ai sottoscrittori. Se questa opzione viene utilizzata con un'opzione di report che richiede una coda ReplyTo, la chiamata non riesce con RC2027 .

REST

La pubblicazione inviata deve essere conservata dal gestore code. Ciò consente al sottoscrittore di richiedere una copia di questa pubblicazione dopo l'ora in cui è stata pubblicata, utilizzando la chiamata MQSUBRQ. Consente inoltre di inviare una pubblicazione alle applicazioni che effettuano la loro sottoscrizione dopo il momento in cui è stata effettuata la pubblicazione, a meno che non scelgano di non inviarla utilizzando l'opzione SONEWP. Se a un'applicazione viene inviata una pubblicazione che è stata conservata, viene indicata dalla proprietà del messaggio mq.IsRetained di tale pubblicazione.

È possibile conservare solo una pubblicazione su ciascun nodo della struttura ad albero degli argomenti. Ciò significa che se esiste già una pubblicazione conservata per questo argomento, pubblicata da qualsiasi altra applicazione, viene sostituita con questa pubblicazione. È quindi preferibile evitare che più di un editore conservi i messaggi sullo stesso argomento.

Quando le pubblicazioni conservate vengono richieste da un sottoscrittore, la sottoscrizione utilizzata può contenere un carattere jolly nell'argomento, nel qual caso un numero di pubblicazioni conservate potrebbe corrispondere (su vari nodi nella struttura ad albero dell'argomento) e diverse pubblicazioni possono essere inviate all'applicazione richiedente. Consultare la descrizione della chiamata “MQSUBRQ - Richiesta di sottoscrizione” a pagina 815 per ulteriori dettagli.

Se questa opzione viene utilizzata e la pubblicazione non può essere conservata, il messaggio non viene pubblicato e la chiamata ha esito negativo con RC2479 .

Opzioni punto di sincronizzazione: le opzioni riportate di seguito sono relative alla partecipazione della chiamata MQPUT o MQPUT1 all'interno di un'unità di lavoro:

PMSYP

Inserire il messaggio con il controllo del punto di sincronizzazione.

La richiesta è di operare all'interno dei normali protocolli di unità di lavoro. Il messaggio non è visibile all'esterno dell'unità di lavoro fino a quando non viene eseguito il commit dell'unità di lavoro. Se viene eseguito il backout dell'unità di lavoro, il messaggio viene eliminato.

Se questa opzione e PMNSYP non vengono specificati, la richiesta di inserimento non si trova all'interno di un'unità di lavoro.

PMSYP non deve essere specificato con PMNSYP.

PMNSYP

Inserire il messaggio senza controllo del punto di sincronizzazione.

La richiesta è di operare al di fuori dei normali protocolli di unità di lavoro. Il messaggio è disponibile immediatamente e non può essere eliminato ripristinando un'unità di lavoro.

Se questa opzione e PMSYP non vengono specificati, la richiesta di inserimento non si trova all'interno di un'unità di lavoro.

PMNSYP non deve essere specificato con PMSYP.

Opzioni identificativo del messaggio e identificativo di correlazione: le seguenti opzioni richiedono al gestore code di creare un nuovo identificativo del messaggio o un nuovo identificativo di correlazione:

IDPMNM

Generare un nuovo identificativo di messaggio.

Questa opzione fa sì che il gestore code sostituisca il contenuto del campo *MDMID* in MQMD con un nuovo ID messaggio. Questo identificativo del messaggio viene inviato con il messaggio e restituito all'applicazione all'output dalla chiamata MQPUT o MQPUT1 .

Questa opzione può essere specificata anche quando il messaggio viene inserito in un elenco di distribuzione; consultare la descrizione del campo *PRMID* nella struttura MQPMR per i dettagli.

L'utilizzo di questa opzione allevia l'applicazione della necessità di reimpostare il campo *MDMID* su MINONE prima di ogni chiamata MQPUT o MQPUT1 .

IDPMN
Generare un nuovo identificativo di correlazione.

Questa opzione fa sì che il gestore code sostituisca il contenuto del campo *MDCID* in MQMD con un nuovo identificativo di correlazione. Questo identificativo di correlazione viene inviato con il messaggio e restituito all'applicazione all'output dalla chiamata MQPUT o MQPUT1 .

Questa opzione può essere specificata anche quando il messaggio viene inserito in un elenco di distribuzione; consultare la descrizione del campo *PRCID* nella struttura MQPMR per i dettagli.

PMNCID è utile in situazioni in cui l'applicazione richiede un identificativo di correlazione univoco.

Opzioni di gruppi e segmenti: la seguente opzione si riferisce all'elaborazione dei messaggi in gruppi e segmenti di messaggi logici. Queste definizioni potrebbero essere utili per comprendere l'opzione:

Messaggio fisico
Questa è l'unità di informazioni più piccola che può essere inserita o rimossa da una coda; spesso corrisponde alle informazioni specificate o richiamate in una singola chiamata MQPUT, MQPUT1o MQGET. Ogni messaggio fisico ha il proprio descrittore di messaggio (MQMD). Generalmente, i messaggi fisici sono distinti da valori differenti per l'identificativo del messaggio (campo*MDMID* in MQMD), sebbene ciò non venga applicato dal gestore code.

Messaggio logico
Questa è una singola unità di informazioni sull'applicazione. In assenza di restrizioni di sistema, un messaggio logico sarebbe lo stesso di un messaggio fisico. Ma quando i messaggi logici sono grandi, i vincoli di sistema possono rendere consigliabile o necessario suddividere un messaggio logico in due o più messaggi fisici, denominati *segmenti*.

Un messaggio logico che è stato segmentato è costituito da due o più messaggi fisici che hanno lo stesso identificativo del gruppo non null (campo*MDGID* in MQMD) e lo stesso numero di sequenza del messaggio (campo*MDSEQ* in MQMD). I segmenti sono distinti da valori differenti per l'offset del segmento (campo*MDOFF* in MQMD), che fornisce l'offset dei dati nel messaggio fisico dall'inizio dei dati nel messaggio logico. Poiché ogni segmento è un messaggio fisico, i segmenti in un messaggio logico generalmente hanno identificativi di messaggio differenti.

Un messaggio logico che non è stato segmentato, ma per cui la segmentazione è stata consentita dall'applicazione mittente, ha anche un identificativo di gruppo non null, anche se in questo caso esiste solo un messaggio fisico con tale identificativo di gruppo se il messaggio logico non appartiene a un gruppo di messaggi. I messaggi logici per i quali la segmentazione è stata inibita dall'applicazione di invio hanno un identificativo di gruppo nullo (GINONE), a meno che il messaggio logico non appartenga ad un gruppo di messaggi.

Gruppo di messaggi
Questa è una serie di uno o più messaggi logici che hanno lo stesso identificativo di gruppo non null. I messaggi logici nel gruppo sono distinti da valori differenti per il numero di sequenza del messaggio, che è un numero intero compreso tra 1 e n, dove n è il numero di messaggi logici nel gruppo. Se uno o più messaggi logici sono segmentati, nel gruppo sono presenti più di n messaggi fisici.

LOGO PM1
I messaggi in gruppi e segmenti di messaggi logici vengono inseriti in ordine logico.

Questa opzione indica al gestore code il modo in cui l'applicazione inserisce i messaggi in gruppi e segmenti di messaggi logici. Può essere specificato solo nella chiamata MQPUT; non è valido nella chiamata MQPUT1 .

Se PMLOGO è specificato, indica che l'applicazione utilizza successive chiamate MQPUT per:

- Inserire i segmenti in ciascun segmento logico nell'ordine di offset crescente dei segmenti, a partire da 0, senza intervalli.
- Inserire tutti i segmenti in un messaggio logico prima di inserire i segmenti nel successivo messaggio logico.
- Inserire i messaggi logici in ciascun gruppo di messaggi nell'ordine crescente del numero di sequenza dei messaggi, a partire da 1, senza intervalli.
- Inserire tutti i messaggi logici in un gruppo di messaggi prima di inserire i messaggi logici nel gruppo di messaggi successivo.

Questo ordine è chiamato "ordine logico".

Poiché l'applicazione ha indicato al gestore code il modo in cui inserisce i messaggi in gruppi e segmenti di messaggi logici, non è necessario che l'applicazione conservi e aggiorni le informazioni sul gruppo e sul segmento relative a ciascuna chiamata MQPUT, poiché il gestore code esegue questa operazione. In particolare, significa che l'applicazione non deve impostare i campi *MDGID*, *MDSEQ*e *MDOFF* in MQMD, poiché il gestore code li imposta sui valori appropriati. L'applicazione deve impostare solo il campo *MDMFL* in MQMD, per indicare quando i messaggi appartengono a gruppi o sono segmenti di messaggi logici e per indicare l'ultimo messaggio in un gruppo o l'ultimo segmento di un messaggio logico.

Una volta avviato un gruppo di messaggi o un messaggio logico, le chiamate MQPUT successive devono specificare gli indicatori MF* appropriati in *MDMFL* in MQMD. Se l'applicazione tenta di inserire un messaggio non in un gruppo quando è presente un gruppo di messaggi non terminati o inserisce un messaggio che non è un segmento quando è presente un messaggio logico non terminato, la chiamata ha esito negativo con il codice di errore RC2241 o RC2242 , come appropriato. Tuttavia, il gestore code conserva le informazioni sul gruppo di messaggi corrente o sul messaggio logico corrente e l'applicazione può terminarli inviando un messaggio (possibilmente senza dati del messaggio dell'applicazione) specificando MFLMIG o MFLSEG come appropriato, prima di emettere di nuovo la chiamata MQPUT per inserire il messaggio che non si trova nel gruppo o non in un segmento.

Tabella 716 a pagina 1211 mostra le combinazioni di opzioni e indicatori validi e i valori dei campi *MDGID*, *MDSEQ*e *MDOFF* utilizzati dal gestore code in ciascun caso. Le combinazioni di opzioni e indicatori non mostrate nella tabella non sono valide. Le colonne della tabella hanno i seguenti significati:

ORD LOG
Indica se l'opzione PMLOGO è specificata nella chiamata.

MIG
Indica se l'opzione MFMIG o MFLMIG è specificata nella chiamata.

SEG
Indica se nella chiamata è specificata l'opzione MFSEG o MFLSEG.

SEG - OK
Indica se l'opzione MFSEGA è specificata sulla chiamata.

Grp corrente
Indica se esiste un gruppo di messaggi corrente prima della chiamata.

Messaggio di log corrente
Indica se esiste un messaggio logico corrente prima della chiamata.

Altre colonne
Mostra i valori utilizzati dal gestore code. "Precedente" indica il valore utilizzato per il campo nel precedente messaggio per l'handle della coda.

PMRLOC
Specifica che il PMRQN nella struttura MQPMO deve essere completato con il nome della coda locale in cui viene effettivamente inserito il messaggio. Il nome ResolvedQMgrviene completato in modo simile con il nome del gestore code locale che ospita la coda locale.

Vedere OORLOQ per il significato di ciò. Se un utente è autorizzato per un inserimento in una coda, dispone dell'autorizzazione richiesta per specificare questo indicatore nella chiamata MQPUT. Non è necessaria alcuna autorizzazione speciale.

Tabella 716. Opzioni MQPUT relative a messaggi in gruppi e segmenti di messaggi logici

Opzioni specificate				**Stato messaggio di log e gruppo prima della chiamata**		**Valori utilizzati dal gestore code**		
ORD LOG	**MIG**	**SEG**	**SEG OK**	**Grp Cur**	**Messaggio log Cur**	*MDGID*	*MDSEQ*	*MDOFF*
Sì	No	No	No	No	No	GINONE	1	0
Sì	No	No	Sì	No	No	Nuovo ID gruppo	1	0
Sì	No	Sì	YES o NO	No	No	Nuovo ID gruppo	1	0
Sì	No	Sì	YES o NO	No	Sì	ID gruppo precedente	1	Offset precedente + lunghezza segmento precedente
Sì	Sì	YES o NO	YES o NO	No	No	Nuovo ID gruppo	1	0
Sì	Sì	YES o NO	YES o NO	Sì	No	ID gruppo precedente	Numero di sequenza precedente + 1	0
Sì	Sì	Sì	YES o NO	Sì	Sì	ID gruppo precedente	Numero sequenza precedente	Offset precedente + lunghezza segmento precedente
No	No	No	No	YES o NO	YES o NO	GINONE	1	0
No	No	No	Sì	YES o NO	YES o NO	Nuovo ID gruppo se GINONE, valore else nel campo	1	0
No	No	Sì	YES o NO	YES o NO	YES o NO	Nuovo ID gruppo se GINONE, valore else nel campo	1	Valore nel campo
No	Sì	No	YES o NO	YES o NO	YES o NO	Nuovo ID gruppo se GINONE, valore else nel campo	Valore nel campo	0
No	Sì	Sì	YES o NO	YES o NO	YES o NO	Nuovo ID gruppo se GINONE, valore else nel campo	Valore nel campo	Valore nel campo

Tabella 716. Opzioni MQPUT relative a messaggi in gruppi e segmenti di messaggi logici (Continua)

Opzioni specificate	Stato messaggio di log e gruppo prima della chiamata	Valori utilizzati dal gestore code

Nota:

- PMLOGO non è valido nella chiamata MQPUT1 .
- Per il campo *MDMID* , il gestore code genera un nuovo identificativo del messaggio se è specificato PMNMID o MINONE e utilizza il valore nel campo in caso contrario.
- Per il campo *MDCID* , il gestore code genera un nuovo identificativo di correlazione se viene specificato PMNCID e utilizza il valore nel campo in caso contrario.

Quando viene specificato PMLOGO, il gestore code richiede che tutti i messaggi in un gruppo e i segmenti in un messaggio logico vengano inseriti con lo stesso valore nel campo *MDPER* in MQMD, ovvero tutti devono essere persistenti o tutti non persistenti. Se questa condizione non viene soddisfatta, la chiamata MQPUT ha esito negativo con codice motivo RC2185 .

L'opzione PMLOGO influisce sulle unità di lavoro come segue:

- Se il primo messaggio fisico in un gruppo o in un messaggio logico viene inserito all'interno di un'unità di lavoro, tutti gli altri messaggi fisici nel gruppo o nel messaggio logico devono essere inseriti all'interno di un'unità di lavoro, se viene utilizzato lo stesso gestore code. Tuttavia, non devono essere inseriti nella stessa unità di lavoro. Ciò consente a un gruppo di messaggi o a un messaggio logico costituito da molti messaggi fisici di essere suddiviso in due o più unità di lavoro consecutive per la gestione della coda.
- Se il primo messaggio fisico in un gruppo o un messaggio logico non viene inserito all'interno di un'unità di lavoro, nessuno degli altri messaggi fisici nel gruppo o messaggio logico può essere inserito all'interno di un'unità di lavoro, se viene utilizzato lo stesso gestore code.

Se queste condizioni non vengono soddisfatte, la chiamata MQPUT ha esito negativo con il codice motivo RC2245 .

Quando viene specificato PMLOGO, l'MQMD fornito nella chiamata MQPUT non deve essere inferiore a MDVER2. Se questa condizione non viene soddisfatta, la chiamata ha esito negativo con codice di errore RC2257 .

Se PMLOGO non è specificato, i messaggi in gruppi e segmenti di messaggi logici possono essere inseriti in qualsiasi ordine e non è necessario inserire gruppi di messaggi completi o messaggi logici completi. È responsabilità dell'applicazione assicurarsi che i campi *MDGID*, *MDSEQ*, *MDOFF*e *MDMFL* abbiano valori appropriati.

Questa è la tecnica che può essere utilizzata per riavviare un gruppo di messaggi o un messaggio logico nel mezzo, dopo che si è verificato un errore di sistema. Quando il sistema viene riavviato, l'applicazione può impostare i campi *MDGID*, *MDSEQ*, *MDOFF*, *MDMFL*e *MDPER* sui valori appropriati e quindi emettere la chiamata MQPUT con PMSYP o PMNSYP impostati come *necessari*, ma senza specificare PMLOGO. Se questa chiamata ha esito positivo, il gestore code conserva le informazioni sul gruppo e sul segmento e le successive chiamate MQPUT che utilizzano tale gestore code possono specificare PMLOGO come normale.

Le informazioni sul gruppo e sul segmento che il gestore code conserva per la chiamata MQPUT sono separate dal gruppo e le informazioni sul segmento che conserva per la chiamata MQGET.

Per qualsiasi handle di coda fornito, l'applicazione è libera di combinare chiamate MQPUT che specificano PMLOGO con chiamate MQPUT che non lo fanno, ma i seguenti punti devono essere annotati:

- Se PMLOGO non è specificato, ogni chiamata MQPUT eseguita correttamente fa sì che il gestore code imposti le informazioni sul gruppo e sul segmento per l'handle della coda sui valori specificati dall'applicazione; ciò sostituisce le informazioni sul gruppo e sul segmento esistenti conservate dal gestore code per l'handle della coda.
- Se PMLOGO non è specificato, la chiamata non ha esito negativo se è presente un gruppo di messaggi corrente o un messaggio logico; la chiamata potrebbe tuttavia riuscire con un codice di completamento CCWARN. Tabella 717 a pagina 1213 mostra i vari casi che possono verificarsi. In questi casi, se il codice di completamento non è CCOK, il codice di errore è uno dei seguenti (come appropriato):
  - RC2241
  - RC2242
  - RC2185
  - RC2245

  Nota: Il gestore code non controlla le informazioni sul gruppo e sul segmento per la chiamata MQPUT1 .

Tabella 717. Risultato quando la chiamata MQPUT o MQCLOSE non è congruente con le informazioni sul gruppo e sul segmento

La chiamata corrente è	La chiamata precedente era MQPUT con PMLOGO	La chiamata precedente era MQPUT senza PMLOGO
MQPUT con PMLOGO	CCNON RIUSCITO	CCNON RIUSCITO
MQPUT senza PMLOGO	AVVCCN	CCOK
MQCLOSE con un gruppo non terminato o un messaggio logico	AVVCCN	CCOK

Le applicazioni che desiderano semplicemente inserire i messaggi e i segmenti in ordine logico sono consigliati per specificare PMLOGO, poiché questa è l'opzione più semplice da usare. Questa opzione allevia l'applicazione della necessità di gestire le informazioni sul gruppo e sul segmento, poiché il gestore code gestisce tali informazioni. Tuttavia, le applicazioni specializzate possono richiedere un controllo maggiore di quello fornito dall'opzione PMLOGO e ciò può essere ottenuto non specificando tale opzione. In questo caso, l'applicazione deve verificare che i campi *MDGID*, *MDSEQ*, *MDOFF*e *MDMFL* in MQMD siano impostati correttamente, prima di ogni chiamata MQPUT o MQPUT1 .

Ad esempio, un'applicazione che desidera inoltrare i messaggi fisici ricevuti, indipendentemente dal fatto che tali messaggi si trovino in gruppi o segmenti di messaggi logici, non deve specificare PMLOGO. Ci sono due ragioni per questo:

- Se i messaggi vengono richiamati e messi in ordine, specificando PMLOGO si determina l'assegnazione di un nuovo identificativo di gruppo ai messaggi e ciò potrebbe rendere difficile o impossibile per il creatore dei messaggi correlare eventuali messaggi di risposta o di report risultanti dal gruppo di messaggi.
- In una rete complessa con più percorsi tra i gestori code di invio e di ricezione, i messaggi fisici potrebbero arrivare fuori ordine. Non specificando PMLOGO e il GMLOGO corrispondente sulla chiamata MQGET, l'applicazione di inoltro può recuperare e inoltrare ogni messaggio fisico non appena arriva, senza dover attendere il successivo in ordine logico.

Le applicazioni che generano messaggi di report per i messaggi in gruppi o segmenti di messaggi logici non devono specificare PMLOGO durante l'inserimento del messaggio di report.

PMLOGO può essere specificato con una qualsiasi delle altre opzioni PM*.

Opzioni di contesto: le seguenti opzioni controllano l'elaborazione del contesto del messaggio:

NOC PM1
Nessun contesto deve essere associato al messaggio.

L'identità e il contesto di origine sono impostati in modo da non indicare alcun contesto. Ciò significa che i campi di contesto in MQMD sono impostati su:

- Spazi vuoti per i campi di caratteri
- Valori null per i campi byte
- Zeri per campi numerici

PMDEFC
Utilizza contesto predefinito.

Il messaggio deve avere informazioni di contesto predefinite associate ad esso, sia per l'identità che per l'origine. Il gestore code imposta i campi di contesto nel descrittore del messaggio nel modo seguente:

Tabella 718. Valori predefiniti delle informazioni di contesto per i campi MQMD

Campo in MQMD	Valore utilizzato
MDUID	Determinato dall'ambiente, se possibile; altrimenti, impostare su spazi vuoti.
MDACC	Determinato dall'ambiente, se possibile; altrimenti impostare su ACNONE.
MDAID	Impostare su spazi vuoti.
MDPAT	Determinato dall'ambiente.
MDPAN	Determinato dall'ambiente, se possibile; altrimenti, impostare su spazi vuoti.
MDPD	Impostare la data di inserimento del messaggio.
MDPT	Impostare l'ora in cui viene inserito il messaggio.
MDAOD	Impostare su spazi vuoti.

Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Questa è l'azione predefinita se non viene specificata alcuna opzione di contesto.

PMPASI
Passare il contesto di identità da un handle della coda di input.

Il messaggio deve essere associato alle informazioni di contesto. Il contesto di identità viene preso dall'handle di coda specificato nel campo *PMCT* . Le informazioni sul contesto di origine vengono generate dal gestore code nello stesso modo in cui vengono generate per PMDEFC (vedere la precedente tabella per i valori). Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Per la chiamata MQPUT, la coda deve essere stata aperta con l'opzione OOPASI (o un'opzione che la implica). Per la chiamata MQPUT1 , viene eseguito lo stesso controllo di autorizzazione della chiamata MQOPEN con l'opzione OOPASI.

PMPASA
Passare tutto il contesto da un handle di coda di immissione.

Il messaggio deve essere associato alle informazioni di contesto. Sia l'identità che il contesto di origine vengono presi dall'handle della coda specificato nel campo *PMCT* . Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Per la chiamata MQPUT, la coda deve essere stata aperta con l'opzione OOPASA (o un'opzione che la implica). Per la chiamata MQPUT1 , viene eseguito lo stesso controllo di autorizzazione della chiamata MQOPEN con l'opzione OOPASA.

PMSETI
Impostare il contesto di identità dall'applicazione.

Il messaggio deve essere associato alle informazioni di contesto. L'applicazione specifica il contesto di identità nella struttura MQMD. Le informazioni sul contesto di origine vengono generate dal gestore code nello stesso modo in cui vengono generate per PMDEFC (vedere la precedente tabella per i valori). Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Per la chiamata MQPUT, la coda deve essere stata aperta con l'opzione OOSETI (o un'opzione che la implica). Per la chiamata MQPUT1 , viene eseguito lo stesso controllo di autorizzazione della chiamata MQOPEN con l'opzione OOSETI.

PMSETA
Impostare tutto il contesto dall'applicazione.

Il messaggio deve essere associato alle informazioni di contesto. L'applicazione specifica l'identità e il contesto di origine nella struttura MQMD. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Per la chiamata MQPUT, la coda deve essere stata aperta con l'opzione OOSETA. Per la chiamata MQPUT1 , viene eseguito lo stesso controllo di autorizzazione della chiamata MQOPEN con l'opzione OOSETA.

È possibile specificare solo una delle opzioni di contesto PM*. Se nessuna di queste opzioni viene specificata, viene assunto PMDEFC.

Tipi di risposta Put. Le seguenti opzioni controllano la risposta restituita a una chiamata MQPUT o MQPUT1 . È possibile specificare solo una di queste opzioni. Se non si specificano PMARES e PMSRES, si presuppone PMRASQ o PMRAST.

PMARES

L'opzione PMARES richiede che un'operazione MQPUT o MQPUT1 venga completata senza che l'applicazione attenda che il gestore code completi la chiamata. L'utilizzo di questa opzione può migliorare le prestazioni della messaggistica, in particolare per le applicazioni che utilizzano i collegamenti client. Un'applicazione può controllare periodicamente, utilizzando il comando MQSTAT, se si è verificato un errore durante le chiamate asincrone precedenti.

Con questa opzione, solo i seguenti campi sono garantiti per essere completati in MQMD;

- ID MDA
- MDPAT
- MDPAN
- MDAOD

Inoltre, se uno o entrambi i valori PMNMID o PMNCID sono specificati come opzioni, vengono completati anche i valori MDMID e MDCID restituiti. (PMNMID può essere specificato implicitamente specificando un campo MDMID vuoto).

Vengono completati solo i campi precedentemente specificati. Altre informazioni che normalmente vengono restituite nella struttura MQMD o MQPMO non sono definite.

Quando si richiede una risposta di inserimento asincrona per MQPUT o MQPUT1, un CMPCOD e un MOTIVO di CCOK e RCNONE non indicano necessariamente che il messaggio è stato inserito correttamente in una coda. Quando si sviluppa un'applicazione MQI che utilizza una risposta di inserimento asincrona e richiede la conferma che i messaggi sono stati inseriti in una coda, è necessario controllare i codici CMPCOD e REASON dalle operazioni di inserimento e utilizzare MQSTAT per eseguire la query delle informazioni di errore asincrone.

Sebbene l'esito positivo o negativo di ogni singola chiamata MQPUT/MQPUT1 potrebbe non essere restituito immediatamente, il primo errore che si è verificato durante una chiamata asincrona può essere determinato in un momento successivo tramite una chiamata a MQSTAT.

Se un messaggio persistente nel punto di sincronizzazione non riesce a essere consegnato utilizzando la risposta put asincrona e si tenta di eseguire il commit della transazione, il commit non riesce e la transazione viene ripristinata con un codice di completamento CCFAIL e un motivo RC2003 . L'applicazione può effettuare una chiamata a MQSTAT per determinare la causa di un errore MQPUT o MQPUT1 precedente

PMSRES
Specificando questo valore per un'opzione di inserimento nella struttura MQPMO si garantisce che l'operazione MQPUT o MQPUT1 venga sempre emessa in modo sincrono. Se l'operazione ha esito positivo, vengono completati tutti i campi in MQMD e MQPMO. Viene fornito per garantire una risposta sincrona indipendentemente dal valore di risposta di inserimento predefinito definito sulla coda o sull'oggetto argomento.

Coda PMR
Se questo valore viene specificato per una chiamata MQPUT, il tipo di risposta di inserimento utilizzato viene preso dal valore DEFPRESP specificato sulla coda quando è stato aperto dall'applicazione. Se un'applicazione client è connessa a un gestore code a un livello precedente a IBM WebSphere MQ 7.0, si comporta come se fosse stato specificato PMSRES.

Se questa opzione viene specificata per una chiamata MQPUT1 , il valore DEFPRESP dalla definizione della coda non viene utilizzato. Se la chiamata MQPUT1 utilizza PMSYP, si comporta come per PMARES e se utilizza PMNSYP si comporta come per PMSRES.

PMRAST
Questo è un sinonimo di PMRASQ da utilizzare con gli oggetti argomento.

Altre opzioni: le seguenti opzioni controllano il controllo dell'autorizzazione e cosa accade quando il gestore code è in fase di sospensione:

PMALTU
Convalidare con l'identificativo utente specificato.

Ciò indica che il campo *ODAU* nel parametro **OBJDSC** della chiamata MQPUT1 contiene un identificativo utente da utilizzare per convalidare l'autorità per inserire i messaggi nella coda. La chiamata può avere esito positivo solo se questo *ODAU* è autorizzato ad aprire la coda con le opzioni specificate, indipendentemente dal fatto che l'identificativo utente con cui l'applicazione è in esecuzione sia autorizzato a farlo. (Ciò non si applica alle opzioni di contesto specificate, tuttavia, che vengono sempre controllate rispetto all'identificativo utente con cui è in esecuzione l'applicazione.)

Questa opzione è valida solo con la chiamata MQPUT1 .

PMFIQ
Errore se il gestore code è in fase di sospensione.

Questa opzione forza la chiamata MQPUT o MQPUT1 ad avere esito negativo se il gestore code è in stato di sospensione.

La chiamata restituisce il codice di completamento CCFAIL con codice motivo RC2161 .

Opzione predefinita: se nessuna delle opzioni descritte in precedenza è richiesta, è possibile utilizzare la seguente opzione:

PMNONE
Nessuna opzione specificata.

Questo valore può essere utilizzato per indicare che non sono state specificate altre opzioni; tutte le opzioni assumono i valori predefiniti. PMNONE è definito per aiutare la documentazione del programma; non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Questo è un campo di immissione. Il valore iniziale del campo *PMOPT* è PMNONE.

PMPRF (numero intero con segno a 10 cifre)

Indicatori che indicano quali campi MQPMR sono presenti.

Questo campo contiene gli indicatori che devono essere impostati per indicare quali campi di MQPMR sono presenti nei record dei messaggi di inserimento forniti dall'applicazione. *PMPRF* viene utilizzato solo quando il messaggio viene inserito in un elenco di distribuzione. Il campo viene ignorato se *PMREC* è zero o se entrambi *PMPRO* e *PMPRP* sono zero.

Per i campi presenti, il gestore code utilizza per ogni destinazione i valori dei campi nel corrispondente record del messaggio di inserimento. Per i campi assenti, il gestore code utilizza i valori della struttura MQMD.

È possibile specificare uno o più dei seguenti indicatori per indicare quali campi sono presenti nei record dei messaggi di inserimento:

IDPFM
Il campo identificativo messaggio è presente.

IDPFC
Il campo Identificativo correlazione è presente.

IDGFP
Il campo identificativo gruppo è presente.

FPFB
Il campo Feedback è presente.

FPACC
Il campo Accounting - token è presente.

Se viene specificato questo indicatore, è necessario specificare PMSETI o PMSETA nel campo *PMOPT* ; se questa condizione non viene soddisfatta, la chiamata ha esito negativo con codice di errore RC2158 .

Se non è presente alcun campo MQPMR, è possibile specificare quanto segue:

FPNONE
Non sono presenti campi record di messaggi di immissione.

Se questo valore viene specificato, *PMREC* deve essere zero o entrambi *PMPRO* e *PMPRP* devono essere zero.

PFNONE è definito per aiutare la documentazione del programma. Non è previsto che questa costante venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Se *PMPRF* contiene indicatori non validi o se vengono forniti record di messaggi di inserimento, ma *PMPRF* ha il valore PFNONE, la chiamata ha esito negativo con codice di errore RC2158 .

Questo è un campo di immissione. Il valore iniziale di questo campo è PFNONE. Questo campo viene ignorato se *PMVER* è minore di PMVER2.

PMPRO (numero intero con segno a 10 cifre)

Offset del primo record del messaggio di inserimento dall'inizio di MQPMO.

Questo è l'offset in byte del primo record del messaggio di inserimento MQPMR dall'inizio della struttura MQPMO. L'offset può essere positivo o negativo. *PMPRO* viene utilizzato solo quando il messaggio viene inserito in un elenco di distribuzione. Il campo viene ignorato se *PMREC* è zero.

Quando il messaggio viene inserito in un elenco di distribuzione, è possibile fornire un array di uno o più record di messaggi di inserimento MQPMR per specificare determinate proprietà del messaggio per ciascuna destinazione singolarmente; queste proprietà sono:

- identificativo del messaggio
- identificativo di correlazione
- identificativo gruppo

- valore di feedback
- token di accounting

Non è necessario specificare tutte queste proprietà, ma indipendentemente dal sottoinsieme scelto, i campi devono essere specificati nell'ordine corretto. Per ulteriori dettagli, consultare la descrizione della struttura MQPMR.

Di solito, il numero di record di messaggi di inserimento deve essere pari al numero di record di oggetti specificati da MQOD quando l'elenco di distribuzione viene aperto; ogni record di messaggi di inserimento fornisce proprietà di messaggio per la coda identificata dal record di oggetto corrispondente. Le code nell'elenco di distribuzione che non riescono ad aprirsi devono avere ancora i record dei messaggi assegnati nelle posizioni appropriate nella schiera, anche se in questo caso le proprietà del messaggio vengono ignorate.

È possibile che il numero di record messaggio di inserimento differisca dal numero di record oggetto. Se il numero di record del messaggio di inserimento è inferiore a quello dei record dell'oggetto, le proprietà del messaggio per le destinazioni che non hanno record del messaggio di inserimento vengono prese dai corrispondenti campi nel descrittore del messaggio MQMD. Se ci sono più record di messaggi di inserimento che record di oggetti, gli eccessi non vengono utilizzati (anche se deve essere ancora possibile accedervi). I record dei messaggi di inserimento sono facoltativi, ma se vengono forniti devono essere *PMREC* .

I record dei messaggi di inserimento possono essere forniti in modo simile ai record degli oggetti in MQOD, specificando un offset in *PMPRO*o specificando un indirizzo in *PMPRP* ; per i dettagli su come eseguire questa operazione, consultare il campo *ODORO* descritto in “MQOD (Object descriptor) su IBM i” a pagina 1192.

Non è possibile utilizzare più di uno tra *PMPRO* e *PMPRP* ; la chiamata ha esito negativo con codice motivo RC2159 se entrambi sono diversi da zero.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *PMVER* è minore di PMVER2.

PMPRP (puntatore)

Indirizzo del primo record del messaggio inserito.

Questo è l'indirizzo del primo record del messaggio di inserimento MQPMR. *PMPRP* viene utilizzato solo quando il messaggio viene inserito in un elenco di distribuzione. Il campo viene ignorato se *PMREC* è zero.

È possibile utilizzare *PMPRP* o *PMPRO* per specificare i record dei messaggi di inserimento, ma non entrambi; per i dettagli, consultare la descrizione del campo PMRRO . Se *PMPRP* non viene utilizzato, deve essere impostato sul puntatore null o sui byte null.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null. Questo campo viene ignorato se *PMVER* è minore di PMVER2.

PMREC (numero intero con segno a 10 cifre)

Numero di record di messaggi di inserimento o di risposta presenti.

Questo è il numero di record di messaggi di inserimento MQPMR o di risposta MQRR forniti dall'applicazione. Questo numero può essere maggiore di zero solo se il messaggio viene inserito in un elenco di distribuzione. I record di messaggi di inserimento e i record di risposta sono facoltativi - l'applicazione non deve necessariamente fornire alcun record oppure può scegliere di fornire record di un solo tipo. Tuttavia, se l'applicazione fornisce record di entrambi i tipi, deve fornire record *PMREC* di ciascun tipo.

Il valore di *PMREC* non deve necessariamente essere uguale al numero di destinazioni nell'elenco di distribuzione. Se vengono forniti troppi record, l'eccesso non viene utilizzato; se viene fornito un numero troppo basso di record, vengono utilizzati i valori predefiniti per le proprietà del messaggio per le destinazioni che non hanno inserito record di messaggi (consultare *PMPRO* più avanti in questo argomento).

Se *PMREC* è inferiore a zero o è maggiore di zero ma il messaggio non viene inserito in un elenco di distribuzione, la chiamata ha esito negativo con codice di errore RC2154 .

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *PMVER* è minore di PMVER2.

PMRMN (stringa di caratteri a 48 byte)

Nome risolto del gestore code di destinazione.

Questo è il nome del gestore code di destinazione dopo che la risoluzione dei nomi è stata eseguita dal gestore code locale. Il nome restituito è il nome del gestore code proprietario della coda identificata da *PMRQN*e può essere il nome del gestore code locale.

Se *PMRQN* è una coda condivisa di proprietà del gruppo di condivisione code a cui appartiene il gestore code locale, *PMRMN* è il nome del gruppo di condivisione code. Se la coda è di proprietà di un altro gruppo di condivisione code, *PMRQN* può essere il nome del gruppo di condivisione code o il nome di un gestore code che è un membro del gruppo di condivisione code (la natura del valore restituito è determinata dalle definizioni di coda che esistono nel gestore code locale).

Un valore non vuoto viene restituito solo se l'oggetto è una singola coda; se l'oggetto è un elenco di distribuzioni o un argomento, il valore restituito non è definito.

Questo è un campo di output. La lunghezza di questo campo è fornita da LNQMN. Il valore iniziale di questo campo è di 48 caratteri vuoti.

PMRQN (stringa di caratteri a 48 byte)

Nome risolto della coda di destinazione.

Questo è il nome della coda di destinazione dopo che la risoluzione del nome è stata eseguita dal gestore code locale. Il nome restituito è il nome di una coda che esiste sul gestore code identificato da *PMRMN*.

Un valore non vuoto viene restituito solo se l'oggetto è una singola coda; se l'oggetto è un elenco di distribuzioni o un argomento, il valore restituito non è definito.

Questo è un campo di output. La lunghezza di questo campo è fornita da LNQN. Il valore iniziale di questo campo è di 48 caratteri vuoti.

PMRRO (numero intero con segno a 10 cifre)

Offset del primo record di risposta dall'inizio di MQPMO.

Questo è l'offset in byte del primo record di risposta MQRR dall'inizio della struttura MQPMO. L'offset può essere positivo o negativo. *PMRRO* viene utilizzato solo quando il messaggio viene inserito in un elenco di distribuzione. Il campo viene ignorato se *PMREC* è zero.

Quando il messaggio viene inserito in un elenco di distribuzione, è possibile fornire un array di uno o più record di risposta MQRR per identificare le code a cui il messaggio non è stato inviato correttamente (campo*RRCC* in MQRR) e il motivo di ciascun errore (campo*RRREA* in MQRR). Il messaggio potrebbe non essere stato inviato perché non è stato possibile aprire la coda o perché l'operazione di inserimento non è riuscita. Il gestore code imposta i record di risposta solo quando il risultato della chiamata è misto (ossia, alcuni messaggi sono stati inviati correttamente mentre altri non sono riusciti o tutti non sono riusciti, ma per motivi diversi); il codice motivo RC2136 dalla chiamata indica questo caso. Se lo stesso codice di errore si applica a tutte le code, tale motivo viene restituito nel parametro **REASON** della chiamata MQPUT o MQPUT1 e i record di risposta non vengono impostati.

Di solito, ci devono essere tanti record di risposta quanti sono i record di oggetto specificati da MQOD quando l'elenco di distribuzione viene aperto; quando necessario, ogni record di risposta è impostato sul codice di completamento e sul codice motivo per l'inserimento nella coda identificata dal record di oggetto corrispondente. Le code nell'elenco di distribuzione che non riescono ad aprirsi devono avere ancora i record di risposta allocati nelle posizioni appropriate nella schiera, anche se sono impostati sul codice di completamento e sul codice di errore risultanti dall'operazione di apertura, piuttosto che sull'operazione di inserimento.

È possibile che il numero di record di risposta differisca dal numero di record oggetto. Se il numero di record di risposta è inferiore a quello dei record di oggetto, potrebbe non essere possibile per l'applicazione identificare tutte le destinazioni per cui l'operazione di inserimento non è riuscita o le cause degli errori. Se ci sono più record di risposta che record di oggetto, gli eccessi non vengono utilizzati (anche se deve essere ancora possibile accedervi). I record di risposta sono facoltativi, ma se vengono forniti devono essere *PMREC* .

I record di risposta possono essere forniti in modo simile ai record di oggetto in MQOD, specificando un offset in *PMRRO*o specificando un indirizzo in *PMRRP* ; per i dettagli su come eseguire questa operazione, consultare il campo *ODORO* descritto in “MQOD (Object descriptor) su IBM i” a pagina 1192. Tuttavia, non è possibile utilizzare più di uno tra *PMRRO* e *PMRRP* ; la chiamata ha esito negativo con codice motivo RC2156 se entrambi sono diversi da zero.

Per la chiamata MQPUT1 , questo campo deve essere zero. Ciò è dovuto al fatto che le informazioni di risposta (se richieste) vengono restituite nei record di risposta specificati dal descrittore dell'oggetto MQOD.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0. Questo campo viene ignorato se *PMVER* è minore di PMVER2.

PMRRP (puntatore)

Indirizzo del primo record di risposta.

Questo è l'indirizzo del primo record di risposta MQRR. *PMRRP* viene utilizzato solo quando il messaggio viene inserito in un elenco di distribuzione. Il campo viene ignorato se *PMREC* è zero.

È possibile utilizzare *PMRRP* o *PMRRO* per specificare i record di risposta, ma non entrambi; per i dettagli, consultare la descrizione del campo PMRRO . Se *PMRRP* non viene utilizzato, deve essere impostato sul puntatore null o sui byte null.

Per la chiamata MQPUT1 , questo campo deve essere il puntatore null o byte null. Ciò è dovuto al fatto che le informazioni di risposta (se richieste) vengono restituite nei record di risposta specificati dal descrittore dell'oggetto MQOD.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null. Questo campo viene ignorato se *PMVER* è minore di PMVER2.

PMSID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

SIDV
Identificativo per la struttura delle opzioni del messaggio di inserimento.

Questo è sempre un campo di input. Il valore iniziale di questo campo è PMSIDV.

PMSL (MQLONG)

Il livello di sottoscrizione indicato da questa pubblicazione.

Solo le sottoscrizioni con il valore *PMSL* minore o uguale a questo valore ricevono questa pubblicazione. Questo valore deve essere compreso tra zero e 9; zero è il livello più basso.

Il valore iniziale di questo campo è 9.

PMTO (numero intero con segno a 10 cifre)

Riservato.

Questo è un campo riservato; il suo valore non è significativo. Il valore iniziale di questo campo è -1.

PMUDC (numero intero con segno a 10 cifre)

Numero di messaggi inviati correttamente alle code remote.

Questo è il numero di messaggi che la chiamata MQPUT o MQPUT1 corrente ha inviato correttamente alle code nell'elenco di distribuzione che si risolvono in code remote. I messaggi che il gestore

code conserva temporaneamente nel modulo dell'elenco di distribuzione vengono conteggiati come numero di singole destinazioni contenute in tali elenchi di distribuzione. Questo campo viene impostato anche quando si immette un messaggio in una singola coda che non è in un elenco di distribuzione.

Questo è un campo di output. Il valore iniziale di questo campo è 0. Questo campo non è impostato se *PMVER* è inferiore a PMVER2.

PMVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere uno dei seguenti.

PMVER1
Version-1 struttura di opzioni put - message.

PMVER2
Version-2 struttura delle opzioni put - message.

I campi esistenti solo nella versione più recente della struttura vengono identificati come tali nelle descrizioni dei campi. La seguente costante specifica il numero di versione della versione corrente:

PMVERC
Versione corrente della struttura delle opzioni put - message.

Questo è sempre un campo di input. Il valore iniziale di questo campo è PMVER1.

Valori iniziali

Tabella 719. Campi in MQPMO

Nome campo	Nome della costante	Valore della costante
PMSID	SIDV	'PMO¬'
PMVER	PMVER1	1
PMOPT	PMNONE	0
PMTO	Nessuna	-1
PMCT	Nessuna	0
PMKDC	Nessuna	0
PMUDC	Nessuna	0
PMIDC	Nessuna	0
PMRQN	Nessuna	Spazi
PMRMN	Nessuna	Spazi
PMREC	Nessuna	0
PMPRF	FPNONE	0
PMPRO	Nessuna	0
PMRRO	Nessuna	0
PMPRP	Nessuna	Puntatore null o byte null
PMRRP	Nessuna	Puntatore null o byte null

Nota:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQPMO Structure
D*
D* Structure identifier
D  PMSID                  1      4    INZ('PMO ')
D* Structure version number
D  PMVER                  5      8I 0 INZ(1)
D* Options that control the action of MQPUT and MQPUT1
D  PMOPT                  9     12I 0 INZ(0)
D* Reserved
D  PMTO                  13     16I 0 INZ(-1)
D* Object handle of input queue
D  PMCT                  17     20I 0 INZ(0)
D* Number of messages sent successfully to local queues
D  PMKDC                 21     24I 0 INZ(0)
D* Number of messages sent successfully to remote queues
D  PMUDC                 25     28I 0 INZ(0)
D* Number of messages that could notbe sent
D  PMIDC                 29     32I 0 INZ(0)
D* Resolved name of destination queue
D  PMRQN                 33     80    INZ
D* Resolved name of destination queue manager
D  PMRMN                 81    128    INZ
D* Number of put message records or response records present
D  PMREC                129    132I 0 INZ(0)
D* Flags indicating which MQPMR fields are present
D  PMPRF                133    136I 0 INZ(0)
D* Offset of first put message record from start of MQPMO
D  PMPRO                137    140I 0 INZ(0)
D* Offset of first response record from start of MQPMO
D  PMRRO                141    144I 0 INZ(0)
D* Address of first put message record
D  PMPRP                145    160*   INZ(*NULL)
D* Address of first response record
D  PMRRP                161    176*   INZ(*NULL)
D* Original message handle
D  PMOMH                177    184I 0
D* New message handle
D  PMNMH                185    190I 0
D* The action being performed
D  PMACT                191    194I 0
D* Reserved
D  PMRE1                195    198I 0
```

IBM i MQPMR (Put - message record) su IBM i

La struttura MQPMR viene utilizzata per specificare diverse proprietà del messaggio per una singola destinazione quando un messaggio viene inserito in un elenco di distribuzione.

Panoramica

Scopo: MQPMR è una struttura di input / output per le chiamate MQPUT e MQPUT1 .

Serie di caratteri e codifica: i dati in MQPMR devono essere nel set di caratteri fornito dall'attributo del gestore code **CodedCharSetId** e nella codifica del gestore code locale fornito da ENNAT. Tuttavia, se l'applicazione è in esecuzione come un client IBM MQ , la struttura deve essere nella serie di caratteri e nella codifica del client.

Utilizzo: fornendo un array di queste strutture nella chiamata MQPUT o MQPUT1 , è possibile specificare valori differenti per ciascuna coda di destinazione in un elenco di distribuzione. Alcuni dei campi sono solo di immissione, altri sono di immissione / emissione.

Nota: Questa struttura è insolita in quanto non ha un layout fisso. I campi in questa struttura sono facoltativi e la presenza o l'assenza di ciascun campo è indicata dagli indicatori nel campo *PMPRF* in MQPMO. I campi presenti in ***devono essere nel seguente ordine*** :

- *PRMID*
- *PRCID*

- *PRGID*
- *PRFB*
- *PRACC*

I campi assenti non occupano spazio nel record.

Poiché MQPMR non ha un layout fisso, non viene fornita alcuna sua definizione nel file COPY. Il programmatore dell'applicazione deve creare una dichiarazione contenente i campi richiesti dall'applicazione e impostare gli indicatori in *PMPRF* per indicare i campi presenti.

- "Campi" a pagina 1223
- "Valori iniziali" a pagina 1224
- "Dichiarazione RPG" a pagina 1224

Campi

La struttura MQPMR contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

PRACC (stringa bit a 32 byte)

Token di account.

Questo è il token di account da utilizzare per il messaggio inviato alla coda con un nome specificato dall'elemento corrispondente nell'array di strutture MQOR fornito nella chiamata MQOPEN o MQPUT1 . Viene elaborato nello stesso modo del campo *MDACC* in MQMD per un inserimento in una singola coda. Consultare la descrizione di *MDACC* in "MQMD (Message Descriptor) su IBM i" a pagina 1140 per informazioni sul contenuto di questo campo.

Se questo campo non è presente, viene utilizzato il valore in MQMD.

Questo è un campo di immissione.

PRCID (stringa bit a 24 byte)

Identificativo di correlazione.

Questo è l'identificativo di correlazione da utilizzare per il messaggio inviato alla coda con il nome specificato dall'elemento corrispondente nell'array di strutture MQOR fornito sulla chiamata MQOPEN o MQPUT1 . Viene elaborato nello stesso modo del campo *MDCID* in MQMD per un inserimento in una singola coda.

Se questo campo non è presente in un record MQPMR o se il numero di record MQPMR è inferiore rispetto alle destinazioni, il valore in MQMD viene utilizzato per quelle destinazioni che non hanno un record MQPMR contenente un campo *PRCID* .

Se viene specificato PMNCID, viene generato un *singolo* nuovo identificativo di correlazione che viene utilizzato per tutte le destinazioni nell'elenco di distribuzione, indipendentemente dal fatto che abbiano o meno record MQPMR. Ciò è diverso dal modo in cui PMNMID viene elaborato (consultare campo *PRMID*).

Questo è un campo di immissione / emissione.

PRFB (numero intero con segno a 10 cifre)

Feedback o codice di errore.

Questo è il codice di feedback da utilizzare per il messaggio inviato alla coda con il nome specificato dall'elemento corrispondente nell'array delle strutture MQOR fornito sulla chiamata MQOPEN o MQPUT1 . Viene elaborato nello stesso modo del campo *MDFB* in MQMD per un inserimento in una singola coda.

Se questo campo non è presente, viene utilizzato il valore in MQMD.

Questo è un campo di immissione.

PRGID (stringa bit a 24 byte)

Identificativo gruppo.

Questo è l'identificativo del gruppo da utilizzare per il messaggio inviato alla coda con il nome specificato dall'elemento corrispondente nell'array di strutture MQOR fornito nella chiamata MQOPEN o MQPUT1 . Viene elaborato nello stesso modo del campo *MDGID* in MQMD per un inserimento in una singola coda.

Se questo campo non è presente in un record MQPMR o se il numero di record MQPMR è inferiore rispetto alle destinazioni, il valore in MQMD viene utilizzato per quelle destinazioni che non hanno un record MQPMR contenente un campo *PRGID* . Il valore viene elaborato come documentato in Tabella 716 a pagina 1211, ma con le seguenti differenze:

- Nei casi in cui viene utilizzato un nuovo identificativo di gruppo, il gestore code genera un identificativo di gruppo differente per ciascuna destinazione (ossia, non vi sono due destinazioni con lo stesso identificativo di gruppo).
- Nei casi in cui viene utilizzato il valore nel campo, la chiamata non riesce con il codice di errore RC2258.

Questo è un campo di immissione / emissione.

PRMID (stringa bit a 24 byte)

L'identificativo del messaggio.

Questo è l'identificativo del messaggio da utilizzare per il messaggio inviato alla coda con il nome specificato dall'elemento corrispondente nell'array di strutture MQOR fornito nella chiamata MQOPEN o MQPUT1 . Viene elaborato nello stesso modo del campo *MDMID* in MQMD per un inserimento in una singola coda.

Se questo campo non è presente in un record MQPMR o se il numero di record MQPMR è inferiore rispetto alle destinazioni, il valore in MQMD viene utilizzato per quelle destinazioni che non hanno un record MQPMR contenente un campo *PRMID* . Se tale valore è MINONE, viene generato un nuovo identificatore di messaggio per *ogni* di tali destinazioni (ossia, non due di tali destinazioni hanno lo stesso identificatore di messaggio).

Se viene specificato PMNMID, vengono creati nuovi identificatori di messaggi per tutte le destinazioni nell'elenco di distribuzione, indipendentemente dal fatto che dispongano o meno di record MQPMR. Ciò è diverso dal modo in cui PMNCID viene elaborato (vedere campo *PRCID*).

Questo è un campo di immissione / emissione.

Valori iniziali

Non ci sono valori iniziali definiti per questa struttura, poiché non viene fornita alcuna dichiarazione di struttura. La seguente dichiarazione di esempio mostra come la struttura deve essere dichiarata dal programmatore dell'applicazione se tutti i campi sono richiesti.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQPMR Structure
D*
D* Message identifier
D  PRMID                  1     24
D* Correlation identifier
D  PRCID                 25     48
D* Group identifier
D  PRGID                 49     72
D* Feedback or reason code
D  PRFB                  73     76I 0
D* Accounting token
D  PRACC                 77    108
```

IBM i MQRFH (regole e intestazione di formattazione) su IBM i

La struttura MQRFH definisce il formato delle regole e l'intestazione di formattazione.

Panoramica

Scopo: Questa intestazione può essere utilizzata per inviare dati stringa sotto forma di coppie nome - valore.

Nome formato: FMRFH.

Serie di caratteri e codifica: i campi nella struttura MQRFH (incluso *RFNVS*) si trovano nella serie di caratteri e nella codifica fornita dai campi *MDCSI* e *MDENC* nella struttura dell'intestazione che precede MQRFH o da quei campi nella struttura MQMD se MQRFH si trova all'inizio dei dati del messaggio dell'applicazione.

La serie di caratteri deve avere caratteri a byte singolo per i caratteri validi nei nomi di coda.

- "Campi" a pagina 1225
- "Valori iniziali" a pagina 1227
- "Dichiarazione RPG" a pagina 1227

Campi

La struttura MQRFH contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

RFCSI (numero intero con segno a 10 cifre)

Identificativo della serie di caratteri dei dati che seguono *RFNVS*.

Specifica l'identificativo della serie di caratteri dei dati che seguono *RFNVS* ; non si applica ai dati carattere nella stessa struttura MQRFH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. È possibile utilizzare il seguente valore speciale:

CINHT
Eredita l'identificativo della serie di caratteri di questa struttura.

I dati carattere nei dati *che seguono* questa struttura si trovano nella stessa serie di caratteri di questa struttura.

Il gestore code modifica questo valore nella struttura inviata nel messaggio nell'effettivo identificativo della serie di caratteri della struttura. Se non si verifica alcun errore, il valore CSINHT non viene restituito dalla chiamata MQGET.

CSINHT non può essere utilizzato se il valore del campo *MDPAT* in MQMD è ATBRKR.

Il valore iniziale di questo campo è CSUNDF.

Codifica numerica dei dati che seguono *RFNVS*.

Specifica la codifica numerica dei dati che seguono *RFNVS* ; non si applica ai dati numerici nella stessa struttura MQRFH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati.

Il valore iniziale di questo campo è ENNAT.

RFFLG (numero intero con segno a 10 cifre)

Indicatori.

È possibile specificare quanto segue:

RFNONE
Nessun indicatore.

Il valore iniziale di questo campo è RFNONE.

RFFMT (stringa di caratteri a 8 byte)

Nome formato dei dati che seguono *RFNVS*.

Specifica il nome del formato dei dati che seguono *RFNVS*.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono le stesse del campo *MDFMT* in MQMD.

Il valore iniziale di questo campo è FMNONE.

RFLEN (numero intero con segno a 10 cifre)

Lunghezza totale di MQRFH incluso *RFNVS*.

Questa è la lunghezza, in byte, della struttura MQRFH, incluso il campo *RFNVS* alla fine della struttura. La lunghezza non include i dati utente che seguono il campo *RFNVS* .

Per evitare problemi con la conversione dei dati utente in alcuni ambienti, considerare l'utilizzo di *RFLEN* come multiplo di quattro.

La seguente costante fornisce la lunghezza della parte *fissa* della struttura, ovvero la lunghezza escluso il campo *RFNVS* :

RFLENV
Lunghezza della parte fissa della struttura MQRFH.

Il valore iniziale di questo campo è RFLENV.

RFNVS (stringa di caratteri n byte)

Stringa contenente coppie nome - valore.

Questa è una stringa di caratteri a lunghezza variabile contenente coppie nome - valore nel formato:

```
name1 value1 name2 value2 name3 value3 ...
```

Ogni nome o valore deve essere separato dal nome o valore adiacente da uno o più caratteri vuoti; questi spazi vuoti non sono significativi. Un nome o un valore può contenere spazi vuoti significativi prefissando e aggiungendo un suffisso al nome o al valore con il carattere virgolette; tutti i caratteri compresi tra le virgolette di apertura e le virgolette di chiusura corrispondenti vengono considerati significativi. Nel seguente esempio, il nome è FAMOUS_WORDSe il valore è `Hello World`:

```
FAMOUS_WORDS "Hello World"
```

Un nome o un valore può contenere qualsiasi carattere diverso dal carattere null (che funge da delimitatore per *RFNVS*). Tuttavia, per facilitare l'interoperabilità, un'applicazione potrebbe preferire limitare i nomi ai seguenti caratteri:

- Primo carattere: alfabetico maiuscolo o minuscolo (da A a Z o da a a z) o carattere di sottolineatura.
- Caratteri successivi: alfabetico maiuscolo o minuscolo, cifra decimale (da 0 a 9), carattere di sottolineatura, trattino o punto.

Se un nome o un valore contiene uno o più apici, il nome o il valore devono essere racchiusi tra apici e ogni apice all'interno della stringa deve essere raddoppiato:

```
Famous_Words "The program displayed ""Hello World"""
```

I nomi e i valori sono sensibili al maiuscolo / minuscolo, ovvero, le lettere minuscole non sono considerate uguali alle lettere maiuscole. Ad esempio, `FAMOUS_WORDS` e `Famous_Words` sono due nomi differenti.

La lunghezza in byte di *RFNVS* è uguale a *RFLEN* meno RFLENV. Per evitare problemi con la conversione dei dati utente in alcuni ambienti, si consiglia che questa lunghezza sia un multiplo di quattro. *RFNVS* deve essere riempito con spazi vuoti fino a questa lunghezza o terminato in precedenza inserendo un carattere null dopo l'ultimo carattere significativo nella stringa. Il carattere null e i byte successivi, fino alla lunghezza specificata di *RFNVS*, vengono ignorati.

Nota: Poiché la lunghezza di questo campo non è fissa, il campo viene omesso dalle dichiarazioni della struttura fornite per i linguaggi di programmazione supportati.

RFSID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

RFSIDV
Identificativo per le regole e la struttura dell'intestazione di formattazione.

Il valore iniziale di questo campo è RFSIDV.

RFVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere:

RFVER1
Version-1 regole e formattazione della struttura dell'intestazione.

Il valore iniziale di questo campo è RFVER1.

Valori iniziali

Tabella 720. Campi in MQRFH

Nome campo	Nome della costante	Valore della costante
RFSID	RFSIDV	'RFH¬'
RFVER	RFVER1	1
RFLEN	RFLENV	32
RFENC	ENNAT	Dipende dall'ambiente
RFCSI	SUNDF	0
RFFMT	FMNONE	Spazi
RFFLG	RFNONE	0

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQRFH Structure
D*
D* Structure identifier
D  RFSID                  1      4    INZ('RFH ')
D* Structure version number
D  RFVER                  5      8I 0 INZ(1)
```

```
     D* Total length of MQRFH includingNameValueString
     D  RFLEN                  9     12I 0 INZ(32)
     D* Numeric encoding of data that followsNameValueString
     D  RFENC                 13     16I 0 INZ(273)
     D* Character set identifier of data thatfollows NameValueString
     D  RFCSI                 17     20I 0 INZ(0)
     D* Format name of data that followsNameValueString
     D  RFFMT                 21     28    INZ('        ')
     D* Flags
     D  RFFLG                 29     32I 0 INZ(0)
```

IBM i MQRFH2 (Regole e intestazione di formattazione 2) su IBM i

La struttura MQRFH2 definisce il formato delle regole version-2 e l'intestazione di formattazione.

Panoramica

Scopo: Questa intestazione può essere utilizzata per inviare i dati codificati utilizzando una sintassi di tipo XML. Un messaggio può contenere due o più strutture MQRFH2 in serie, con i dati utente che seguono facoltativamente l'ultima struttura MQRFH2 nelle serie.

Nome formato: FMRFH2.

Serie di caratteri e codifica: le regole speciali si applicano alla serie di caratteri e alla codificazione utilizzata per la struttura MQRFH2 :

- I campi diversi da *RF2NVD* sono nella serie di caratteri e nella codifica forniti dai campi *MDCSI* e *MDENC* nella struttura dell'intestazione che precede MQRFH2o da tali campi nella struttura MQMD se MQRFH2 si trova all'inizio dei dati del messaggio dell'applicazione.

  La serie di caratteri deve avere caratteri a byte singolo per i caratteri validi nei nomi di coda.

  Quando GMCONV viene specificato nella chiamata MQGET, il gestore code converte questi campi nella serie di caratteri e nella codifica richiesti.
- *RF2NVD* si trova nella serie di caratteri fornita dal campo *RF2NVC* . Solo alcune serie di caratteri Unicode sono valide per *RF2NVC* (consultare la descrizione di *RF2NVC* per dettagli).

  Alcune serie di caratteri hanno una rappresentazione che dipende dalla codifica. Se *RF2NVC* è una di queste serie di caratteri, *RF2NVD* deve essere nella stessa codifica degli altri campi in MQRFH2.

  Quando GMCONV viene specificata nella chiamata MQGET, il gestore code converte *RF2NVD* nella codifica richiesta, ma non ne modifica la serie di caratteri.

- “Campi” a pagina 1228
- “Valori iniziali” a pagina 1233
- “Dichiarazione RPG” a pagina 1234

Campi

La struttura MQRFH2 contiene i seguenti campi; i campi sono descritti in ordine alfabetico:

RF2CSI (numero intero con segno a 10 cifre)

Identificativo della serie di caratteri dei dati che seguono l'ultimo campo *RF2NVD* .

Specifica l'identificativo della serie di caratteri dei dati che seguono l'ultimo campo *RF2NVD* . Non si applica ai dati carattere nella stessa struttura MQRFH2

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. È possibile utilizzare il seguente valore speciale:

CINHT
Eredita l'identificativo della serie di caratteri di questa struttura.

I dati carattere nei dati *che seguono* questa struttura si trovano nella stessa serie di caratteri di questa struttura.

Il gestore code modifica questo valore nella struttura inviata nel messaggio nell'effettivo identificativo della serie di caratteri della struttura. Se non si verifica alcun errore, il valore CSINHT non viene restituito dalla chiamata MQGET.

CSINHT non può essere utilizzato se il valore del campo *MDPAT* in MQMD è ATBRKR.

Il valore iniziale di questo campo è CSINHT.

RF2ENC (numero intero con segno a 10 cifre)

Codifica numerica dei dati che seguono l'ultimo campo *RF2NVD* .

Specifica la codifica numerica dei dati che seguono l'ultimo campo *RF2NVD* ; non si applica ai dati numerici nella struttura MQRFH2 .

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati.

Il valore iniziale di questo campo è ENNAT.

RF2FLG (numero intero con segno a 10 cifre)

Indicatori.

È necessario specificare il seguente valore:

RFNONE
Nessun indicatore.

Il valore iniziale di questo campo è RFNONE.

RF2FMT (stringa di caratteri a 8 byte)

Nome formato dei dati che seguono l'ultimo campo *RF2NVD* .

Specifica il nome formato dei dati che seguono l'ultimo campo *RF2NVD* .

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono le stesse del campo *MDFMT* in MQMD.

Il valore iniziale di questo campo è FMNONE.

RF2LEN (numero intero con segno a 10 cifre)

Lunghezza totale di MQRFH2 inclusi tutti i campi *RF2NVL* e *RF2NVD* .

È la lunghezza in byte della struttura MQRFH2 , inclusi i campi *RF2NVL* e *RF2NVD* alla fine della struttura. È valido che vi siano più coppie di campi *RF2NVL* e *RF2NVD* alla fine della struttura, nella sequenza:

```
length1, data1, length2, data2, ...
```

RF2LEN non include i dati utente che possono seguire l'ultimo campo *RF2NVD* alla fine della struttura.

Per evitare problemi con la conversione dei dati utente in alcuni ambienti, considerare l'utilizzo di *RF2LEN* come multiplo di quattro.

La seguente costante fornisce la lunghezza della parte *fissa* della struttura, ovvero la lunghezza escludendo i campi *RF2NVL* e *RF2NVD* :

RFLEN2
Lunghezza della parte fissa della struttura MQRFH2 .

Il valore iniziale di questo campo è RFLEN2.

RF2NVC (numero intero con segno a 10 cifre)

Identificativo della serie di caratteri di *RF2NVD*.

Specifica il CCSID (coded character set identifier) dei dati nel campo *RF2NVD* . Questo è diverso dalla serie di caratteri delle altre stringhe nella struttura MQRFH2 e può essere diverso dalla serie di caratteri dei dati (se presenti) che seguono l'ultimo campo *RF2NVD* alla fine della struttura.

RF2NVC deve avere uno dei seguenti valori CCSID:

1200
UTF-16, versione Unicode più recente supportata

13488
UTF-16, sottoinsieme Unicode versione 2.0

17584
UTF-16, sottoinsieme Unicode versione 3.0 (include il simbolo Euro)

1208
UTF-8, versione Unicode più recente supportata

Per le serie di caratteri UTF-16 , la codifica (ordine byte) di *RF2NVD* deve corrispondere alla codifica degli altri campi nella struttura MQRFH2 . I caratteri surrogati (da X'D800'a X'DFFF') non sono supportati.

Nota: Se *RF2NVC* non dispone di uno dei valori elencati in precedenza e la struttura MQRFH2 richiede la conversione nella chiamata MQGET, la chiamata viene completata con il codice di errore RC2111 e il messaggio viene restituito non convertito.

Il valore iniziale di questo campo è 1208.

RF2NVD (stringa di caratteri a n byte)

Dati nome / valore.

Si tratta di una stringa di caratteri a lunghezza variabile che contiene dati codificati utilizzando una sintassi di tipo XML. La lunghezza in byte di questa stringa viene fornita dal campo *RF2NVL* che precede il campo *RF2NVD* ; questa lunghezza deve essere un multiplo di quattro.

I campi *RF2NVL* e *RF2NVD* sono facoltativi, ma se presenti devono essere una coppia e adiacenti. La coppia di campi può essere ripetuta tutte le volte necessarie, ad esempio:

```
length1 data1 length2 data2 length3 data3
```

Poiché questi campi sono facoltativi, vengono omessi dalle dichiarazioni della struttura fornite per i vari linguaggi di programmazione supportati.

RF2NVD è inusuale perché non viene convertito nella serie di caratteri specificata nella chiamata MQGET quando il messaggio viene richiamato con l'opzione GMCONV attiva; *RF2NVD* rimane nella relativa serie di caratteri originale. Tuttavia, *RF2NVD* viene convertito nella codifica specificata nella chiamata MQGET.

Sintassi dei dati nome / valore: la stringa è composta da una singola "cartella" che contiene zero o più proprietà. La cartella è delimitata da tag di inizio e fine XML con lo stesso nome della cartella:

```
<folder> property1 property2 ... </folder>
```

I caratteri che seguono la tag di fine cartella, fino alla lunghezza definita da *RF2NVL*, devono essere vuoti. All'interno della cartella, ogni proprietà è composta da un nome e da un valore e, facoltativamente, da un tipo di dati:

```
<name dt="datatype">value</name>
```

In questi esempi:

- I caratteri delimitatori (<, =, ", / e>) devono essere specificati esattamente come mostrato.

- `name` è il nome della proprietà specificato dall'utente; consultare il seguente esempio per ulteriori informazioni sui nomi.
- `datatype` è un tipo di dati facoltativo specificato dall'utente della proprietà; consultare il seguente esempio per tipi di dati validi.
- `value` è il valore della proprietà specificato dall'utente; consultare i seguenti paragrafi per ulteriori informazioni sui valori.
- Gli spazi vuoti sono significativi tra il carattere > che precede un valore e il carattere < che segue il valore e almeno uno spazio deve precedere `dt=`. Altrove gli spazi vuoti possono essere codificati liberamente tra tag, o prima o dopo tag (ad esempio, per migliorare la leggibilità); questi spazi vuoti non sono significativi.

Se le proprietà sono correlate tra loro, possono essere raggruppate racchiudendole all'interno di tag di inizio e di fine XML con lo stesso nome del gruppo:

```
<folder> <group> property1 property2 ... </group> </folder>
```

I gruppi possono essere nidificati all'interno di altri gruppi, senza limiti, e un gruppo può essere presente più di una volta all'interno di una cartella. È anche valido che una cartella contenga alcune proprietà nei gruppi e altre non nei gruppi.

Nomi delle proprietà, dei gruppi e delle cartelle: i nomi delle proprietà, dei gruppi e delle cartelle devono essere nomi di tag XML validi, ad eccezione del carattere due punti, che non è consentito in un nome proprietà, gruppo o cartella. In particolare:

- I nomi devono iniziare con una lettera o un carattere di sottolineatura. Le lettere valide vengono definite nella specifica XML W3C e consistono essenzialmente in categorie Unicode Ll, Lu, Lo, Lt e Nl.
- I caratteri rimanenti in un nome possono essere lettere, cifre decimali, caratteri di sottolineatura, trattini o punti. Questi corrispondono alle categorie Unicode Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm e Nd.
- I caratteri di compatibilità Unicode (X'F900' e versioni successive) non sono consentiti in nessuna parte di un nome.
- I nomi non devono iniziare con la stringa XML in una combinazione di lettere maiuscole o minuscole.

Inoltre:

- I nomi rispettano la distinzione maiuscole/minuscole. Ad esempio, ABC, abce Abc sono tre nomi diversi.
- Ogni cartella ha uno spazio dei nomi separato. Di conseguenza, un gruppo o una proprietà in una cartella non è in conflitto con un gruppo o una proprietà con lo stesso nome in un'altra cartella.
- I gruppi e le proprietà occupano lo stesso spazio dei nomi all'interno di una cartella. Di conseguenza, una proprietà non può avere lo stesso nome di un gruppo all'interno della cartella che contiene tale proprietà.

Generalmente, i programmi che analizzano il campo *RF2NVD* devono ignorare le proprietà o i gruppi che hanno nomi che il programma non riconosce, purché tali proprietà o gruppi siano formati correttamente.

Tipi di dati delle proprietà: ogni proprietà può avere un tipo di dati facoltativo. Se specificato, il tipo di dati deve essere uno dei seguenti valori, in maiuscolo, minuscolo o con caratteri misti:

Tabella 721. Tipi di dati e relativo utilizzo

Tipo dati	**Utilizzato per**
`string`	Qualsiasi sequenza di caratteri. È necessario specificare alcuni caratteri utilizzando le sequenze di escape.
`boolean`	Il carattere 0 o 1 (1 indica TRUE).

Tabella 721. Tipi di dati e relativo utilizzo (Continua)	
Tipo dati	**Utilizzato per**
`bin.hex`	Cifre esadecimali che rappresentano gli ottetti.
`i1`	Numero intero compreso tra -128 e +127, espresso utilizzando solo cifre decimali e segno facoltativo.
`i2`	Numero intero compreso tra -32 768 e +32 767, espresso utilizzando solo cifre decimali e segno facoltativo.
`i4`	Numero intero compreso tra -2 147 483 648 e + 2 147 483 647, espresso utilizzando solo cifre decimali e segno facoltativo.
`i8`	Numero intero compreso tra -9 223 372 036 854 775 808 e + 9 223 372 036 854 775 807, espresso utilizzando solo cifre decimali e segno facoltativo.
`int`	Numero intero compreso tra -9 223 372 036 854 775 808 e + 9 223 372 036 854 775 807, espresso utilizzando solo cifre decimali e segno facoltativo. Può essere utilizzato al posto di `i1`, `i2`, `i4`o `i8` se il mittente non desidera che implichi una particolare precisione.
`r4`	Numero a virgola mobile con grandezza compresa nell'intervallo tra 1.175E-37 e 3.402 823 47E+38, espresso utilizzando cifre decimali, segno facoltativo, cifre frazionarie facoltative ed esponente facoltativo.
`r8`	Numero a virgola mobile con magnitudine compresa nell'intervallo tra 2.225E-307 e 1.797 693 134 862 3E+308 espresso con cifre decimali, segno facoltativo, cifre frazionarie facoltative ed esponente facoltativo.

Valori delle proprietà: il valore di una proprietà può essere costituito da qualsiasi carattere, ad eccezione dei caratteri speciali che hanno una sequenza di escape associata obbligatoria. Ogni ricorrenza nel valore di un carattere contrassegnato come "obbligatorio" nella seguente tabella deve essere sostituita dalla sequenza di escape corrispondente. La tabella mostra anche i caratteri a cui è associata una sequenza di escape facoltativa. Ogni ricorrenza nel valore di un carattere contrassegnato come "facoltativo" può essere sostituita dalla sequenza di escape corrispondente, ma non è richiesta.

Tabella 722. Caratteri di escape e relativo utilizzo		
Carattere	**Sequenza di escape**	**Utilizzo**
`&`	`&`	Obbligatorio
`<`	`<`	Obbligatorio
`>`	`>`	Facoltativo
`"`	`&quot;`	Facoltativo
`'`	`&apos;`	Facoltativo

Nota: Il carattere & all'inizio di una sequenza di escape non deve essere sostituito da `&`.

Nel seguente esempio, gli spazi vuoti nel valore sono significativi; tuttavia, non sono necessarie sequenze di escape:

```
<Famous_Words>The program displayed "Hello World"</Famous_Words>
```

RF2NVL (numero intero con segno a 10 cifre)

Lunghezza di *RF2NVD*.

Specifica la lunghezza in byte dei dati nel campo *RF2NVD* . Per evitare problemi con la conversione dei dati (se presenti) che seguono il campo *RF2NVD* , *RF2NVL* deve essere un multiplo di quattro.

Nota: I campi *RF2NVL* e *RF2NVD* sono facoltativi, ma se presenti devono essere una coppia e adiacenti. La coppia di campi può essere ripetuta tutte le volte necessarie, ad esempio:

```
length1 data1 length2 data2 length3 data3
```

Poiché questi campi sono facoltativi, vengono omessi dalle dichiarazioni della struttura fornite per i vari linguaggi di programmazione supportati.

RF2SID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

RFSIDV
Identificativo per le regole e la struttura dell'intestazione di formattazione.

Il valore iniziale di questo campo è RFSIDV.

RF2VER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere:

RFVER2
Regole Version-2 e struttura dell'intestazione di formattazione.

Il valore iniziale di questo campo è RFVER2.

Valori iniziali

Tabella 723. Campi in MQRFH2

Nome campo	Nome della costante	Valore della costante
RF2SID	RFSIDV	'RFH¬'
RF2VER	RFVER2	2
RF2LEN	RFLEN2	36
RF2ENC	ENNAT	Dipende dall'ambiente
RF2CSI	CINHT	-2
RF2FMT	FMNONE	Spazi
RF2FLG	RFNONE	0
RF2NVC	Nessuna	1208

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQRFH2 Structure
D*
D* Structure identifier
D  RF2SID                 1      4    INZ('RFH ')
D* Structure version number
D  RF2VER                 5      8I 0 INZ(2)
D* Total length of MQRFH2 including allNameValueLength and
D* NameValueDatafields
D  RF2LEN                 9     12I 0 INZ(36)
D* Numeric encoding of data that followslast NameValueData field
D  RF2ENC                13     16I 0 INZ(273)
D* Character set identifier of data thatfollows last NameValueData field
D  RF2CSI                17     20I 0 INZ(-2)
D* Format name of data that follows lastNameValueData field
D  RF2FMT                21     28    INZ('        ')
D* Flags
D  RF2FLG                29     32I 0 INZ(0)
D* Character set identifier ofNameValueData
D  RF2NVC                33     36I 0 INZ(1208)
```

IBM i MQRMH (Reference message header) su IBM i

La struttura MQRMH definisce il formato di un'intestazione del messaggio di riferimento.

Panoramica

Scopo: questa intestazione viene utilizzata con le uscite del canale messaggi scritte dall'utente per inviare grandi quantità di dati (denominate "dati di massa") da un gestore code ad un altro. La differenza rispetto alla messaggistica normale è che i dati di massa non vengono memorizzati su una coda; invece, solo un *riferimento* ai dati di massa viene memorizzato sulla coda. Ciò riduce la possibilità che le risorse IBM MQ vengano esaurite da alcuni messaggi di grandi dimensioni.

Nome formato: FMRMH.

Serie di caratteri e codifica: i dati dei caratteri in MQRMH e le stringhe indirizzate dai campi di offset devono trovarsi nella serie di caratteri del gestore code locale; ciò viene fornito dall'attributo del gestore code **CodedCharSetId** . I dati numerici in MQRMH devono essere nella codifica della macchina nativa; ciò è dato dal valore di ENNAT per il linguaggio di programmazione C.

La serie di caratteri e la codifica di MQRMH devono essere impostate nei campi *MDCSI* e *MDENC* in:

- MQMD (se la struttura MQRMH è all'inizio dei dati del messaggio) oppure
- La struttura dell'intestazione che precede la struttura MQRMH (tutti gli altri casi).

Utilizzo: un'applicazione inserisce un messaggio composto da MQRMH, ma omettendo i dati di massa. Quando il messaggio viene letto dalla coda di trasmissione da un MCA (message channel agent), viene richiamata un'uscita messaggio fornita dall'utente per elaborare l'intestazione del messaggio di riferimento. L'uscita può aggiungere al messaggio di riferimento i dati di massa identificati dalla struttura MQRMH, prima che l'MCA invii il messaggio attraverso il canale al gestore code successivo.

All'estremità di ricezione, deve esistere un'exit di messaggi che attende i messaggi di riferimento. Quando viene ricevuto un messaggio di riferimento, l'uscita deve creare l'oggetto dai dati di massa che seguono MQRMH nel messaggio e quindi trasmettere il messaggio di riferimento senza i dati di massa. Il messaggio di riferimento può essere successivamente richiamato da un'applicazione che legge il messaggio di riferimento (senza i dati di massa) da una coda.

Normalmente, la struttura MQRMH è tutto ciò che si trova nel messaggio. Tuttavia, se il messaggio si trova su una coda di trasmissione, una o più intestazioni aggiuntive precederanno la struttura MQRMH.

Un messaggio di riferimento può anche essere inviato ad un elenco di distribuzione. In tal caso, la struttura MQDH e i relativi record precedono la struttura MQRMH quando il messaggio si trova su una coda di trasmissione.

Nota: Un messaggio di riferimento non deve essere inviato come messaggio segmentato, perché l'exit dei messaggi non può elaborarlo correttamente.

Conversione dati

Per la conversione dei dati, la conversione della struttura MQRMH include la conversione dei dati dell'ambiente di origine, del nome dell'oggetto di origine, dei dati dell'ambiente di destinazione e del nome dell'oggetto di destinazione. Tutti gli altri byte all'interno di *RMLEN* byte dell'inizio della struttura vengono scartati o hanno valori non definiti dopo la conversione dei dati. I dati di massa verranno convertiti a condizione che tutte le seguenti istruzioni siano vere:

- I dati di massa sono presenti nel messaggio quando viene eseguita la conversione dei dati.
- Il campo *RMFMT* in MQRMH ha un valore diverso da FMNONE.
- Esiste un'uscita di conversione dati scritta dall'utente con il nome formato specificato.

Tenere presente, tuttavia, che di solito i dati di massa non sono presenti nel messaggio quando il messaggio si trova su una coda e che, di conseguenza, i dati di massa non verranno convertiti dall'opzione GMCONV.

Campi

La struttura MQRMH contiene i campi riportati di seguito; i campi sono descritti in **ordine alfabetico**:

RMCSI (numero intero con segno a 10 cifre)

L'identificativo della serie di caratteri dei dati di massa.

Specifica l'identificativo della serie di caratteri dei dati di massa; non viene applicato ai dati di carattere nella stessa struttura MQRMH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. È possibile utilizzare il seguente valore speciale:

CINHT
Eredita l'identificativo della serie di caratteri di questa struttura.

I dati carattere nei dati *che seguono* questa struttura si trovano nella stessa serie di caratteri di questa struttura.

Il gestore code modifica questo valore nella struttura inviata nel messaggio nell'effettivo identificativo della serie di caratteri della struttura. Se non si verifica alcun errore, il valore CSINHT non viene restituito dalla chiamata MQGET.

CSINHT non può essere utilizzato se il valore del campo *MDPAT* in MQMD è ATBRKR.

Il valore iniziale di questo campo è CSUNDF.

RMDEL (numero intero con segno a 10 cifre)

Lunghezza dei dati di ambiente di destinazione.

Se questo campo è zero, non ci sono dati di ambiente di destinazione e *RMDEO* viene ignorato.

RMDEO (numero intero con segno a 10 cifre)

Offset dei dati di ambiente di destinazione.

Questo campo specifica l'offset dei dati di ambiente di destinazione dall'inizio della struttura MQRMH. I dati di ambiente di destinazione possono essere specificati dal creatore del messaggio di riferimento, se tali dati sono noti al creatore. Ad esempio, i dati dell'ambiente di destinazione potrebbero essere il percorso di directory dell'oggetto in cui devono essere archiviati i dati di massa. Tuttavia, se il creatore non conosce i dati dell'ambiente di destinazione, è responsabilità dell'uscita messaggi fornita dall'utente determinare le informazioni sull'ambiente necessarie.

La lunghezza dei dati dell'ambiente di destinazione è fornita da *RMDEL* ; se questa lunghezza è zero, non vi sono dati di ambiente di destinazione e *RMDEO* viene ignorato. Se presenti, i dati dell'ambiente di destinazione devono trovarsi completamente all'interno di *RMLEN* byte dall'inizio della struttura.

Le applicazioni non devono presumere che i dati dell'ambiente di destinazione siano contigui con i dati indirizzati dai campi *RMSEO*, *RMSNO*e *RMDNO* .

Il valore iniziale di questo campo è 0.

RMDL (numero intero con segno a 10 cifre)

Lunghezza dei dati di massa.

Il campo *RMDL* specifica la lunghezza dei dati di massa a cui fa riferimento la struttura MQRMH.

Se i dati di massa sono presenti nel messaggio, i dati iniziano con un offset di *RMLEN* byte dall'inizio della struttura MQRMH. La lunghezza dell'intero messaggio meno *RMLEN* fornisce la lunghezza dei dati di massa presenti.

Se i dati sono presenti nel messaggio, *RMDL* specifica la quantità di tali dati rilevante. Il caso normale è che *RMDL* abbia lo stesso valore della lunghezza dei dati presenti nel messaggio.

Se la struttura MQRMH rappresenta i restanti dati nell'oggetto (a partire dall'offset logico specificato), è possibile utilizzare il valore zero per *RMDL*, se i dati di massa non sono presenti nel messaggio.

Se non è presente alcun dato, la fine di MQRMH coincide con la fine del messaggio.

Il valore iniziale di questo campo è 0.

RMDNL (numero intero con segno a 10 cifre)

Lunghezza del nome oggetto di destinazione.

Se questo campo è zero, non c'è alcun nome oggetto di destinazione e *RMDNO* viene ignorato.

RMDNO (numero intero con segno a 10 cifre)

Offset del nome oggetto di destinazione.

Questo campo specifica lo scostamento del nome oggetto di destinazione dall'inizio della struttura MQRMH. Il nome dell'oggetto di destinazione può essere specificato dal creatore del messaggio di riferimento, se tali dati sono noti al creatore. Tuttavia, se il creatore non conosce il nome dell'oggetto di destinazione, è responsabilità dell'uscita messaggi fornita dall'utente identificare l'oggetto da creare o modificare.

La lunghezza del nome oggetto di destinazione è fornita da *RMDNL* ; se questa lunghezza è zero, non esiste alcun nome oggetto di destinazione e *RMDNO* viene ignorato. Se presente, il nome dell'oggetto di destinazione deve trovarsi completamente all'interno di *RMLEN* byte dall'inizio della struttura.

Le applicazioni non devono presumere che il nome dell'oggetto di destinazione sia contiguo con i dati indirizzati dai campi *RMSEO*, *RMSNO*e *RMDEO* .

Il valore iniziale di questo campo è 0.

RMDO (numero intero con segno a 10 cifre)

Offset basso dei dati di massa.

Questo campo specifica l'offset minimo dei dati di massa dall'inizio dell'oggetto di cui fanno parte i dati di massa. L'offset dei dati di massa dall'inizio dell'oggetto è denominato *offset logico*. Questo non è l'offset fisico dei dati di massa dall'inizio della struttura MQRMH - tale offset viene fornito da *RMLEN*.

Per consentire l'invio di oggetti di grandi dimensioni utilizzando messaggi di riferimento, lo scostamento logico è diviso in due campi e lo scostamento logico effettivo è dato dalla somma di questi due campi:

- *RMDO* rappresenta il resto ottenuto quando l'offset logico è diviso per 1 000 000 000. È quindi un valore compreso nell'intervallo tra 0 e 999 999 999.
- *RMDO2* rappresenta il risultato ottenuto quando l'offset logico è diviso per 1 000 000 000. È quindi il numero di multipli completi di 1 000 000 000 che esistono nell'offset logico. Il numero di multipli è compreso tra 0 e 999 999 999.

Il valore iniziale di questo campo è 0.

RMDO2 (numero intero con segno a 10 cifre)

Offset elevato dei dati di massa.

Questo campo specifica l'offset elevato dei dati di massa dall'inizio dell'oggetto di cui fanno parte i dati di massa. È un valore compreso tra 0 e 999 999 999. Consultare *RMDO* per i dettagli.

Il valore iniziale di questo campo è 0.

RMENC (numero intero con segno a 10 cifre)

Codifica numerica dei dati di massa.

Specifica la codifica numerica dei dati di massa; non si applica ai dati numerici nella stessa struttura MQRMH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati.

Il valore iniziale di questo campo è ENNAT.

RMFLG (numero intero con segno a 10 cifre)

Indicatori del messaggio di riferimento.

Sono definiti i seguenti indicatori:

RMLAST
Il messaggio di riferimento contiene o rappresenta l'ultima parte dell'oggetto.

Questo indicatore indica che il messaggio di riferimento rappresenta o contiene l'ultima parte dell'oggetto di riferimento.

RMNLST
Il messaggio di riferimento non contiene o rappresenta l'ultima parte dell'oggetto.

RMNLST è definito per aiutare la documentazione del programma. Non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

Il valore iniziale di questo campo è RMNLST.

RMFMT (stringa di caratteri a 8 byte)

Nome formato dei dati di massa.

Specifica il nome del formato dei dati di massa.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono le stesse del campo *MDFMT* in MQMD.

Il valore iniziale di questo campo è FMNONE.

RMLEN (numero intero con segno a 10 cifre)

Lunghezza totale di MQRMH, incluse le stringhe alla fine dei campi fissi, ma non i dati di massa.

Il valore iniziale di questo campo è zero.

RMOII (stringa bit a 24 byte)

Identificativo istanza oggetto.

Questo campo può essere utilizzato per identificare un'istanza specifica di un oggetto. Se non è necessario, deve essere impostato sul seguente valore:

OINATTIVO
Nessun identificativo istanza oggetto specificato.

Il valore è zero binario per la lunghezza del campo.

La lunghezza di questo campo è fornita da LNOIID. Il valore iniziale di questo campo è OIINON.

RMOT (stringa di caratteri a 8 byte)

Tipo di oggetto.

Questo è un nome che può essere utilizzato dall'uscita del messaggio per riconoscere i tipi di messaggio di riferimento che supporta. Considerare di rendere il nome conforme alle stesse regole del campo *RMFMT* .

Il valore iniziale di questo campo è di 8 spazi.

RMSEL (numero intero con segno a 10 cifre)

Lunghezza dei dati dell'ambiente di origine.

Se questo campo è zero, non ci sono dati di ambiente di origine e *RMSEO* viene ignorato.

Il valore iniziale di questo campo è 0.

RMSEO (numero intero con segno a 10 cifre)

Offset dei dati dell'ambiente di origine.

Questo campo specifica l'offset dei dati dell'ambiente origine dall'inizio della struttura MQRMH. I dati dell'ambiente di origine possono essere specificati dal creatore del messaggio di riferimento, se tali dati sono noti al creatore. Ad esempio, i dati dell'ambiente di origine potrebbero essere il percorso di directory dell'oggetto che contiene i dati di massa. Tuttavia, se il creatore non conosce i dati di ambiente di origine, è responsabilità dell'uscita messaggi fornita dall'utente determinare le informazioni di ambiente necessarie.

La lunghezza dei dati dell'ambiente di origine è fornita da *RMSEL* ; se questa lunghezza è zero, non vi sono dati dell'ambiente di origine e *RMSEO* viene ignorato. Se presenti, i dati dell'ambiente di origine devono trovarsi completamente all'interno di *RMLEN* byte dall'inizio della struttura.

Le applicazioni non devono presumere che i dati di ambiente inizino immediatamente dopo l'ultimo campo fisso nella struttura o che siano contigui con i dati indirizzati dai campi *RMSNO*, *RMDEO*e *RMDNO* .

Il valore iniziale di questo campo è 0.

RMSID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

RMSIDV
Identificativo per la struttura dell'intestazione del messaggio di riferimento.

Il valore iniziale di questo campo è RMSIDV.

RMSNL (numero intero con segno a 10 cifre)

Lunghezza del nome oggetto di origine.

Se questo campo è zero, non vi è alcun nome oggetto di origine e *RMSNO* viene ignorato.

Il valore iniziale di questo campo è 0.

RMSNO (numero intero con segno a 10 cifre)

Offset del nome oggetto di origine.

Questo campo specifica lo scostamento del nome oggetto di origine dall'inizio della struttura MQRMH. Il nome dell'oggetto di origine può essere specificato dal creatore del messaggio di riferimento, se tali dati sono noti al creatore. Tuttavia, se il creatore non conosce il nome dell'oggetto di origine, è responsabilità dell'uscita del messaggio fornito dall'utente identificare l'oggetto a cui accedere.

La lunghezza del nome oggetto di origine è fornita da *RMSNL* ; se questa lunghezza è zero, non esiste alcun nome oggetto di origine e *RMSNO* viene ignorato. Se presente, il nome dell'oggetto di origine deve trovarsi completamente all'interno di *RMLEN* byte dall'inizio della struttura.

Le applicazioni non devono presumere che il nome dell'oggetto di origine sia contiguo con i dati indirizzati dai campi *RMSEO*, *RMDEO*e *RMDNO* .

Il valore iniziale di questo campo è 0.

RMVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere:

RMVER1
Version-1 fa riferimento alla struttura dell'intestazione del messaggio.

La seguente costante specifica il numero di versione della versione corrente:

RMVERC
La versione corrente della struttura dell'intestazione del messaggio di riferimento.

Il valore iniziale di questo campo è RMVER1.

Valori iniziali

Tabella 724. Campi in MQRMH

Nome campo	Nome della costante	Valore della costante
RMSID	RMSIDV	'RMH¬'
RMVER	RMVER1	1
RMLEN	Nessuna	0
RMENC	ENNAT	Dipende dall'ambiente
RMCSI	SUNDF	0
RMFMT	FMNONE	Spazi
RMFLG	RMNLST	0
RMOT	Nessuna	Spazi
RMOII	OINATTIVO	Valori null
RMSEL	Nessuna	0
RMSEO	Nessuna	0
RMSNL	Nessuna	0
RMSNO	Nessuna	0
RMDEL	Nessuna	0
RMDEO	Nessuna	0

Tabella 724. Campi in MQRMH (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
RMDNL	Nessuna	0
RMDNO	Nessuna	0
RMDL	Nessuna	0
RMDO	Nessuna	0
RMDO2	Nessuna	0
Note: 1. Il simbolo ¬ rappresenta un singolo carattere vuoto.		

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQRMH Structure
D*
D* Structure identifier
D  RMSID                  1      4    INZ('RMH ')
D* Structure version number
D  RMVER                  5      8I 0 INZ(1)
D* Total length of MQRMH, includingstrings at end of fixed fields,but not
D* the bulk data
D  RMLEN                  9     12I 0 INZ(0)
D* Numeric encoding of bulk data
D  RMENC                 13     16I 0 INZ(273)
D* Character set identifier of bulkdata
D  RMCSI                 17     20I 0 INZ(0)
D* Format name of bulk data
D  RMFMT                 21     28    INZ('        ')
D* Reference message flags
D  RMFLG                 29     32I 0 INZ(0)
D* Object type
D  RMOT                  33     40    INZ
D* Object instance identifier
D  RMOII                 41     64    INZ(X'00000000000000-
D                                     00000000000000000000-
D                                     000000000000')
D* Length of source environmentdata
D  RMSEL                 65     68I 0 INZ(0)
D* Offset of source environmentdata
D  RMSEO                 69     72I 0 INZ(0)
D* Length of source object name
D  RMSNL                 73     76I 0 INZ(0)
D* Offset of source object name
D  RMSNO                 77     80I 0 INZ(0)
D* Length of destination environmentdata
D  RMDEL                 81     84I 0 INZ(0)
D* Offset of destination environmentdata
D  RMDEO                 85     88I 0 INZ(0)
D* Length of destination objectname
D  RMDNL                 89     92I 0 INZ(0)
D* Offset of destination objectname
D  RMDNO                 93     96I 0 INZ(0)
D* Length of bulk data
D  RMDL                  97    100I 0 INZ(0)
D* Low offset of bulk data
D  RMDO                 101    104I 0 INZ(0)
D* High offset of bulk data
D  RMDO2                105    108I 0 INZ(0)
```

Dichiarazione RPG

IBM i MQRR (Response record) su IBM i

La struttura MQRR viene utilizzata per ricevere il codice di completamento e il codice motivo risultanti dall'operazione di apertura o inserimento per una singola coda di destinazione, quando la destinazione è un elenco di distribuzione.

Panoramica

Scopo: MQRR è una struttura di output per chiamate MQOPEN, MQPUT e MQPUT1 .

Serie di caratteri e codifica: i dati in MQRR devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e la codifica del gestore code locale fornita da ENNAT. Tuttavia, se l'applicazione è in esecuzione come un client IBM MQ , la struttura deve essere nella serie di caratteri e nella codifica del client.

Utilizzo: fornendo un array di queste strutture sulle chiamate MQOPEN e MQPUT o sulla chiamata MQPUT1 , è possibile determinare i codici di completamento e i codici motivo per tutte le code in un elenco di distribuzione quando il risultato della chiamata è misto, ovvero quando la chiamata ha esito positivo per alcune code nell'elenco ma ha esito negativo per altre. Il codice di errore RC2136 dalla chiamata indica che i record di risposta (se forniti dall'applicazione) sono stati impostati dal gestore code.

- “Campi” a pagina 1241
- “Valori iniziali” a pagina 1241
- “Dichiarazione RPG” a pagina 1241

Campi

La struttura MQRR contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

RRCC (numero intero con segno a 10 cifre)

Codice di completamento per la coda.

Questo è il codice di completamento risultante dall'operazione di apertura o di inserimento per la coda con il nome specificato dall'elemento corrispondente nell'array di strutture MQOR fornito nella chiamata MQOPEN o MQPUT1 .

Questo è sempre un campo di output. Il valore iniziale di questo campo è CCOK.

RRREA (numero intero con segno a 10 cifre)

Codice di errore per la coda.

Questo è il codice di errore risultante dall'operazione di apertura o di inserimento per la coda con il nome specificato dall'elemento corrispondente nell'array di strutture MQOR fornito nella chiamata MQOPEN o MQPUT1 .

Questo è sempre un campo di output. Il valore iniziale di questo campo è RCNONE.

Valori iniziali

Tabella 725. Campi in MQRR

Nome campo	Nome della costante	Valore della costante
RRCC	CCOK	0
RRREA	RCNONE	0

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
```

```
     D* MQRR Structure
     D*
     D* Completion code for queue
     D  RRCC                   1      4I 0 INZ(0)
     D* Reason code for queue
     D  RRREA                  5      8I 0 INZ(0)
```

IBM i MQSCO (opzioni di configurazione TLS) su IBM i

La struttura MQSCO (con i campi TLS nella struttura MQCD) consente a un'applicazione in esecuzione come IBM MQ MQI client di specificare opzioni di configurazione che controllano l'utilizzo di TLS per la connessione client quando il protocollo del canale è TCP/IP.

Panoramica

Scopo: La struttura è un parametro di immissione sulla chiamata MQCONNX.

Se il protocollo del canale per il canale client non è TCP/IP, la struttura MQSCO viene ignorata.

Serie di caratteri e codifica: i dati in MQSCO devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e nella codifica del gestore code locale fornito da ENNAT.

- "Campi" a pagina 1242
- "Valori iniziali" a pagina 1246
- "Dichiarazione RPG" a pagina 1247

Campi

La struttura MQSCO contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

SCAIC (numero intero con segno a 10 cifre)

Questo è il numero di record di informazioni di autenticazione (MQAIR) indirizzati dai campi *SCAIP* o *SCAIO* . Per ulteriori informazioni, consultare"MQAIR (Record delle informazioni di autenticazione) su IBM i" a pagina 1041. Il valore deve essere maggiore o uguale a zero. Se il valore non è valido, la chiamata ha esito negativo con codice di errore RC2383.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

SCAIO (numero intero con segno a 10 cifre)

Questo è lo scostamento, in byte, del primo record delle informazioni di autenticazione dall'inizio della struttura MQSCO. L'offset può essere positivo o negativo. Il campo viene ignorato se *SCAIC* è zero.

È possibile utilizzare *SCAIO* o *SCAIP* per specificare i record MQAIR, ma non entrambi; consultare la descrizione del campo *SCAIP* per i dettagli.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

SCAIP (numero intero con segno a 10 cifre)

Questo è l'indirizzo del primo record di informazioni di autenticazione. Il campo viene ignorato se *SCAIC* è zero.

È possibile fornire l'array di record MQAIR in uno dei seguenti due modi:

- Utilizzando il campo puntatore *SCAIP*

  In questo caso, l'applicazione può dichiarare un array di record MQAIR separato dalla struttura MQSCO e impostare *SCAIP* sull'indirizzo dell'array.

  Considerare l'utilizzo di *SCAIP* per i linguaggi di programmazione che supportano il tipo di dati del puntatore in un modo che sia portabile in ambienti differenti (ad esempio, il linguaggio di programmazione C).
- Utilizzando il campo offset *SCAIO*

In questo caso, l'applicazione deve dichiarare una struttura composta contenente un MQSCO seguito dall'array di record MQAIR e impostare *SCAIO* sull'offset del primo record nell'array dall'inizio della struttura MQSCO. Verificare che questo valore sia corretto e che abbia un valore che possa essere utilizzato all'interno di un MQLONG (il linguaggio di programmazione più restrittivo è COBOL, per cui l'intervallo valido è compreso tra -999 999 999 e +999 999 999).

Considerare l'utilizzo di *SCAIO* per linguaggi di programmazione che non supportano il tipo di dati puntatore o che implementano il tipo di dati puntatore in un modo non portabile in ambienti differenti (ad esempio, il linguaggio di programmazione COBOL).

Indipendentemente dalla tecnica scelta, è possibile utilizzare solo uno tra *SCAIP* e *SCAIO* ; la chiamata ha esito negativo con codice di errore RC2384 se entrambi sono diversi da zero.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null in quei linguaggi di programmazione che supportano i puntatori e, in caso contrario, una stringa di byte completamente null.

Nota: Su piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata.

SCCERLBL (numero intero con segno a 10 cifre)

Questo campo fornisce i dettagli dell'etichetta del certificato utilizzata.

IBM MQ inizializza il valore per il campo SCCERLBL come spazi. Immettere il valore richiesto o accettare il valore predefinito.

`ibmwebspheremq`*`user_id`* è un valore valido per questo campo per tutte le versioni del prodotto e per le versioni MQSCO inferiori a 5.0 è l'unico valore valido. Pertanto, il valore di questo campo viene interpretato in fase di runtime e, se necessario, modificato. Se si specifica una versione MQSCO precedente a 5.0o si accetta il valore predefinito di spazi vuoti per il campo SCCERLBL, il sistema utilizza il valore `ibmwebspheremq`*`user_id`*.

Questo è un campo di immissione.

SCCERTVPOL (numero intero con segno a 10 cifre)

Questo campo specifica quale tipo di politica di convalida del certificato viene utilizzato. Il campo può essere impostato su uno dei seguenti valori:

MQ_CERT_VAL_POLICY_ANY
Applicare ciascuna delle politiche di convalida del certificato supportate dalla libreria dei socket sicuri. Accettare la catena di certificati se una delle politiche considera valida la catena di certificati.

MQ_CERT_VAL_POLICY_RFC5280
Applicare solo la politica di convalida del certificato conforme a RFC5280 . Questa impostazione fornisce una convalida più rigorosa rispetto all'impostazione ANY, ma rifiuta alcuni certificati digitali meno recenti.

Il valore iniziale di questo campo è MQ_CERT_VAL_POLICY_ANY

SCCH (numero intero con segno a 10 cifre)

Questo campo fornisce i dettagli di configurazione per l'hardware crittografico collegato al sistema client.

Impostare il campo su una stringa nel seguente formato oppure lasciarlo vuoto o null:

```
GSK_PKCS11=the PKCS #11 driver path and file name;the PKCS #11
token label;the PKCS #11 token password;symmetric cipher setting>;
```

Per utilizzare l'hardware crittografico che è conforme all'interfaccia PKCS11 , ad esempio, IBM 4960 o IBM 4963, specificare il percorso del driver PKCS11 , l'etichetta del token PKCS11 e le stringhe della password del token PKCS11 , ciascuna terminata con un punto e virgola.

Il percorso del driver PKCS #11 è un percorso assoluto della libreria condivisa che fornisce supporto per la scheda PKCS #11 . Il nome file del driver PKCS #11 è il nome della libreria condivisa. Un esempio del valore richiesto per il percorso e il nome file PKCS #11 è:

```
/usr/lib/pkcs11/PKCS11_API.so
```

L'etichetta del token PKCS #11 deve essere interamente in minuscolo. Se l'hardware è stato configurato con un'etichetta con caratteri maiuscoli o maiuscoli, riconfigurarlo con questa etichetta minuscola.

Se non è richiesta alcuna configurazione hardware crittografica, impostare il campo su uno spazio vuoto o null.

Se il valore è più breve della lunghezza del campo, terminare il valore con un carattere null o riempirlo con spazi vuoti fino alla lunghezza del campo. Se il valore non è valido o causa un errore quando viene utilizzato per configurare l'hardware di crittografia, la chiamata ha esito negativo con codice di errore RC2382.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da LNSSCH. Il valore iniziale di questo campo è costituito da spazi.

SCETSUITEB (numero intero con segno a 10 cifre)
Questo campo specifica se viene utilizzata la crittografia conforme a Suite B e quale livello di intensità viene utilizzato. Il valore può essere uno o più dei seguenti:

- SCEPSUITEB0

  La crittografia conforme alla suite B non viene utilizzata.
- SCEPSUITEB1

  Viene utilizzata la sicurezza della suite B a 128 bit.
- SCEPSUITEB2

  Viene utilizzata la sicurezza della suite B a 192 bit.

Nota: L'utilizzo di SCEPSUITEB0 con qualsiasi altro valore in questo campo non è valido.

SCFR (numero intero con segno a 10 cifre)

IBM MQ può essere configurato con hardware crittografico in modo che i moduli di codifica utilizzati siano quelli forniti dal prodotto hardware; questi possono essere certificati FIPS ad un determinato livello a seconda del prodotto hardware crittografico in uso.

Utilizzare questo campo per specificare che vengono utilizzati solo algoritmi certificati FIPS se la crittografia viene fornita nel software fornito da IBM MQ.

Quando IBM MQ è installato, viene installata anche un'implementazione della crittografia TLS che fornisce alcuni moduli certificati FIPS.

I valori possono essere:

MQSSL_FIPS_NO
Questo è il valore predefinito. Quando impostato su questo valore:

- È possibile utilizzare qualsiasi CipherSpec supportato su una particolare piattaforma.
- Se si esegue senza utilizzare l'hardware crittografico, i seguenti CipherSpecs vengono eseguiti utilizzando la crittografia certificata FIPS 140-2 sulle piattaforme IBM MQ :
  - TLS_RSA_WITH_3DES_EDE_CBC_SHA
  - TLS_RSA_WITH_AES_128_CBC_SHA
  - TLS_RSA_WITH_AES_256_CBC_SHA

SÌ MQSSL_FIPS
Quando è impostato su questo valore, a meno che non si stia utilizzando l'hardware crittografico per eseguire la crittografia, è possibile essere certi che

- Solo gli algoritmi di crittografia certificati FIPS possono essere utilizzati in CipherSpec che si applica a questa connessione client.
- Le connessioni del canale TLS in entrata e in uscita hanno esito positivo solo se viene utilizzata una delle seguenti specifiche di cifratura:
  - TLS_RSA_WITH_3DES_EDE_CBC_SHA
  - TLS_RSA_WITH_AES_128_CBC_SHA
  - TLS_RSA_WITH_AES_256_CBC_SHA

Note:

1. Deprecated CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA è obsoleto.
2. Laddove possibile, se è configurato CipherSpecs solo FIPS, il client MQI rifiuta le connessioni che specificano una CipherSpec non FIPS withRC2393. IBM MQ non garantisce di rifiutare tutte queste connessioni ed è responsabilità dell'utente determinare se la configurazione di IBM MQ è conforme a FIPS.

SCKEYPWL (numero intero con segno a 10 cifre)

Questa è la lunghezza della passphrase del repository chiavi TLS.

La lunghezza massima della passphrase del repository delle chiavi è 128 caratteri. Se la passphrase del repository delle chiavi è maggiore della lunghezza massima consentita, la connessione non riesce con RC2381.

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

SCKEYPWO (numero intero con segno a 10 cifre)

Questo è l'offset in byte della passphrase del repository chiavi TLS. L'offset può essere positivo o negativo.

È possibile utilizzare SCKEYPWO o SCKEYPWP per specificare la passphrase del repository delle chiavi, ma non entrambe. Per ulteriori informazioni, consultare la descrizione del campo SCKEYPWP .

Questo è un campo di immissione. Il valore iniziale di questo campo è 0.

SCKEYPWP (puntatore)

Questo è l'indirizzo della passphrase del repository chiavi TLS.

Questo è un campo di immissione. Il valore iniziale di questo campo è il puntatore null.

La passphrase del repository delle chiavi può essere specificata come una stringa di testo semplice o una passphrase che è stata codificata utilizzando il programma di utilità **runmqicred** .

La passphrase del repository di chiavi specificata utilizzando questo campo sovrascrive qualsiasi passphrase del repository di chiavi specificata utilizzando la variabile di ambiente *MQKEYRPWD* o la proprietà *SSLKeyRepositoryPassword* nella sezione SSL del file di configurazione del client.

È possibile utilizzare SCKEYPWO o SCKEYPWP per specificare la passphrase del repository delle chiavi, ma non entrambe.

SCKR (numero intero con segno a 10 cifre)

Questo campo specifica l'ubicazione del file database delle chiavi in cui sono memorizzate le chiavi e i certificati. Se il suffisso del file non è specificato, viene aggiunto automaticamente un suffisso `.kdb` .

Ogni file di database delle chiavi può avere un *file stash delle password*associato. Contiene le password codificate utilizzate per consentire l'accesso programmatico al database di chiavi. Il file stash delle password deve risiedere nella stessa directory e avere lo stesso file system del database delle chiavi e deve terminare con il suffisso `.sth`.

Ad esempio, se il file del database delle chiavi è `/xxx/yyy/key.kdb`, il file stash delle parole d'ordine deve essere `/xxx/yyy/key.sth`, dove `xxx` e `yyy` rappresentano i nomi delle directory.

La password del database di chiavi può essere specificata anche utilizzando i campi *SCKEYPWP* o *SCKEYPWO* .

Se il valore è più breve della lunghezza del campo, terminare il valore con un carattere null o riempirlo con spazi vuoti fino alla lunghezza del campo. Il valore non è selezionato; se si verifica un errore durante l'accesso al repository delle chiavi, la chiamata ha esito negativo con codice di errore RC2381.

Per eseguire una connessione TLS da un IBM MQ MQI client, impostare *SCKR* su un nome file database di chiavi valido.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da LNSSKR. Il valore iniziale di questo campo è un carattere vuoto.

SCSID (numero intero con segno a 10 cifre)

Questo è l'identificatore della struttura; il valore deve essere:

SSID
Identificativo per la struttura di opzioni di configurazione TLS.

Questo è sempre un campo di input. Il valore iniziale di questo campo è SCSIDV.

SCVER (numero intero con segno a 10 cifre)

Questo è il numero di versione della struttura; il valore deve essere:

SCVER1
Version-1 Struttura delle opzioni di configurazione TLS.

SCVER2
Version-2 Struttura delle opzioni di configurazione TLS.

SCVER3
Version-3 Struttura delle opzioni di configurazione TLS.

SCVER4
Version-4 Struttura delle opzioni di configurazione TLS.

SCVER5
Version-5 Struttura delle opzioni di configurazione TLS.

SCVER6
Version-6 Struttura delle opzioni di configurazione TLS.

La seguente costante specifica il numero di versione della versione corrente:

SCVERC
La versione corrente della struttura delle opzioni di configurazione TLS.

Questo è sempre un campo di input. Il valore iniziale di questo campo Š SCVER1.

Valori iniziali

Tabella 726. Campi in MQSCO

Nome campo	Nome della costante	Valore della costante
SCSID	SSID	'SCO¬'
SCVER	SCVER1	1
SCKR	Nessuna	Stringa null o spazi vuoti
SCCH	Nessuna	Stringa null o spazi vuoti
SCAIC	Nessuna	0
SCAIO	Nessuna	0

Tabella 726. Campi in MQSCO (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
SCAIP	Nessuna	Puntatore null o byte null
SCKRC	Nessuna	Puntatore null o byte null
SCFR	Nessuna	Puntatore null o byte null
SCEPSUITEB	Nessuna	Puntatore null o byte null
SCCERTVPOL	Nessuna	Puntatore null o byte null
SCCERLBL	Nessuna	Puntatore null o byte null
SCKEYPWP	Nessuna	Puntatore null o byte null
SCKEYPWO	Nessuna	0
SCKEYPWL	Nessuna	0

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Consultare “Dichiarazione RPG” a pagina 1247 per le opzioni SCEPSUITEB .

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQSCO Structure
D*
D* Structure identifier
D  SCSID                  1      4    INZ('SCO ')
D* Structure version number
D  SCVER                  5      8I 0 INZ(1)
D* Location of TLS key repository
D  SCKR                   9    264    INZ
D* Cryptographic hardware configuration string
D  SCCH                 265    520    INZ
D* Number of MQAIR records present
D  SCAIC                521    524I 0 INZ(0)
D* Offset of first MQAIR record from start of MQSCO structure
D  SCAIO                525    528I 0 INZ(0)
D* Address of first MQAIR record
D SCAIP                 529    544*   INZ(*NULL)
D* Ver:1 **
D* Number of unencrypted bytes sent/received before secret key is
D* reset
D SCKRC                 545    548I 0 INZ(0)
D* Using FIPS-certified algorithms
D SCFR                  549    552I 0 INZ(0)
D* Ver:2 **
* Use only Suite B cryptographic algorithms
D SCEPSUITEB0
D SCEPSUITEB1           553    556I 0 INZ(1)
D SCEPSUITEB2           557    560I 0 INZ(0)
D SCEPSUITEB3           561    564I 0 INZ(0)
D SCEPSUITEB4           565    568I 0 INZ(0)
D SCEPSUITEB                    10I 0 DIM(4) OVERLAY(SCEPSUITEB0)
D* Ver:3 **
D* Certificate validation policy
```

```
     D SCCERTVPOL            569    572I 0 INZ(0)
     D* Ver:4 **
```

IBM i MQSD (Subscription descriptor) su IBM i

La struttura MQSD viene utilizzata per specificare i dettagli relativi alla sottoscrizione che si sta effettuando.

Panoramica

Finalità
La struttura è un parametro di input / output nella chiamata MQSUB.

Sottoscrizioni gestite
Se un'applicazione non ha la necessità specifica di utilizzare una particolare coda come destinazione per le pubblicazioni che corrispondono alla relativa sottoscrizione, può utilizzare la funzione di sottoscrizione gestita. Se un'applicazione sceglie di utilizzare una sottoscrizione gestita, il gestore code informa il sottoscrittore della destinazione in cui vengono inviati i messaggi pubblicati, fornendo un handle dell'oggetto come output dalla chiamata MQSUB. Per ulteriori informazioni, consultare HOBJ (10 - digit signed integer) - input/output.

Quando la sottoscrizione viene rimossa, il gestore code si impegna anche a ripulire i messaggi che non sono stati richiamati dalla destinazione gestita, nelle seguenti situazioni:

- Quando la sottoscrizione viene rimossa - utilizzando MQCLOSE con CORMSB - e l'Hobj gestito viene chiuso.
- In modo implicito, quando la connessione viene persa per un'applicazione che utilizza una sottoscrizione non durevole (SONDUR)
- Per scadenza quando una sottoscrizione viene rimossa perché è scaduta e l'Hobj gestito è chiuso.

È necessario utilizzare le sottoscrizioni gestite con sottoscrizioni non durevoli, in modo che la ripulitura possa verificarsi e in modo che i messaggi per le sottoscrizioni non durevoli chiuse non occupano spazio nel gestore code. Le sottoscrizioni durevoli possono anche utilizzare destinazioni gestite.

Serie di caratteri e codifica
I dati in MQSD devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da ENNAT. Tuttavia, se l'applicazione è in esecuzione come un client IBM MQ , la struttura deve essere nella serie di caratteri e nella codifica del client.

- “Campi” a pagina 1248
- “Valori iniziali” a pagina 1261
- “Dichiarazione RPG” a pagina 1262

Campi

La struttura MQSD contiene i campi riportati di seguito; i campi sono descritti in ordine alfabetico:

SDAID (stringa di caratteri a 32 byte)

Questo valore si trova nel campo *MDAID* di MQMD (Message Descriptor) di tutti i messaggi di pubblicazione corrispondenti a questa sottoscrizione. *SDAID* fa parte del contesto di identità del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio.

Per ulteriori informazioni su *MDAID* , consultare MDAID.

Se l'opzione SOSETI non viene specificata, il *MDAID* impostato in ogni messaggio pubblicato per questa sottoscrizione è vuoto, come informazioni di contesto predefinite.

Se viene specificata l'opzione SOSETI , *SDAID* viene generato dall'utente e questo campo è un campo di input che contiene il *MDAID* da impostare in ogni pubblicazione per questa sottoscrizione.

La lunghezza di questo campo è fornita da LNAIDD. Il valore iniziale di questo campo è di 32 caratteri vuoti.

Se si modifica una sottoscrizione esistente utilizzando l'opzione SOALT, è possibile modificare il *SDAID* di eventuali messaggi di pubblicazione futuri.

Al ritorno da una chiamata MQSUB che utilizza SORES, questo campo è impostato sul *MDAID* corrente utilizzato per la sottoscrizione.

SDACC (stringa di caratteri a 32 byte)

Questo valore si trova nel campo *MDACC* di MQMD (Message Descriptor) di tutti i messaggi di pubblicazione corrispondenti a questa sottoscrizione. *MDACC* fa parte del contesto di identità del messaggio. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio.

Per ulteriori informazioni su *MDACC* , consultare MDACC.

È possibile utilizzare il seguente valore speciale per il campo *SDACC* :

ACNONE
Nessun token di account specificato.

Il valore è zero binario per la lunghezza del campo.

Se l'opzione SOSETI non viene specificata, il token di account viene generato dal gestore code come informazioni di contesto predefinite e questo campo è un campo di output che contiene il *MDACC* impostato in ogni messaggio pubblicato per questa sottoscrizione.

Se viene specificata l'opzione SOSETI , il token di account viene generato dall'utente e questo campo è un campo di input che contiene il *MDACC* da impostare in ogni pubblicazione per questa sottoscrizione.

La lunghezza di questo campo è fornita da LNACCT. Il valore iniziale di questo campo è ACNONE.

Se si modifica una sottoscrizione esistente utilizzando l'opzione SOALT , è possibile modificare il valore di *MDACC* in eventuali messaggi di pubblicazione futuri.

Al ritorno da una chiamata MQSUB utilizzando SORES, questo campo è impostato sul *MDACC* corrente utilizzato per la sottoscrizione.

SDASI (stringa di bit a 40 byte)

Si tratta di un identificativo di sicurezza che viene passato con *SDAU* al servizio di autorizzazione per consentire l'esecuzione dei controlli di autorizzazione appropriati.

SDASI viene utilizzato solo se viene specificato SOALTU e il campo *SDAU* non è completamente vuoto fino al primo carattere null o alla fine del campo.

Al ritorno da una chiamata MQSUB utilizzando SORES, questo campo non viene modificato.

Per ulteriori informazioni, consultare la descrizione di ODASI nel tipo di dati MQOD.

SDAU (stringa di caratteri a 12 byte)

Se si specifica SOALTU, questo campo contiene un identificativo utente alternativo utilizzato per controllare l'autorizzazione per la sottoscrizione e per l'output nella coda di destinazione (specificato nel parametro **Hobj** della chiamata MQSUB), al posto dell'identificativo utente con cui l'applicazione è attualmente in esecuzione.

Se l'operazione ha esito positivo, l'identificativo utente specificato in questo campo viene registrato come l'identificativo utente proprietario della sottoscrizione al posto dell'identificativo utente con cui è attualmente in esecuzione l'applicazione.

Se si specifica SOALTU e questo campo è completamente vuoto fino al primo carattere null o alla fine del campo, la sottoscrizione può riuscire solo se non è richiesta alcuna autorizzazione utente per sottoscrivere questo argomento con le opzioni specificate o la coda di destinazione per l'emissione.

Se SOALTU non viene specificato, questo campo viene ignorato.

Al ritorno da una chiamata MQSUB utilizzando SORES, questo campo non viene modificato.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da LNUID. Il valore iniziale di questo campo è di 12 caratteri vuoti.

SDCID (stringa di bit a 24 byte)

Tutte le pubblicazioni inviate per corrispondere a questa sottoscrizione contengono questo identificativo di correlazione nel descrizione del messaggio. Se più sottoscrizioni utilizzano la stessa coda da cui ottenere le proprie pubblicazioni, l'uso di MQGET per ID correlazione consente di ottenere solo le pubblicazioni per una sottoscrizione specifica. Questo identificativo di correlazione può essere generato dal gestore code o dall'utente.

Se l'opzione SOSCID non viene specificata, l'identificativo di correlazione viene generato dal gestore code e questo campo è un campo di output che contiene l'identificativo di correlazione impostato in ogni messaggio pubblicato per questa sottoscrizione.

Se viene specificata l'opzione SOSCID , l'identificativo di correlazione viene generato dall'utente e questo campo è un campo di immissione che contiene l'identificativo di correlazione da impostare in ogni pubblicazione per questa sottoscrizione. In questo caso, se il campo contiene CINONE, l'identificativo di correlazione impostato in ogni messaggio pubblicato per questa sottoscrizione è l'identificativo di correlazione creato dall'inserimento originale del messaggio.

Se viene specificata l'opzione SOGRP e l'identificatore di correlazione specificato è lo stesso di una sottoscrizione raggruppata esistente che utilizza la stessa coda e una stringa di argomento sovrapposta, solo la sottoscrizione più significativa nel gruppo viene fornita con una copia della pubblicazione.

La lunghezza di questo campo è fornita da LNCID. Il valore iniziale di questo campo è CINONE.

Se si modifica una sottoscrizione esistente utilizzando l'opzione SOALT e questo campo è un campo di input, l'ID di correlazione della sottoscrizione può essere modificato, a meno che la sottoscrizione non sia stata creata utilizzando l'opzione SOGRP .

Al ritorno da una chiamata MQSUB che utilizza SORES, questo campo è impostato sull'ID di correlazione corrente per la sottoscrizione.

SDEXP (numero intero con segno a 10 cifre)

Questo è il tempo espresso in decimi di secondo dopo il quale scade la sottoscrizione. Nessun' altra pubblicazione corrisponderà a questa sottoscrizione una volta trascorso questo intervallo. Viene utilizzato anche come valore nel campo *MDEXP* nell'MQMD delle pubblicazioni inviate a questo sottoscrittore.

Viene riconosciuto il seguente valore speciale:

EIULIM
La sottoscrizione ha un tempo di scadenza illimitato.

Se si modifica una sottoscrizione esistente utilizzando l'opzione SOALT , la scadenza della sottoscrizione può essere modificata.

Al ritorno da una chiamata MQSUB utilizzando l'opzione SORES , questo campo è impostato sulla scadenza originale dell'abbonamento e non sulla scadenza rimanente.

SDON (stringa di caratteri a 48 byte)

È il nome dell'oggetto argomento come definito sul gestore code locale.

Il nome può contenere i seguenti caratteri:

- Caratteri alfabetici maiuscoli (da A a Z)
- Caratteri alfabetici minuscoli (da a a z)
- Cifre numeriche (da 0 a 9)
- Punto (.), barra (/), sottolineatura (_), percentuale (%)

Il nome non deve contenere spazi iniziali o intermedi, ma può contenere spazi finali. Utilizzare un carattere null per indicare la fine dei dati significativi nel nome; il valore null e i caratteri che lo seguono vengono trattati come spazi vuoti. Sono applicate le seguenti limitazioni:

- Sui sistemi che utilizzano EBCDIC Katakana, non è possibile utilizzare caratteri minuscoli.
- I nomi contenenti caratteri minuscoli, barra o percentuale devono essere racchiusi tra virgolette quando vengono specificati nei comandi. Questi apici non devono essere specificati per i nomi che si verificano come campi nelle strutture o come parametri nelle chiamate.

Il *SDON* viene utilizzato per formare il nome completo dell'argomento.

Il nome completo dell'argomento può essere creato da due campi differenti: *SDON* e *SDOS*. Per i dettagli sul modo in cui questi due campi vengono utilizzati, consultare Combinazione di stringhe argomento.

Al ritorno da una chiamata MQSUB utilizzando l'opzione S0RES , questo campo non viene modificato.

La lunghezza di questo campo è fornita da LNTOPN. Il valore iniziale di questo campo è di 48 caratteri vuoti.

Se si modifica una sottoscrizione esistente utilizzando l'opzione SDALT, il nome dell'oggetto argomento sottoscritto non può essere modificato. Questo campo e *SDOS* possono essere omessi. Se vengono forniti, devono risolversi nello stesso nome dell'argomento completo oppure la chiamata ha esito negativo con RC2510 .

SDOPT (numero intero con segno a 10 cifre)

È possibile specificare almeno una delle seguenti opzioni:

- S0ALT
- S0RES
- S0CRT

I valori possono essere aggiunti. Non aggiungere la stessa costante più di una volta. La tabella mostra come è possibile combinare queste opzioni: vengono annotate le combinazioni non valide; tutte le altre combinazioni sono valide.

Opzioni di accesso o di creazione

Le opzioni di accesso e creazione controllano se una sottoscrizione viene creata o se una sottoscrizione esistente viene restituita o modificata. È necessario specificare almeno una di queste opzioni. La tabella visualizza combinazioni valide di opzioni di accesso o di creazione.

Tabella 727. Combinazioni valide di opzioni di accesso e creazione

Combinazione di opzioni	**Note**
SOCRT	Crea una sottoscrizione se non esiste; non riesce se la sottoscrizione esiste.
SORES	Riprende una sottoscrizione esistente, non riesce se non esiste alcuna sottoscrizione.
SOCRT + SORES	Crea una sottoscrizione se non esiste e ne riprende una corrispondente, se esiste. Combinazione utile se utilizzata in un'applicazione che potrebbe essere eseguita più volte.
SORES + SOALT (vedi nota)	Riprende una sottoscrizione esistente, modificando i campi in modo che corrispondano a quelli specificati in MQSD, non riesce se non esiste alcuna sottoscrizione.

Tabella 727. Combinazioni valide di opzioni di accesso e creazione (Continua)	
Combinazione di opzioni	**Note**
SOCRT + SOALT (vedere nota)	Crea una sottoscrizione se non ne esiste una e ne riprende una corrispondente, se esiste, modificando i campi in modo che corrispondano a quelli specificati in MQSD. Combinazione utile se utilizzata in un'applicazione che desidera assicurarsi che la propria sottoscrizione si trovi in un determinato stato prima di continuare.

Nota:

Le opzioni che specificano SOALT possono specificare anche SORES, ma questa combinazione non ha alcun effetto aggiuntivo sulla specifica di SOALT da sola. SOALT implica SORES, poiché il richiamo di MQSUB per modificare una sottoscrizione implica anche la ripresa delle sottoscrizioni. Ma non è vero il contrario: riprendere un abbonamento non significa modificarlo.

SOCRT

Crea una sottoscrizione per l'argomento specificato. Se esiste una sottoscrizione che utilizza lo stesso *SDSN* , la chiamata non riesce con RC2432 . Questo errore può essere evitato combinando l'opzione SOCRT con SORES. *SDSN* non è sempre necessario. Per ulteriori dettagli, consultare la descrizione di tale campo.

La combinazione di SOCRT con SORES verifica prima se è presente una sottoscrizione esistente per il *SDSN*specificato e se è presente un handle per tale sottoscrizione preesistente; ma se non è presente alcuna sottoscrizione esistente, ne viene creata una nuova utilizzando tutti i campi forniti in MQSD.

SOCRT può anche essere combinato con SOALT per ottenere un effetto simile (vedere i dettagli su SOALT più avanti in questo argomento).

SORES

Restituisce un handle a una sottoscrizione preesistente che corrisponde a quelli specificati da *SDSN*. Non vengono apportate modifiche agli attributi di sottoscrizione corrispondenti e vengono restituiti nell'output della struttura MQSD. La maggior parte del contenuto di MQSD non è utilizzata: i campi utilizzati sono *SDSID*, *SDVER*, *SDOPT*, *SDAID* e *SDASI*e *SDSN*.

La chiamata non riesce con il codice di errore RC2428 se non esiste una sottoscrizione corrispondente al nome completo della sottoscrizione. Questo errore può essere evitato combinando l'opzione SOCRT con SORES. Per i dettagli su SOCRT, consultare SOCRT.

L'ID utente della sottoscrizione è l'ID utente che ha creato la sottoscrizione oppure, se è stato successivamente modificato da un ID utente diverso, è l'ID utente della modifica più recente e riuscita. Se viene utilizzato *SDAID* e l'utilizzo di ID utente alternativi è consentito per tale utente, *SDAID* viene registrato come l'ID utente che ha creato la sottoscrizione invece dell'ID utente con cui è stata effettuata la sottoscrizione.

L'ID utente che ha creato la sottoscrizione viene registrato come *SDAU* se tale campo viene utilizzato e l'utilizzo di ID utente alternativi è consentito per tale utente.

Se esiste una sottoscrizione corrispondente che è stata creata senza l'opzione SOAUID e l'ID utente della sottoscrizione è diverso da quello dell'applicazione che richiede un handle per la sottoscrizione, la chiamata ha esito negativo con il codice di errore RC2434 .

Se esiste una sottoscrizione corrispondente ed è attualmente utilizzata da un'altra applicazione, la chiamata ha esito negativo con il codice di errore RC2429 . Se è attualmente utilizzato dalla stessa connessione, la chiamata non ha esito negativo e viene restituito un handle per la sottoscrizione.

Se la sottoscrizione indicata in SubName non è una sottoscrizione valida da riprendere o modificare da un'applicazione, la chiamata non riesce con RC2523 .

SORES è implicito in SOALT e pertanto non è necessario combinarlo con tale opzione, tuttavia, non è un errore se queste due opzioni sono combinate.

SOALT

Restituisce un handle a una sottoscrizione preesistente con il nome sottoscrizione completo corrispondente a quelli specificati in *SDSN*. Tutti gli attributi della sottoscrizione diversi da quelli specificati in MQSD vengono modificati nella sottoscrizione a meno che la modifica non sia disconsentita per tale attributo. I dettagli sono riportati nella descrizione di ciascun attributo e riepilogati nella seguente tabella. Se si tenta di modificare un attributo che non può essere modificato, la chiamata ha esito negativo con il codice di errore mostrato nella seguente tabella.

La chiamata non riesce con il codice di errore RC2428 se non esiste una sottoscrizione corrispondente al nome completo della sottoscrizione. Questo errore può essere evitato combinando l'opzione SOCRT con SOALT.

La combinazione di SOCRT con SOALT verifica prima se esiste una sottoscrizione esistente per il nome sottoscrizione completo specificato e se viene restituito un handle a quella sottoscrizione preesistente con le modifiche effettuate come precedentemente descritto; ma se non esiste alcuna sottoscrizione esistente, ne viene creata una nuova utilizzando tutti i campi forniti in MQSD.

L'ID utente della sottoscrizione è l'ID utente che ha creato la sottoscrizione oppure, se è stato successivamente modificato da un ID utente diverso, è l'ID utente della modifica più recente. Se si utilizza *SDAU* (e l'utilizzo di ID utente alternativi è consentito per tale utente), l'ID utente alternativo viene registrato come l'ID utente che ha creato la sottoscrizione invece dell'ID utente con cui è stata effettuata la sottoscrizione.

Se esiste una sottoscrizione corrispondente che è stata creata senza l'opzione SOAUID e l'ID utente della sottoscrizione è diverso da quello dell'applicazione che richiede un handle per la sottoscrizione, la chiamata non riesce con il codice di errore RC2434 .

Se esiste una sottoscrizione corrispondente ed è attualmente utilizzata da un'altra applicazione, la chiamata ha esito negativo con RC2429 . Se è attualmente utilizzato dalla stessa connessione, la chiamata non ha esito negativo e viene restituito un handle per la sottoscrizione.

Se la sottoscrizione indicata in SubName non è una sottoscrizione valida da riprendere o modificare da un'applicazione, la chiamata non riesce con RC2523 .

Le seguenti tabelle mostrano gli attributi della sottoscrizione che possono essere modificati da SOALT.

Tabella 728. Attributi in MQSD e MQSUB che possono essere modificati

Descrittore tipo di dati o chiamata funzione	**Nome campo**	**Questo attributo può essere modificato utilizzando SOALT?**	**Codice di errore**
MQSD	Opzioni di durata	No	RC2509
MQSD	Opzioni di destinazione	Sì	Nessuna
MQSD	Opzioni di registrazione	Sì (vedere la nota 1)	RC2515 se si tenta di modificare SOGRP
MQSD	Opzioni di pubblicazione	Sì (vedere la nota 2)	Nessuna
MQSD	Opzioni carattere jolly	No	RC2510
MQSD	Altre opzioni	No (vedere la nota 3)	Nessuna
MQSD	ObjectName	No	RC2510
MQSD	SDAU	No (vedere la nota 4)	Nessuna
MQSD	SDASI	No (vedere la nota 4)	Nessuna
MQSD	SDEXP	Sì	Nessuna

Tabella 728. Attributi in MQSD e MQSUB che possono essere modificati (Continua)			
Descrittore tipo di dati o chiamata funzione	**Nome campo**	**Questo attributo può essere modificato utilizzando SOALT?**	**Codice di errore**
MQSD	SDOS	No	RC2510
MQSD	SDSN	No (vedere la nota 5)	Nessuna
MQSD	SDSUD	Sì	Nessuna
MQSD	IDSDC	Sì (vedere la nota 6)	RC2515 quando si trova in una sottoscrizione raggruppata
MQSD	SDPRI	Sì	Nessuna
MQSD	SDACC	Sì	Nessuna
MQSD	ID SDA	Sì	Nessuna
MQSD	SDSL	No	RC2512
MQSUB	HOBJ	Sì (vedere la nota 6)	RC2515 quando si trova in una sottoscrizione raggruppata

Note:

1. SOGRP non può essere modificato.
2. SONEWP non può essere modificato perché non fa parte della sottoscrizione
3. Queste opzioni non fanno parte della sottoscrizione
4. Questo attributo non fa parte della sottoscrizione
5. Questo attributo è l'identità della sottoscrizione che si sta modificando
6. Modificabile tranne quando parte di una parte raggruppata (SOGRP)

Opzioni di durata: le seguenti opzioni controllano la durata della sottoscrizione. È possibile specificare solo una di queste opzioni. Se si sta modificando una sottoscrizione esistente utilizzando l'opzione SOALT , non è possibile modificare la durata della sottoscrizione. Al ritorno da una chiamata MQSUB che utilizza SORES, viene impostata l'opzione di durata appropriata.

SODUR
Richiedere che la sottoscrizione a questo argomento rimanga fino a che non viene esplicitamente rimossa utilizzando MQCLOSE con l'opzione CORMSB . Se questa sottoscrizione non viene esplicitamente rimossa, rimarrà anche dopo la chiusura di questa applicazione che si connette al gestore code.

Se una sottoscrizione durevole viene richiesta a un argomento definito come non consentendo sottoscrizioni durevoli, la chiamata non riesce con RC2436 .

SONDUR
Richiedere che la sottoscrizione a questo argomento venga rimossa quando la connessione dell'applicazione al gestore code viene chiusa, se non è già stata esplicitamente rimossa. SONDUR è l'opposto dell'opzione SODUR ed è definito per la documentazione del programma. È il valore predefinito se non viene specificato nessuno dei due.

Opzioni di destinazione: le seguenti opzioni controllano la destinazione a cui vengono inviate le pubblicazioni per un argomento a cui è stata effettuata la sottoscrizione. Se si modifica una sottoscrizione esistente utilizzando l'opzione SOALT, la destinazione utilizzata per le pubblicazioni per la sottoscrizione può essere modificata. Al ritorno da una chiamata MQSUB utilizzando SORES, questa opzione viene impostata, se appropriato.

SOMAN

Richiedere che la destinazione a cui vengono inviate le pubblicazioni sia gestita dal gestore code.

L'handle dell'oggetto restituito in *HOBJ* rappresenta una coda gestita del gestore code ed è da utilizzare con le successive chiamate MQGET, MQCB, MQINQ o MQCLOSE.

Non è possibile fornire un handle di oggetto restituito da una precedente chiamata MQSUB nel parametro **Hobj** quando SOMAN non viene specificato.

Opzioni di registrazione: le seguenti opzioni controllano i dettagli della registrazione effettuata al gestore code per questa sottoscrizione. Se si modifica una sottoscrizione esistente utilizzando l'opzione SOALT , è possibile modificare tali opzioni di registrazione. Al ritorno da una chiamata MQSUB utilizzando SORES , vengono impostate le opzioni di registrazione appropriate.

SOGRP

Questa sottoscrizione viene raggruppata con altre sottoscrizioni dello stesso *SDSL* utilizzando la stessa coda e specificando lo stesso ID correlazione in modo che tutte le pubblicazioni per argomenti che potrebbero causare la fornitura di più di un messaggio di pubblicazione al gruppo di sottoscrizioni, a causa della sovrapposizione di una serie di stringhe di argomenti utilizzate, provochino la consegna di un solo messaggio alla coda. Se questa opzione non viene utilizzata, ogni sottoscrizione univoca (identificata da *SDSN*) che corrisponde viene fornita con una copia della pubblicazione, il che potrebbe significare che più di una copia della pubblicazione potrebbe essere collocata nella coda condivisa da un numero di sottoscrizioni.

Solo la sottoscrizione più significativa nel gruppo viene fornita con una copia della pubblicazione. La sottoscrizione più significativa si basa sul nome dell'argomento completo fino al punto in cui viene trovato un carattere jolly. Se viene utilizzata una combinazione di schemi di caratteri jolly all'interno del gruppo, è importante solo la posizione del carattere jolly. Si consiglia di non combinare diversi schemi di caratteri jolly in un gruppo di sottoscrizioni che condividono la stessa coda.

Quando si crea una nuova sottoscrizione raggruppata, deve ancora avere un *SDSN*univoco, ma se corrisponde al nome dell'argomento completo di una sottoscrizione esistente nel gruppo, la chiamata ha esito negativo con RC2514 .

Se la sottoscrizione più significativa nel gruppo specifica anche SONOLC e si tratta di una pubblicazione dalla stessa applicazione, non viene consegnata alcuna pubblicazione alla coda.

Quando si modifica una sottoscrizione effettuata con questa opzione, non è possibile modificare i campi che implicano il raggruppamento, *Hobj* nella chiamata MQSUB (che rappresentano il nome della coda e del gestore code) e *SDCID* . Se si tenta di modificarli, la chiamata non riesce con RC2515 .

Questa opzione deve essere combinata con SOSCID con un *SDCID* non impostato su CINONEe non può essere combinata con SOMAN.

IDSOA

Quando si specifica SOAUID , l'identit ... del sottoscrittore non Š limitata a un singolo ID utente. Ciò consente a qualsiasi utente di modificare o riprendere la sottoscrizione quando dispone dell'autorizzazione appropriata. Solo un singolo utente può avere la sottoscrizione in qualsiasi momento. Un tentativo di riprendere l'utilizzo di una sottoscrizione attualmente in uso da parte di un'altra applicazione causa l'esito negativo della chiamata con RC2429 .

Per aggiungere questa opzione ad una sottoscrizione esistente, la chiamata MQSUB, utilizzando SOALT, deve provenire dallo stesso ID utente della sottoscrizione originale.

Se una chiamata MQSUB fa riferimento ad una sottoscrizione esistente con SOAUID impostato e l'ID utente è diverso dalla sottoscrizione originale, la chiamata ha esito positivo solo se il nuovo ID utente dispone dell'autorizzazione per sottoscrivere l'argomento. Una volta completato correttamente, le pubblicazioni future per questo sottoscrittore vengono inserite nella coda del sottoscrittore con il nuovo ID utente impostato nel messaggio di pubblicazione.

Non specificare sia SOAUID che SOFUID. Se non viene specificato nessuno dei due, il valore predefinito è SOFUID.

IDSOSO

Quando viene specificato SOFUID , la sottoscrizione può essere modificata o ripresa solo dall'ultimo ID utente che ha modificato la sottoscrizione. Se la sottoscrizione non è stata modificata, è l'ID utente che ha creato la sottoscrizione.

Se un verbo MQSUB fa riferimento a una sottoscrizione esistente con SOAUID impostato e modifica la sottoscrizione utilizzando SOALT per utilizzare SOFUID, l'ID utente della sottoscrizione viene ora fissato con questo nuovo ID utente. La chiamata ha esito positivo solo se il nuovo ID utente dispone dell'autorizzazione per sottoscrivere l'argomento.

Se un ID utente diverso da quello registrato come proprietario di una sottoscrizione tenta di riprendere o modificare una sottoscrizione SOFUID , la chiamata ha esito negativo con RC2434 . L'ID utente proprietario di una sottoscrizione può essere visualizzato utilizzando il comando **DISPLAY SBSTATUS** .

Non specificare sia SOAUID che SOFUID. Se non viene specificato nessuno dei due, il valore predefinito è SOFUID.

Opzioni di pubblicazione: le seguenti opzioni controllano il modo in cui le pubblicazioni vengono inviate a questo sottoscrittore. Se si modifica una sottoscrizione esistente utilizzando l'opzione SOALT , è possibile modificare queste opzioni di pubblicazione.

SONOLC
Comunica al broker che l'applicazione non desidera visualizzare le proprie pubblicazioni. Le pubblicazioni vengono considerate originate dalla stessa applicazione se gli handle di connessione sono gli stessi. Al ritorno da una chiamata MQSUB utilizzando SORES , questa opzione è impostata, se appropriato.

SONEWP
Nessuna pubblicazione attualmente conservata deve essere inviata, quando viene creata questa sottoscrizione, solo nuove pubblicazioni. Questa opzione si applica solo quando è specificato SOCRE . Eventuali modifiche successive ad una sottoscrizione non modificano il flusso di pubblicazioni e, pertanto, tutte le pubblicazioni che sono state conservate su un argomento, sono già state inviate al sottoscrittore come nuove pubblicazioni.

Se questa opzione viene specificata senza SOCRE , la chiamata non riesce con RC2046 . Al ritorno da una chiamata MQSUB che utilizza SORES , questa opzione non viene impostata anche se la sottoscrizione è stata creata utilizzando questa opzione.

Se questa opzione non viene utilizzata, i messaggi precedentemente conservati vengono inviati alla coda di destinazione fornita. Se questa azione non riesce a causa di un errore, RC2525 o RC2526 , la creazione della sottoscrizione non riesce.

Questa opzione non è valida in combinazione con SOPUBR.

SOPUBR
L'impostazione di questa opzione indica che il sottoscrittore richiede le informazioni in modo specifico quando richiesto. Il gestore code non invia messaggi non richiesti al sottoscrittore. La pubblicazione conservata (o probabilmente più pubblicazioni se viene specificato un carattere jolly nell'argomento) viene inviata al sottoscrittore ogni volta che viene effettuata una chiamata MQSUBRQ utilizzando l'handle Hsub da una precedente chiamata MQSUB. Non viene inviata alcuna pubblicazione come risultato della chiamata MQSUB utilizzando questa opzione. Al ritorno da una chiamata MQSUB utilizzando SORES , questa opzione è impostata, se appropriato.

Questa opzione non è valida in combinazione con SONEWP.

Opzioni carattere jolly: le seguenti opzioni controllano come i caratteri jolly vengono interpretati nella stringa fornita nel campo *SDOS* di MQSD. È possibile specificare solo una di queste opzioni. Se si modifica una sottoscrizione esistente utilizzando l'opzione SOALT , queste opzioni jolly non possono essere modificate. Al ritorno da una chiamata MQSUB utilizzando SORES , viene impostata l'opzione del carattere jolly appropriata.

CHRSOW

I caratteri jolly operano solo sui caratteri all'interno della stringa argomento. Il campo SOWCHR considera la barra (/) come un altro carattere senza alcuna significatività speciale.

Il comportamento definito da SOWCHR viene mostrato nella seguente tabella:

Tabella 729. Come vengono interpretati i caratteri jolly

Carattere speciale	Comportamento
*	Carattere jolly, zero o più caratteri
?	Carattere jolly, un carattere
%	Carattere di escape per consentire ai caratteri '*', '?' o '%' di essere utilizzati in una stringa e non essere interpretati come un carattere speciale, ad esempio, '% *', '%?' o' %% '.

Ad esempio, la pubblicazione sul seguente argomento:

```
/level0/level1/level2/level3/level4
```

corrisponde ai sottoscrittori utilizzando i seguenti argomenti:

```
*
/*
/ level0/level1/level2/level3/*
/ level0/level1/*/level3/level4
/ level0/level1/le?el2/level3/level4
```

Nota: Questo utilizzo di caratteri jolly fornisce esattamente il significato fornito in IBM MQ V6 e WebSphere MB V6 quando si utilizzano i messaggi formattati MQRFH1 per la pubblicazione / sottoscrizione. Si consiglia di non utilizzarlo per le applicazioni appena scritte e di utilizzarlo solo per le applicazioni che erano precedentemente in esecuzione rispetto a tale versione e che non sono state modificate per utilizzare il comportamento del carattere jolly predefinito come descritto in SOWTOP.

SOWTOP

I caratteri jolly operano solo sugli elementi argomento all'interno della stringa argomento. Questo è il comportamento predefinito se non ne viene scelto alcuno.

Il comportamento richiesto da SOWTOP viene mostrato nella seguente tabella:

Tabella 730. Come vengono interpretati i caratteri jolly

Carattere speciale	Comportamento
/	Separatore livello argomento
#	Carattere jolly: più livelli di argomento
+	Carattere jolly: livello argomento singolo

Nota:

I caratteri '+' e '#' non vengono considerati come caratteri jolly se vengono combinati con altri caratteri (inclusi se stessi) all'interno di un livello di argomento. Nella seguente stringa, i caratteri '#' e '+' vengono considerati come normali caratteri.

```
level0/level1/#+/level3/level#
```

Ad esempio, la pubblicazione sul seguente argomento:

```
/level0/level1/level2/level3/level4
```

corrisponde ai sottoscrittori utilizzando i seguenti argomenti:

```
#
/#
/ level0/level1/level2/level3/#
/ level0/level1/+/level3/level4
```

Nota: Questo utilizzo di caratteri jolly fornisce il significato fornito in WebSphere Message Broker 6 quando si utilizzano messaggi formattati MQRFH2 per la pubblicazione / sottoscrizione.

Altre opzioni: le seguenti opzioni controllano il modo in cui viene emessa la chiamata API anziché la sottoscrizione. Al ritorno da una chiamata MQSUB che utilizza S0RES , queste opzioni non vengono modificate.

SOALTU
Il campo SDAU contiene un identificativo utente da utilizzare per convalidare questa chiamata MQSUB. La chiamata può avere esito positivo solo se questa SDAU è autorizzata ad aprire l'oggetto con le opzioni di accesso specificate, indipendentemente dal fatto che l'identificativo utente con cui l'applicazione è in esecuzione sia autorizzato a farlo.

IDSOS
La sottoscrizione deve utilizzare l'identificativo di correlazione fornito nel campo *SDCID* . Se questa opzione non viene specificata, un identificativo di correlazione viene creato automaticamente dal gestore code al momento della sottoscrizione e viene restituito all'applicazione nel campo *SDCID* . Consultare SDCID (stringa di bit a 24 byte) SDCID per ulteriori informazioni.

SOSETI

La sottoscrizione utilizza il token di account e i dati di identità dell'applicazione forniti nei campi *SDACC* e *SDAID* .

Se questa opzione viene specificata, viene eseguito lo stesso controllo di autorizzazione come se si accedesse alla coda di destinazione utilizzando una chiamata MQOPEN con 00SETI, ad eccezione del caso in cui viene utilizzata anche l'opzione S0MAN , nel qual caso non vi è alcun controllo di autorizzazione sulla coda di destinazione.

Se questa opzione non viene specificata, le pubblicazioni inviate a questo sottoscrittore hanno le informazioni di contesto predefinite associate come segue:

Tabella 731. Informazioni di contesto predefinite per pubblicazioni inviate a questo sottoscrittore

Campo in MQMD	**Valore utilizzato**
MDUID	L'ID utente associato alla sottoscrizione nel momento in cui è stata effettuata la sottoscrizione.
MDACC	Determinato dall'ambiente, se possibile; impostare su ACNONE in caso contrario.
MDAID	Imposta su spazi vuoti

Questa opzione è valida solo con S0CRE e S0ALT. Se utilizzato con S0RES, i campi *SDACC* e *SDAID* vengono ignorati, quindi questa opzione non ha alcun effetto.

Se una sottoscrizione viene modificata senza utilizzare questa opzione, dove in precedenza la sottoscrizione aveva fornito le informazioni sul contesto di identità, vengono generate le informazioni sul contesto predefinito per la sottoscrizione modificata.

Se una sottoscrizione che consente a ID utente differenti di utilizzarla con l'opzione S0AUIDviene ripresa da un ID utente differente, viene generato un contesto di identit ...

predefinito per il nuovo ID utente che ora possiede la sottoscrizione e tutte le pubblicazioni successive vengono consegnate contenenti il nuovo contesto di identit ....

SOFIQ
La chiamata MQSUB ha esito negativo se il gestore code è in stato di inattività. Su z/OS, per un'applicazione CICS o IMS , questa opzione forza anche l'esito negativo della chiamata MQSUB se la connessione è in stato di inattività.

SDAU (stringa di caratteri a 12 byte)

Se si specifica S0ALTU, questo campo contiene un identificativo utente alternativo utilizzato per controllare l'autorizzazione per la sottoscrizione e per l'output nella coda di destinazione (specificato nel parametro **Hobj** della chiamata MQSUB), al posto dell'identificativo utente con cui l'applicazione è attualmente in esecuzione.

Se l'operazione ha esito positivo, l'identificativo utente specificato in questo campo viene registrato come l'identificativo utente proprietario della sottoscrizione al posto dell'identificativo utente con cui è attualmente in esecuzione l'applicazione.

Se si specifica S0ALTU e questo campo è completamente vuoto fino al primo carattere null o alla fine del campo, la sottoscrizione può avere esito positivo solo se nessuna autorizzazione utente deve sottoscriversi a questo argomento con le opzioni specificate o la coda di destinazione per l'emissione.

Se S0ALTU non viene specificato, questo campo viene ignorato.

Al ritorno da una chiamata MQSUB utilizzando S0RES, questo campo non viene modificato.

Questo è un campo di immissione. La lunghezza di questo campo è fornita da LNUID. Il valore iniziale di questo campo è di 12 caratteri vuoti.

SDPRI (numero intero con segno a 10 cifre)

Questo è il valore che si trova nel campo *MQPRI* di MQMD (Message Descriptor) di tutti i messaggi di pubblicazione corrispondenti a questa sottoscrizione. Per ulteriori informazioni sul campo *MQPRI* in MQMD, consultare MDPRI.

Il valore deve essere maggiore o uguale a zero; zero è la priorità più bassa. È inoltre possibile utilizzare i seguenti valori speciali:

PRQDEF

Quando una coda di sottoscrizione viene fornita nel campo Hobj nella chiamata MQSUB e non è un handle gestito, la priorità del messaggio viene presa dall'attributo **DefPriority** di questa coda. Se la coda così identificata è una coda cluster o è presente più di una definizione nel percorso di risoluzione del nome della coda, la priorità viene determinata quando il messaggio di pubblicazione viene inserito nella coda come descritto per MDPRI.

Se la chiamata MQSUB utilizza un handle gestito, la priorità per il messaggio viene ricavata dall'attributo **DefPriority** della coda modello associata all'argomento sottoscritto.

PRPUB
La priorità del messaggio è la priorità della pubblicazione originale. Questo è il valore iniziale del campo.

Se si modifica una sottoscrizione esistente utilizzando l'opzione S0ALT , è possibile modificare il *MQPRI* di eventuali messaggi di pubblicazione futuri.

Al ritorno da una chiamata MQSUB utilizzando S0RES, questo campo è impostato sulla priorità corrente utilizzata per la sottoscrizione.

SDRO (MQCHARV)

SDRO è il nome oggetto lungo dopo che il gestore code ha risolto il nome fornito in *SDON*.

Se il nome oggetto lungo viene fornito in *SDOS* e non viene fornito nulla in *SDON*, il valore restituito in questo campo è uguale a quello fornito in *SDOS*.

Se questo campo viene omesso (ovvero SDRO.VSBufSize è zero), il *SDRO* non viene restituito, ma la lunghezza viene restituita in SDRO SDRO.VSLength. Se la lunghezza è più breve del *SDRO*completo, viene troncato e restituisce il numero massimo di caratteri più a destra che possono rientrare nella lunghezza fornita.

Se *SDRO* viene specificato in modo non corretto, in base alla descrizione di come utilizzare la struttura MQCHARV o se supera la lunghezza massima, la chiamata ha esito negativo con il codice di errore RC2520 .

SDSID (stringa di caratteri a 4 byte)

Questo è l'identificatore della struttura; il valore deve essere:

SDSIDV
Identificativo per la struttura del descrittore di sottoscrizione.

Questo è sempre un campo di input. Il valore iniziale di questo campo è SDSIDV

SDSL (numero intero con segno a 10 cifre)

Questo è il livello associato alla sottoscrizione. Le pubblicazioni vengono consegnate a questa sottoscrizione solo se si trova nella serie di sottoscrizioni con il valore *SDSL* più alto minore o uguale al PubLevel utilizzato al momento della pubblicazione.

Il valore deve essere compreso tra zero e 9. Zero è il livello più basso.

Il valore iniziale di questo campo è 1.

Se si modifica una sottoscrizione esistente utilizzando l'opzione SOALT , *SDSL* non può essere modificato.

SDSN (MQCHARV)

SDSN specifica il nome della sottoscrizione.

Questo campo è obbligatorio solo se *SDOPT* specifica l'opzione SODUR , ma se viene fornito viene utilizzato anche dal gestore code per SONDUR . Se specificato, *SDSN* deve essere univoco all'interno del gestore code, poiché è il campo utilizzato per identificare le sottoscrizioni.

La lunghezza massima di *SDSN* è 10240.

Questo campo ha due scopi. Per una sottoscrizione SODUR, è il mezzo mediante il quale si identifica una sottoscrizione per riprenderla dopo che è stata creata, se è stato chiuso l'handle per la sottoscrizione (utilizzando l'opzione COKPSB) o è stato disconnesso dal gestore code. L'identificazione di una sottoscrizione per rimuoverla dopo che è stata creata viene eseguita utilizzando la chiamata MQSUB con l'opzione SORES . Il campo SDSN viene visualizzato anche nella vista di amministrazione delle sottoscrizioni nel campo *SDSN* in DISPLAY SBSTATUS.

Se *SDSN* viene specificato in modo non corretto, in base alla descrizione di come utilizzare la struttura MQCHARV , se supera la lunghezza massima o se viene omesso quando è richiesto (ovvero *SDSN*. *VCHRL* è zero) o se supera la lunghezza massima, la chiamata ha esito negativo con codice di errore RC2440 .

Questo è un campo di immissione. I valori iniziali dei campi in questa struttura sono gli stessi della struttura MQCHARV.

Se si modifica una sottoscrizione esistente utilizzando l'opzione SOALT , il nome della sottoscrizione non può essere modificato perché è il campo utilizzato per identificare la sottoscrizione. Non viene modificato nell'output da una chiamata MQSUB con l'opzione SORES .

SDSS (MQCHARV)

SDSS è la stringa che fornisce il criterio di selezione utilizzato durante la sottoscrizione per i messaggi da un argomento.

Questo campo di lunghezza variabile viene restituito all'output di una chiamata MQSUB utilizzando l'opzione SORES , se viene fornito un buffer e se è presente anche una lunghezza buffer positiva in VSBufSize. Se non viene fornito alcun buffer nella chiamata, viene restituita solo la lunghezza della

stringa di selezione nel campo VSLength di MQCHARV. Se il buffer fornito è inferiore allo spazio richiesto per restituire il campo, nel buffer fornito vengono restituiti solo VSBufSize byte.

Se *SDSS* viene specificato in modo non corretto, in base alla descrizione di come utilizzare la struttura MQCHARV o se supera la lunghezza massima, la chiamata ha esito negativo con il codice di errore RC2519 .

SDSUD (MQCHARV)

I dati forniti nella sottoscrizione in questo campo sono inclusi come proprietà del messaggio mq.SubUserData di ogni pubblicazione inviata a questa sottoscrizione.

La lunghezza massima di *SDSUD* è 10240.

Se *SDSUD* viene specificato in modo non corretto, in base alla descrizione di come utilizzare la struttura MQCHARV o se supera la lunghezza massima, la chiamata ha esito negativo con codice motivo RC2431.

Questo è un campo di immissione. I valori iniziali dei campi in questa struttura sono gli stessi della struttura MQCHARV.

Se si modifica una sottoscrizione esistente utilizzando l'opzione SOALT , è possibile modificare i dati utente della sottoscrizione.

Questo campo di lunghezza variabile viene restituito all'output di una chiamata MQSUB utilizzando l'opzione SORES , se viene fornito un buffer e in *VSBufLen*è presente una lunghezza buffer positiva. Se non viene fornito alcun buffer nella chiamata, nel campo *VCHRL* di MQCHARV viene restituita solo la lunghezza dei dati utente della sottoscrizione. Se il buffer fornito è più piccolo dello spazio richiesto per restituire il campo, nel buffer fornito vengono restituiti solo *VSBufLen* byte.

SDVER (numero intero con segno a 10 cifre)

Questo è il numero di versione della struttura; il valore deve essere:

SDVER1
Version-1 La struttura del descrittore di sottoscrizione.

La seguente costante specifica il numero di versione della versione corrente:

SDVERC
La versione corrente della struttura del descrittore di sottoscrizione.

Questo è sempre un campo di input. Il valore iniziale del campo è SDVER1.

Valori iniziali

Tabella 732. Campi in MQSD

Nome campo	Nome della costante	Valore della costante
SDSID	SDSIDV	'SD¬¬'
SDVER	SDVER1	1
SDOPT	SONDUR	0
SDON	Nessuna	Spazi
SDAU	Nessuna	Spazi
SDASI	SINONE	Valori null
SDEXP	EIULIM	-1
SDOS	Nomi e valori come definiti per MQCHARV	

Tabella 732. Campi in MQSD (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
SDSN	Nomi e valori come definiti per MQCHARV	
SDSUD	Nomi e valori come definiti per MQCHARV	
SDCID	CINONE	Valori null
SDPRI	PRQDEF	-3
SDACC	ACNONE	Valori null
SDAID	Nessuna	Spazi
SDSL	Nessuna	1
SDRO	Nomi e valori come definiti in MQCHARV	
Nota: 1. Il simbolo ¬ rappresenta un singolo carattere vuoto.		

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQSD Structure
D*
D* Structure identifier
D  SDSID                  1      4
D* Structure version number
D  SDVER                  5      8I 0
D* Options associated with subscribing
D  SDOPT                  9     12I 0
D* Object name
D  SDON                  13     60
D* Alternate user identifier
D  SDAU                  61     72
D* Alternate security identifier
D  SDASI                 73    112
D* Expiry of Subscription
D  SDEXP                113    116I 0
D* Object Long name
D SDOSP                 117    132*
D SDOSO                 133    136I 0
D SDOSS                 137    140I 0
D SDOSL                 141    144I 0
D SDOSC                 145    148I 0
D* Subscription name
D SDSNP                 149    164*
D SDSNO                 165    168I 0
D SDSNS                 169    172I 0
D SDSNL                 173    176I 0
D SDSNC                 177    180I 0
D* Subscription User data
D SDSUDP                181    196*
D SDSUDO                197    200I 0
D SDSUDS                201    204I 0
D SDSUDL                205    208I 0
D SDSUDC                209    212I 0
D* Correlation Id related to this subscription
D SDCID                 213    236
D* Priority set in publications
D SDPRI                 237    240I 0
D* Accounting Token set in publications
D SDACC                 241    272
D* Appl Identity Data set in publications
D SDAID                 273    304
D* Message Selector
```

```
D SDSSP                 305    320*
D SDSSO                 321    324I 0
D SDSSS                 325    328I 0
D SDSSL                 329    332I 0
D SDSSC                 333    336
D* Subscription level
D SDSL                  337    340  0
D* Resolved Long object name
D SDROP                 341    356*
D SDROO                 357    360I 0
D SDROS                 361    364I 0
D SDROL                 365    368I 0
D SDROC                 369    372I 0
```

IBM i MQSMPO (Impostazione delle opzioni della proprietà del messaggio) su IBM i

La struttura **MQSMPO** consente alle applicazioni di specificare le opzioni che controllano la modalità di impostazione delle proprietà dei messaggi.

Panoramica

Scopo: La struttura è un parametro di immissione sulla chiamata **MQSETMP** .

Serie di caratteri e codifica: i dati in **MQSMPO** devono essere nella serie di caratteri dell'applicazione e nella codifica dell'applicazione (ENNAT).

- "Campi" a pagina 1263
- "Valori iniziali" a pagina 1264
- "Dichiarazione RPG" a pagina 1264

Campi

La struttura MQSMPO contiene i seguenti campi, descritti in **ordine alfabetico**:

SPOPT (numero intero con segno a 10 cifre)

Opzioni di posizione: le opzioni riportate di seguito si riferiscono alla posizione relativa della proprietà confrontata con il cursore della proprietà:

SSETF
Imposta il valore della prima proprietà che corrisponde al nome specificato oppure, se non esiste, aggiunge una nuova proprietà dopo tutte le altre proprietà con una gerarchia corrispondente.

SPSETC
Imposta il valore della proprietà a cui punta il cursore della proprietà. La proprietà indicata dal cursore della proprietà è quella che è stata interrogata l'ultima volta utilizzando l'opzione IPINQF o IPINQN.

Il cursore della proprietà viene reimpostato quando viene riutilizzato l'handle del messaggio o quando l'handle del messaggio viene specificato nel campo *HMSG* della struttura MQGMO su una chiamata MQGET o la struttura MQPMO su una chiamata MQPUT.

Se questa opzione viene utilizzata quando il cursore della proprietà non è ancora stato stabilito o se la proprietà indicata dal cursore della proprietà è stata eliminata, la chiamata ha esito negativo con codice di completamento CCFAIL e codice motivo RC2471.

SPSETA
Imposta una nuova proprietà dopo la proprietà indicata dal cursore della proprietà. La proprietà indicata dal cursore della proprietà è quella che è stata interrogata per ultima utilizzando l'opzione IPINQF o IPINQO.

Il cursore della proprietà viene reimpostato quando viene riutilizzato l'handle del messaggio o quando l'handle del messaggio viene specificato nel campo *HMSG* della struttura MQGMO su una chiamata MQGET o la struttura MQPMO su una chiamata MQPUT.

Se questa opzione viene utilizzata quando il cursore della proprietà non è ancora stato stabilito o se la proprietà indicata dal cursore della proprietà è stata eliminata, la chiamata ha esito negativo con codice di completamento CCFAIL e codice motivo RC2471.

Se non è necessaria alcuna delle opzioni descritte, utilizzare la seguente opzione:

SPNONE
Nessuna opzione specificata.

Questo è sempre un campo di input. Il valore iniziale di questo campo è SPSETF.

SPSID (numero intero con segno a 10 cifre)

Questo è l'identificatore della struttura; il valore deve essere:

SSID

Identificativo per la struttura di opzioni della proprietà del messaggio impostato.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **SPSIDV**.

SPVAKCSI (numero intero con segno a 10 cifre)

La serie di caratteri del valore della proprietà da impostare se il valore è una stringa di caratteri.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **CSAPL**.

SPVALENC (numero intero con segno a 10 cifre)

La codifica del valore della proprietà da impostare se il valore è numerico.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **ENNAT**.

SPVER (numero intero con segno a 10 cifre)

Questo è il numero di versione della struttura; il valore deve essere:

SPVER1
Version-1 imposta la struttura delle opzioni delle proprietà del messaggio.

La seguente costante specifica il numero di versione della versione corrente:

SPVERC
Versione corrente della struttura di opzioni della proprietà del messaggio impostato.

Questo è sempre un campo di input. Il valore iniziale di questo campo è **SPVER1**.

Valori iniziali

Tabella 733. Campi in MQSMPO

Nome campo	Nome della costante	Valore della costante
SPSID	SSID	'SMPO'
SPVER	SPVER1	1
SPOPT	SPNONE	0
SPVALENC	ENNAT	Dipende dall'ambiente
SPVALCSI	CSAPL	-3

Dichiarazione RPG

```
D* MQSMPO Structure
D*
D*
```

```
     D* Structure identifier
     D  SPSID                  1      4    INZ('SMPO')
     D*
     D* Structure version number
     D  SPVER                  5      8I 0 INZ(1)
     D*
     ** Options that control the action of
     D* MQSETMP
     D  SPOPT                  9     12I 0 INZ(0)
     D*
     D* Encoding of Value
     D  SPVALENC              13     16I 0 INZ(273)
     D*
     D* Character set identifier of Value
     D  SPVALCSI              17     20I 0 INZ(-3)
```

IBM i MQSRO (Subscription Request Options) su IBM i

La struttura MQSRO consente all'applicazione di specificare le opzioni che controllano come viene effettuata una richiesta di sottoscrizione.

Panoramica

Scopo: La struttura è un parametro di input/output nella chiamata MQSUBRQ.

Versione: la versione corrente di MQSRO è SRVER1.

- "Campi" a pagina 1265
- "Valori iniziali" a pagina 1266
- "Dichiarazione RPG" a pagina 1266

Campi

La struttura MQSRO contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

SRNMP (numero intero con segno a 10 cifre)

Si tratta di un campo di emissione, restituito all'applicazione per indicare il numero di pubblicazioni inviate alla coda di sottoscrizione come risultato di questa chiamata. Sebbene questo numero di pubblicazioni sia stato inviato come risultato di questa chiamata, non vi è alcuna garanzia che questo numero di messaggi sarà disponibile per l'applicazione, soprattutto se si tratta di messaggi non persistenti.

È possibile che vi sia più di una pubblicazione se l'argomento sottoscritto conteneva un carattere jolly. Se non era presente alcun carattere jolly nella stringa di argomenti quando è stata creata la sottoscrizione rappresentata da *HSUB* , viene inviata al massimo una pubblicazione come risultato di questa chiamata.

SROPT (numero intero con segno a 10 cifre)

È necessario specificare una delle seguenti opzioni. È possibile specificare una sola opzione.

Altre opzioni: la seguente opzione controlla cosa accade quando il gestore code è in fase di sospensione:

SRFIQ

La chiamata MQSUBRQ ha esito negativo se il gestore code è in stato di inattività.

Opzione predefinita: se l'opzione descritta precedentemente non è richiesta, è necessario utilizzare la seguente opzione:

NESSUNO

Utilizzare questo valore per indicare che non sono state specificate altre opzioni; tutte le opzioni assumono i propri valori predefiniti.

SRNONE aiuta la documentazione del programma. Sebbene non sia previsto che questa opzione venga utilizzata con altre, poiché il suo valore è zero, questo utilizzo non può essere rilevato.

SRSID (stringa di caratteri a 4 byte)

Questo è l'identificatore della struttura; il valore deve essere:

SRSIDV
Identificativo per la struttura SROPT della richiesta di sottoscrizione.

Questo è sempre un campo di input. Il valore iniziale di questo campo è SRSIDV.

SRVER (numero intero con segno a 10 cifre)

Questo è il numero di versione della struttura; il valore deve essere:

SRVER1
Version-1 Struttura Opzioni richiesta di sottoscrizione.

La seguente costante specifica il numero di versione della versione corrente:

SRVERC
La versione corrente della struttura delle opzioni della richiesta di sottoscrizione.

Questo è sempre un campo di input. Il valore iniziale di questo campo è SRVER1.

Valori iniziali

Tabella 734. Campi in MQSRO

Nome campo	Nome della costante	Valore della costante
SRSID	SRSIDV	'SRO¬'
SRVER	SRVER1	1
SROPT	NESSUNO	0
SRNMP	Nessuna	0

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Il valore Stringa null o spazi vuoti indica la stringa null in C e gli spazi vuoti in altri linguaggi di programmazione.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQSRO Structure
D*
D* Structure identifier
D  SRSID                  1      4
D* Structure version number
D  SRVER                  5      8I 0
D* Options that control the action of MQSUBRQ
D  SROPT                  9     12I 0
D* Number of publications sent
D  SRNMP                 13     16I 0
```

IBM i MQSTS (Status reporting structure) su IBM i

La struttura MQSTS descrive i dati nella struttura di stato restituita dal comando MQSTAT.

Panoramica

Serie di caratteri e codifica: i dati dei caratteri in MQSTS si trovano nella serie di caratteri del gestore code locale; ciò viene fornito dall'attributo del gestore code *CodedCharSetId* . I dati numerici in MQSTS sono nella codifica della macchina nativa; ciò viene fornito da *ENNAT*.

Utilizzo: il comando MQSTAT viene utilizzato per richiamare le informazioni sullo stato. Queste informazioni vengono restituite in una struttura MQSTS. Per informazioni su MQSTAT, consultare "MQSTAT (Richiamo delle informazioni sullo stato) su IBM i" a pagina 1397.

- "Campi" a pagina 1267
- "Valori iniziali" a pagina 1270
- "Dichiarazione RPG" a pagina 1270

Campi

La struttura MQSTS contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

STSCC (numero intero con segno a 10 cifre)

Questo è il codice di completamento risultante dal primo errore riportato nella struttura MQSTS.

Questo è sempre un campo di output. Il valore iniziale di questo campo è CCOK.

STSFC (numero intero con segno a 10 cifre)

Questo è il numero di chiamate put asincrone non riuscite.

Questo è un campo di output. Il valore iniziale di questo campo è 0.

STSOBJN (stringa di caratteri a 48 byte)

È il nome locale dell'oggetto coinvolto nel primo errore.

Questo è un campo di output. Il valore iniziale di questo campo è di 48 caratteri vuoti.

STSOQMGR (stringa di caratteri a 48 byte)

Questo è il nome del gestore code su cui è definito l'oggetto *STSOBJN* . Un nome che è completamente vuoto fino al primo carattere null o alla fine del campo indica il gestore code a cui è connessa l'applicazione (il gestore code locale).

Questo è un campo di output. Il valore iniziale di questo campo è di 48 caratteri vuoti.

STSOO (numero intero con segno a 10 cifre)

Il STS00 utilizzato per aprire l'oggetto su cui viene creato il report. Presente solo nella versione 2 di MQSTS o superiore.

Il valore di STS00 dipende dal valore del parametro MQSTAT **STYPE** .

STATAPT

Zero.

STATREC

Zero.

STATRER

Il STS00 utilizzato quando si è verificato l'errore. Il motivo dell'errore viene riportato nei campi *STSCC* e *STSRC* nella struttura MQSTS .

STS00 è un campo di output. Il valore iniziale è zero.

STSOS (MQCHARV)

Nome oggetto lungo dell'oggetto in errore su cui si sta eseguendo il report. Presente solo nella versione 2 di MQSTS o superiore.

STS0S è un campo MQCHARV con una lunghezza massima di 10240. Consultare MQCHARV per una descrizione di come utilizzare la struttura MQCHARV.

L'interpretazione di STS0S dipende dal valore del parametro MQSTAT **STYPE** .

STATAPT

Questo è il nome dell'oggetto lungo della coda o dell'argomento utilizzato nell'operazione MQPUT , che non è riuscita.

STATREC

Stringa di lunghezza zero

STATRER

Questo è il nome oggetto lungo dell'oggetto che ha causato la mancata riuscita della riconnessione.

STS0S è un campo di output. Il valore iniziale è una stringa di lunghezza zero.

STSOT (numero intero con segno a 10 cifre)

Il tipo di oggetto denominato in *ObjectName*. I possibili valori sono:

OTALSQ
Coda alias.

OLOCQ
Coda locale.

OTMODQ
Coda modello.

OTQ
Coda.

OTREMQ
Coda remota.

OTTOP
.

Questo è sempre un campo di output. Il valore iniziale di questo campo è OTQ.

STSRC (numero intero con segno a 10 cifre)

Questo è il codice di origine errore risultante dal primo errore riportato nella struttura MQSTS

Questo è sempre un campo di output. Il valore iniziale di questo campo è RCNONE.

STSROBJN (stringa di caratteri a 48 byte)

Questo è il nome della coda di destinazione denominata in *STSOBJN* dopo che il gestore code locale ha risolto il nome. Il nome restituito è il nome di una coda che esiste sul gestore code identificato da *STSRQMGR*.

Un valore non vuoto viene restituito solo se l'oggetto è una singola coda aperta per la ricerca, l'immissione o l'emissione (o qualsiasi combinazione). Se l'oggetto aperto è uno dei seguenti, *STSROBJN* viene impostato su spazi vuoti:

- Un argomento
- Una coda, ma non aperta per la ricerca, l'immissione o l'emissione

Questo è un campo di output. Il valore iniziale di questo campo è di 48 caratteri vuoti.

STSRQMGR (stringa di caratteri a 48 byte)

È il nome del gestore code di destinazione dopo che il gestore code locale ha risolto il nome. Il nome restituito è il nome del gestore code proprietario della coda identificata da *STSROBJN*. *STSRQMGR* può essere il nome del gestore code locale.

Se *STSROBJN* è una coda condivisa di proprietà del gruppo di condivisione code a cui appartiene il gestore code locale, *STSRQMGR* è il nome del gruppo di condivisione code. Se la coda è di proprietà di un altro gruppo di condivisione code, *STSROBJN* può essere il nome del gruppo di condivisione code o il nome di un gestore code che è un membro del gruppo di condivisione code (la natura del valore restituito è determinata dalle definizioni di coda che esistono nel gestore code locale).

Un valore non vuoto viene restituito solo se l'oggetto è una singola coda aperta per la ricerca, l'immissione o l'emissione (o qualsiasi combinazione). Se l'oggetto aperto è uno dei seguenti, *STSRQMGR* viene impostato su spazi vuoti:

- Un argomento
- Una coda, ma non aperta per la ricerca, l'immissione o l'emissione
- Una coda cluster con OOBNDN specificato (o con OOBNDQ attivo quando l'attributo della coda **DefBind** ha il valore OOBNDN)

Questo è un campo di output. Il valore iniziale di questo campo è di 48 caratteri vuoti.

STSSC (numero intero con segno a 10 cifre)

Questo è il numero di chiamate put asincrone riuscite.

Questo è un campo di output. Il valore iniziale di questo campo è 0.

STSSID (stringa di caratteri a 4 byte)

Questo è l'identificativo della struttura. Il valore deve essere:

IDSTSS
Identificativo per la struttura di report di stato.

Il valore iniziale di questo campo è STSSID.

STSSO (numero intero con segno a 10 cifre)

Il STSSO utilizzato per aprire la sottoscrizione in errore. Presente solo nella versione 2 di MQSTS o superiore.

L'interpretazione di STSSO dipende dal valore del parametro MQSTAT **STYPE** .

STATAPT

Zero.

STATREC

Zero.

STATRER

Il STSSO utilizzato quando si è verificato l'errore. Il motivo dell'errore viene riportato nei campi *STSCC* e *STSRC* nella struttura MQSTS . Se l'errore non è relativo alla sottoscrizione di un argomento, il valore restituito è zero.

STSSO è un campo di output. Il valore iniziale è zero.

STSSUN (MQCHARV)

Il nome della sottoscrizione non riuscita. Presente solo nella versione 2 di MQSTS o superiore.

STSSUN è un campo MQCHARV con una lunghezza massima di 10240. Consultare MQCHARV per una descrizione di come utilizzare la struttura MQCHARV.

L'interpretazione di STSSUN dipende dal valore del parametro MQSTAT **STYPE** .

STATAPT

Stringa di lunghezza zero.

STATREC

Stringa di lunghezza zero.

STATRER

Il nome della sottoscrizione che ha causato la mancata riuscita della riconnessione. Se non è disponibile alcun nome di sottoscrizione o se l'errore non è correlato a una sottoscrizione, si tratta di una stringa di lunghezza zero.

STSSUN è un campo di output. Il valore iniziale è una stringa di lunghezza zero.

STSVER (numero intero con segno a 10 cifre)

Questo è il numero di versione della struttura. Il valore deve essere:

STSVR1
Numero di versione per la struttura di reporting di stato.

La seguente costante specifica il numero di versione della versione corrente:

SSVRC
Versione corrente della struttura di report di stato.

Il valore iniziale di questo campo è STSVR1.

STSWC (numero intero con segno a 10 cifre)

Questo è il numero di chiamate di inserimento asincrone completate con un avviso.

Questo è un campo di output. Il valore iniziale di questo campo è 0.

Valori iniziali

Tabella 735. Campi in MQSTS

Nome campo	Nome della costante	Valore della costante
STSSID	SID	
STSVER	SSVRC	STSVR1
STSCC	CCOK	0
STSRC	RCNONE	0
STSSC	Nessuna	0
STSWC	Nessuna	0
STSFC	Nessuna	0
STSOT	Nessuna	0
STSOBJN	Nessuna	Spazi
STSOQMGR	Nessuna	Spazi
STSROBJN	Nessuna	Spazi
STSRQMGR	Nessuna	Spazi
STSOS	Nomi e valori come definiti per MQCHARV	
STSSUN	Nomi e valori come definiti per MQCHARV	
STSOO	Nessuna	0
STSSO	Nessuna	0

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
```

```
D* MQSTS Structure
D*
D* Structure identifier
D  STSSID                 1      4
D* Structure version number
D  STSVER                 5      8I 0
D* Completion code
D  STSCC                  9     12I 0
D* Reason code
D  STSRC                 13     16I 0
D* Success count
D  STSSC                 17     20I 0
D* Warning count
D  STSWC                 21     24I 0
D* Failure count
D  STSFC                 25     28I 0
D* Object type
D  STSOT                 29     32I 0
D* Object name
D  STSOBJN               33     80
D* Object queue manager
D  STSOQMGR              81     128
D* Resolved object name
D  STSROBJN             129     176
D* Resolved object queue manager name
D  STSRQMGR             177     224
D* Ver:1 **
D* Failing object long name
D* Address of variable length string
D  STSOSCHRP            225    240*
D* Offset of variable length string
D  STSOSCHRO            241    244I 0
D* Size of buffer
D  STSOSVSBS            245    248I 0
D* Length of variable length string
D  STSOSCHRL            249    252I 0
D* CCSID of variable length string
D  STSOSCHRC            253    256I 0
D* Failing subscription name
D* Address of variable length string
D  STSSUNCHRP           257    272*
D* Offset of variable length string
D  STSSUNCHRO           273    276I 0
D* Size of buffer
D  STSSUNVSBS           277    280I 0
D* Length of variable length string
D  STSSUNCHRL           281    284I 0
D* CCSID of variable length string
D  STSSUNCHRC           285    288I 0
D* Failing open options
D  STSOO                289    292I 0
D* Failing subscription options
D  STSSO                293    296I 0
D* Ver:2 **
```

MQTM - Messaggio trigger

La struttura MQTM descrive i dati nel messaggio trigger che viene inviato dal gestore code a un'applicazione di controllo trigger quando si verifica un evento trigger per una coda.

Panoramica

Scopo: Questa struttura fa parte di TMI (Trigger Monitor Interface) IBM MQ , che è una delle interfacce framework IBM MQ .

Nome formato: FMTM.

Serie di caratteri e codifica: i dati del carattere in MQTM si trovano nel set di caratteri del gestore code che genera MQTM. I dati numerici in MQTM si trovano nella codifica macchina del gestore code che genera MQTM.

La serie di caratteri e la codifica di MQTM sono fornite dai campi *MDCSI* e *MDENC* in:

- MQMD (se la struttura MQTM si trova all'inizio dei dati del messaggio) oppure
- La struttura dell'intestazione che precede la struttura MQTM (tutti gli altri casi).

Utilizzo: un'applicazione trigger - monitor potrebbe dover passare alcune o tutte le informazioni nel messaggio trigger all'applicazione avviata dall'applicazione trigger - monitor. Le informazioni che possono essere necessarie per l'applicazione avviata includono *TMQN*, *TMTD*e *TMUD*. L'applicazione di controllo dei trigger può passare la struttura MQTM direttamente all'applicazione avviata oppure passare una struttura MQTMC2 , in base a quanto consentito dall'ambiente e conveniente per l'applicazione avviata. Per informazioni su MQTMC2, consultare "MQTMC2 (Trigger message 2 - character format) su IBM i" a pagina 1276.

- Su IBM i, l'applicazione di controllo dei trigger fornita con IBM MQ passa una struttura MQTMC2 all'applicazione avviata.

Per informazioni sui trigger, consultare Prerequisiti per l'attivazione.

MQMD per un messaggio trigger

Tabella 736. Impostazioni per i campi in MQMD di un messaggio trigger generato dal gestore code

Campo in MQMD	Valore utilizzato
MDSID	MSIDV
MDVER	MDVER1
MDREP	RONONE
MDMT	MTDGRM
MDEXP	EIULIM
MDFB	FBNONE
MDENC	ENNAT
MDCSI	Attributo **CodedCharSetId** del Gestore code
MDFMT	FMTM
MDPRI	Attributo **DefPriority** della coda di avvio
MDPER	PENPER
MDMID	Un valore univoco
MDCID	CINONE
MDBOC	0
MDRQ	Spazi
MDRM	Nome del gestore code
MDUID	Spazi
MDACC	ACNONE
MDAID	Spazi
MDPAT	ATQM o come appropriato per l'agent del canale dei messaggi
MDPAN	I primi 28 byte del nome del gestore code
MDPD	Data in cui viene inviato il messaggio trigger
MDPT	Ora in cui viene inviato il messaggio trigger

Tabella 736. Impostazioni per i campi in MQMD di un messaggio trigger generato dal gestore code (Continua)

Campo in MQMD	Valore utilizzato
MDAOD	Spazi

Un'applicazione che genera un messaggio trigger è consigliata per impostare valori simili, ad eccezione dei seguenti:

- Il campo *MDPRI* può essere impostato su PRQDEF (il gestore code lo modificherà con la priorità predefinita per la coda di iniziazione quando il messaggio viene inserito).
- Il campo *MDRM* può essere impostato su spazi vuoti (il gestore code lo modificherà con il nome del gestore code locale quando viene inserito il messaggio).
- I campi di contesto devono essere impostati in modo appropriato per l'applicazione.

Campi

La struttura MQTM contiene i campi riportati di seguito; i campi sono descritti in **ordine alfabetico**:

TMAI (stringa di caratteri a 256 byte)

Identificativo applicazione.

Si tratta di una stringa di caratteri che identifica l'applicazione da avviare e viene utilizzata dall'applicazione di controllo trigger che riceve il messaggio di trigger. Il gestore code inizializza questo campo con il valore dell'attributo **ApplId** dell'oggetto processo identificato dal campo *TMPN* ; consultare “Attributi per le definizioni di processo su IBM i .” a pagina 1437 per i dettagli di questo attributo. Il contenuto di questi dati non è significativo per il gestore code.

Il significato di *TMAI* è determinato dall'applicazione trigger - monitor. Il controllo dei trigger fornito da IBM MQ richiede *TMAI* come nome di un programma eseguibile.

La lunghezza di questo campo è fornita da LNPROA. Il valore iniziale di questo campo è 256 caratteri vuoti.

TMAT (numero intero con segno a 10 cifre)

Il tipo di applicazione.

Ciò identifica la natura del programma da avviare e viene utilizzato dall'applicazione di controllo trigger che riceve il messaggio trigger. Il gestore code inizializza questo campo con il valore dell'attributo **ApplType** dell'oggetto processo identificato dal campo *TMPN* ; consultare “Attributi per le definizioni di processo su IBM i .” a pagina 1437 per i dettagli di questo attributo. Il contenuto di questi dati non è significativo per il gestore code.

TMAT può avere uno dei seguenti valori standard. Possono essere utilizzati anche tipi definiti dall'utente, ma devono essere limitati ai valori compresi tra ATUFST e ATULST:

ACICS
Transazione CICS .

ATVSE
Transazione CICS/VSE .

AT400
Applicazione IBM i .

ATUFST
Il valore più basso per il tipo di applicazione definito dall'utente.

ATULST
Il valore più alto per il tipo di applicazione definito dall'utente.

Il valore iniziale di questo campo è 0.

TMED (stringa di caratteri a 128 byte)

Dati di ambiente.

Si tratta di una stringa di caratteri che contiene informazioni relative all'ambiente relative all'applicazione da avviare e viene utilizzata dall'applicazione di controllo trigger che riceve il messaggio del trigger. Il gestore code inizializza questo campo con il valore dell'attributo **EnvData** dell'oggetto processo identificato dal campo *TMPN* ; consultare "Attributi per le definizioni di processo su IBM i ." a pagina 1437 per i dettagli di questo attributo. Il contenuto di questi dati non è significativo per il gestore code.

La lunghezza di questo campo viene fornita da LNPROE. Il valore iniziale di questo campo è 128 caratteri vuoti.

TMPN (stringa di caratteri a 48 byte)

Nome dell'oggetto processo.

Questo è il nome dell'oggetto processo del gestore code specificato per la coda attivata e può essere utilizzato dall'applicazione di controllo trigger che riceve il messaggio del trigger. Il gestore code inizializza questo campo con il valore dell'attributo **ProcessName** della coda identificata dal campo *TMQN* ; consultare "Attributi per le code" a pagina 1406 per i dettagli di questo attributo.

I nomi più brevi della lunghezza definita del campo vengono sempre riempiti a destra con spazi vuoti; non vengono terminati prematuramente con un carattere null.

La lunghezza di questo campo è fornita da LNPRON. Il valore iniziale di questo campo è di 48 caratteri vuoti.

TMQN (stringa di caratteri a 48 byte)

Nome della coda attivata.

Questo è il nome della coda per cui si è verificato un evento trigger e viene utilizzato dall'applicazione avviata dall'applicazione trigger - monitor. Il gestore code inizializza questo campo con il valore dell'attributo **QName** della coda attivata; consultare "Attributi per le code" a pagina 1406 per i dettagli di questo attributo.

I nomi che sono più brevi della lunghezza definita del campo vengono riempiti a destra con spazi; non vengono terminati prematuramente con un carattere null.

La lunghezza di questo campo è fornita da LNQN. Il valore iniziale di questo campo è di 48 caratteri vuoti.

TMSID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

TMSIDV
Identificativo per la struttura del messaggio trigger.

Il valore iniziale di questo campo è TMSIDV.

TMTD (stringa di caratteri a 64 byte)

I dati del trigger.

Si tratta di dati in formato libero per l'utilizzo da parte dell'applicazione di controllo trigger che riceve il messaggio trigger. Il gestore code inizializza questo campo con il valore dell'attributo **TriggerData** della coda identificata dal campo *TMQN* ; consultare "Attributi per le code" a pagina 1406 per i dettagli di questo attributo. Il contenuto di questi dati non è significativo per il gestore code.

La lunghezza di questo campo viene fornita da LNTRGD. Il valore iniziale di questo campo è di 64 caratteri vuoti.

TMUD (stringa di caratteri a 128 byte)

Dati utente.

Si tratta di una stringa di carattere che contiene le informazioni utente relative all'applicazione da avviare e viene utilizzata dall'applicazione di controllo trigger che riceve il messaggio del trigger. Il gestore code inizializza questo campo con il valore dell'attributo **UserData** dell'oggetto processo identificato dal campo *TMPN* ; consultare “Attributi per le definizioni di processo su IBM i .” a pagina 1437 per i dettagli di questo attributo. Il contenuto di questi dati non è significativo per il gestore code.

La lunghezza di questo campo è fornita da LNPROU. Il valore iniziale di questo campo è 128 caratteri vuoti.

TMVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere:

TMVER1
Numero di versione per la struttura del messaggio trigger.

La seguente costante specifica il numero di versione della versione corrente:

TMVERC
La versione corrente della struttura del messaggio trigger.

Il valore iniziale di questo campo è TMVER1.

Valori iniziali

Tabella 737. Campi in MQTM

Nome campo	Nome della costante	Valore della costante
TMSID	TMSIDV	'TM¬¬'
TMVER	TMVER1	1
TMQN	Nessuna	Spazi
TMPN	Nessuna	Spazi
TMTD	Nessuna	Spazi
TMAT	Nessuna	0
TMAI	Nessuna	Spazi
TMED	Nessuna	Spazi
TMUD	Nessuna	Spazi

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQTM Structure
D*
D* Structure identifier
D  TMSID                  1      4    INZ('TM  ')
D* Structure version number
D  TMVER                  5      8I 0 INZ(1)
D* Name of triggered queue
D  TMQN                   9     56    INZ
D* Name of process object
D  TMPN                  57    104    INZ
D* Trigger data
```

```
     D  TMTD                 105    168    INZ
     D* Application type
     D  TMAT                 169    172I 0 INZ(0)
     D* Application identifier
     D  TMAI                 173    428    INZ
     D* Environment data
     D  TMED                 429    556    INZ
     D* User data
     D  TMUD                 557    684    INZ
```

IBM i MQTMC2 (Trigger message 2 - character format) su IBM i

Quando un'applicazione di controllo trigger richiama un messaggio trigger (MQTM) da una coda di iniziazione, il controllo trigger potrebbe dover trasmettere alcune o tutte le informazioni nel messaggio trigger all'applicazione avviata dal controllo trigger.

Panoramica

Scopo: Le informazioni che possono essere necessarie per l'applicazione avviata includono *TC2QN*, *TC2TD*e *TC2UD*. L'applicazione di controllo dei trigger può passare la struttura MQTM direttamente all'applicazione avviata oppure passare una struttura MQTMC2 , a seconda di ciò che è consentito dall'ambiente e conveniente per l'applicazione avviata.

Questa struttura fa parte di IBM MQ Trigger Monitor Interface (TMI), che è una delle interfacce del framework IBM MQ .

Serie di caratteri e codifica: i dati di caratteri in MQTMC2 si trovano nella serie di caratteri del gestore code locale; ciò viene fornito dall'attributo del gestore code **CodedCharSetId** .

Utilizzo: la struttura MQTMC2 è simile al formato della struttura MQTM. La differenza è che i campi non di carattere in MQTM vengono modificati in MQTMC2 in campi di carattere della stessa lunghezza e il nome gestore code viene aggiunto alla fine della struttura.

- Su IBM i, l'applicazione di controllo dei trigger fornita con IBM MQ passa una struttura MQTMC2 all'applicazione avviata.
- “Campi” a pagina 1276
- “Valori iniziali” a pagina 1277
- “Dichiarazione RPG” a pagina 1278

Campi

La struttura MQTMC2 contiene i campi riportati di seguito; i campi sono descritti in **ordine alfabetico**:

TC2AI (stringa di caratteri a 256 byte)

Identificativo applicazione.

Consultare il campo *TMAI* nella struttura MQTM.

TC2AT (stringa di caratteri a 4 byte)

Il tipo di applicazione.

Questo campo contiene sempre spazi vuoti, qualunque sia il valore nel campo *TMAT* nella struttura MQTM del messaggio trigger originale.

TC2ED (stringa di caratteri a 128 byte)

Dati di ambiente.

Consultare il campo *TMED* nella struttura MQTM.

TC2PN (stringa di caratteri a 48 byte)

Nome dell'oggetto processo.

Consultare il campo *TMPN* nella struttura MQTM.

TC2QMN (stringa di caratteri a 48 byte)

È il nome del gestore code.

Questo è il nome del gestore code in cui si è verificato l'evento trigger.

TC2QN (stringa di caratteri a 48 byte)

Nome della coda attivata.

Consultare il campo *TMQN* nella struttura MQTM.

TC2SID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

TCSIDV
Identificativo per la struttura del messaggio trigger (formato carattere).

TC2TD (stringa di caratteri a 64 byte)

I dati del trigger.

Consultare il campo *TMTD* nella struttura MQTM.

TC2UD (stringa di caratteri a 128 byte)

Dati utente.

Consultare il campo *TMUD* nella struttura MQTM.

TC2VER (stringa di caratteri a 4 byte)

Numero di versione della struttura.

Il valore deve essere:

TCVER2
La versione 2 attiva la struttura del messaggio (formato carattere).

La seguente costante specifica il numero di versione della versione corrente:

TCVERC
La versione corrente della struttura del messaggio trigger (formato carattere).

Valori iniziali

Tabella 738. Campi in MQTMC2

Nome campo	Nome della costante	Valore della costante
TC2SID	TCSIDV	'TMC¬'
TC2VER	TCVER2	'¬¬¬2'
TC2QN	Nessuna	Spazi
TC2PN	Nessuna	Spazi
TC2TD	Nessuna	Spazi
TC2AT	Nessuna	Spazi
TC2AI	Nessuna	Spazi
TC2ED	Nessuna	Spazi
TC2UD	Nessuna	Spazi
TC2QMN	Nessuna	Spazi

Tabella 738. Campi in MQTMC2 (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
Note: 1. Il simbolo ¬ rappresenta un singolo carattere vuoto.		

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQTMC2 Structure
D*
D* Structure identifier
D  TC2SID                 1      4
D* Structure version number
D  TC2VER                 5      8
D* Name of triggered queue
D  TC2QN                  9     56
D* Name of process object
D  TC2PN                 57    104
D* Trigger data
D  TC2TD                105    168
D* Application type
D  TC2AT                169    172
D* Application identifier
D  TC2AI                173    428
D* Environment data
D  TC2ED                429    556
D* User data
D  TC2UD                557    684
D* Queue manager name
D  TC2QMN               685    732
```

IBM i MQWIH (Work information header) su IBM i

La struttura MQWIH descrive le informazioni che devono trovarsi all'inizio di un messaggio che deve essere gestito da z/OS Workload Manager.

Panoramica

Nome formato: FMWIH.

Serie di caratteri e codifica: i campi nella struttura MQWIH si trovano nella serie di caratteri e nella codifica fornita dai campi *MDCSI* e *MDENC* nella struttura dell'intestazione che precede MQWIH o da tali campi nella struttura MQMD se MQWIH si trova all'avvio dei dati del messaggio dell'applicazione.

La serie di caratteri deve avere caratteri a byte singolo per i caratteri validi nei nomi di coda.

Utilizzo: se un messaggio deve essere elaborato da z/OS Workload Manager, il messaggio deve iniziare con una struttura MQWIH.

- "Campi" a pagina 1278
- "Valori iniziali" a pagina 1280
- "Dichiarazione RPG" a pagina 1281

Campi

La struttura MQWIH contiene i seguenti campi; i campi vengono descritti in **ordine alfabetico**:

WICSI (numero intero con segno a 10 cifre)

Identificativo della serie di caratteri dei dati che seguono MQWIH.

Specifica l'identificatore della serie di caratteri dei dati che seguono la struttura MQWIH; non si applica ai dati di caratteri nella stessa struttura MQWIH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. È possibile utilizzare il seguente valore speciale:

CINHT
Eredita l'identificativo della serie di caratteri di questa struttura.

I dati carattere nei dati *che seguono* questa struttura si trovano nella stessa serie di caratteri di questa struttura.

Il gestore code modifica questo valore nella struttura inviata nel messaggio nell'effettivo identificativo della serie di caratteri della struttura. Se non si verifica alcun errore, il valore CSINHT non viene restituito dalla chiamata MQGET.

CSINHT non può essere utilizzato se il valore del campo *MDPAT* in MQMD è ATBRKR.

Il valore iniziale di questo campo è CSUNDF.

WIENC (numero intero con segno a 10 cifre)

Codifica numerica dei dati che seguono MQWIH.

Specifica la codifica numerica dei dati che seguono la struttura MQWIH; non si applica ai dati numerici nella struttura MQWIH stessa.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati.

Il valore iniziale di questo campo è 0.

WIFLG (numero intero con segno a 10 cifre)

Indicatori

Il valore deve essere:

WINONE
Nessun indicatore.

Il valore iniziale di questo campo è WINONE.

WIFMT (stringa di caratteri a 8 byte)

Nome formato dei dati che seguono MQWIH.

Specifica il formato dei dati che seguono la struttura MQWIH.

Nella chiamata MQPUT o MQPUT1 , l'applicazione deve impostare questo campo sul valore appropriato per i dati. Le regole per la codifica di questo campo sono le stesse del campo *MDFMT* in MQMD.

La lunghezza di questo campo è fornita da LNFMT. Il valore iniziale di questo campo è FMNONE.

WILEN (numero intero con segno a 10 cifre)

Lunghezza della struttura MQWIH.

Il valore deve essere:

WILEN1
Lunghezza della struttura dell'intestazione delle informazioni di lavoro version-1 .

La seguente costante specifica la lunghezza della versione corrente:

WILENC
Lunghezza della versione corrente della struttura dell'intestazione delle informazioni di lavoro.

Il valore iniziale di questo campo è WILEN1.

WIRSV (stringa di caratteri a 32 byte)

Riservato.

Questo è un campo riservato; deve essere vuoto.

WISID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

IDISV
Identificativo per la struttura di intestazione delle informazioni di lavoro.

Il valore iniziale di questo campo è WISIDV.

WISNM (stringa di caratteri a 32 byte)

Nome servizio.

È il nome del servizio che deve elaborare il messaggio.

La lunghezza di questo campo è data da LNSVNM. Il valore iniziale di questo campo è di 32 caratteri vuoti.

WISST (stringa di caratteri a 8 byte)

Nome passo del servizio.

Questo è il nome del passo di *WISNM* a cui è correlato il messaggio.

La lunghezza di questo campo è fornita da LNSVST. Il valore iniziale di questo campo è di 8 caratteri vuoti.

WITOK (stringa bit a 16 byte)

Token messaggio.

Questo è un token di messaggio che identifica in modo univoco il messaggio.

Per le chiamate MQPUT e MQPUT1 , questo campo viene ignorato. La lunghezza di questo campo è fornita da LNMTOK. Il valore iniziale di questo campo è MTKNON.

WIVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere:

WIVER1
Struttura dell'intestazione delle informazioni di lavoro Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

WVERC
La versione corrente della struttura dell'intestazione delle informazioni di lavoro.

Il valore iniziale di questo campo è WIVER1.

Valori iniziali

Tabella 739. Campi in MQWIH

Nome campo	Nome della costante	Valore della costante
WISID	IDISV	'WIH¬'
WIVER	WIVER1	1
WILEN	WILEN1	120
WIENC	Nessuna	0
WICSI	SUNDF	0
WIFMT	FMNONE	Spazi

Tabella 739. Campi in MQWIH (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
WIFLG	WINONE	0
WISNM	Nessuna	Spazi
WISST	Nessuna	Spazi
WITOK	NON MTK	Valori null
WIRSV	Nessuna	Spazi

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQWIH Structure
D*
D* Structure identifier
D  WISID                  1      4    INZ('WIH ')
D* Structure version number
D  WIVER                  5      8I 0 INZ(1)
D* Length of MQWIH structure
D  WILEN                  9     12I 0 INZ(120)
D* Numeric encoding of data that followsMQWIH
D  WIENC                 13     16I 0 INZ(0)
D* Character-set identifier of data thatfollows MQWIH
D  WICSI                 17     20I 0 INZ(0)
D* Format name of data that followsMQWIH
D  WIFMT                 21     28    INZ('        ')
D* Flags
D  WIFLG                 29     32I 0 INZ(0)
D* Service name
D  WISNM                 33     64    INZ
D* Service step name
D  WISST                 65     72    INZ
D* Message token
D  WITOK                 73     88    INZ(X'00000000000000-
D                                     000000000000000000')
D* Reserved
D  WIRSV                 89    120    INZ
```

IBM i MQXQH (Transmission - queue header) su IBM i

La struttura MQXQH descrive le informazioni prefissate ai dati dei messaggi dell'applicazione quando si trovano nelle code di trasmissione.

Panoramica

Scopo: una coda di trasmissione è un tipo speciale di coda locale che contiene temporaneamente i messaggi destinati a code remote (ovvero, destinati a code non appartenenti al gestore code locale). Una coda di trasmissione è denotata dall'attributo coda **Usage** con il valore USTRAN.

Nome formato: FMXQH.

Serie di caratteri e codifica: i dati in MQXQH devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da ENNAT per il linguaggio di programmazione C.

La serie di caratteri e la codifica di MQXQH devono essere impostate nei campi *MDCSI* e *MDENC* in:

- MQMD separato (se la struttura MQXQH si trova all'inizio dei dati del messaggio) oppure

- La struttura dell'intestazione che precede la struttura MQXQH (tutti gli altri casi).

Utilizzo: un messaggio che si trova su una coda di trasmissione ha *due* descrittori di messaggi:

- Un descrittore di messaggi viene archiviato separatamente dai dati del messaggio; viene denominato *descrittore di messaggi separato*e viene generato dal gestore code quando il messaggio viene inserito nella coda di trasmissione. Alcuni dei campi nel descrittore del messaggio separato vengono copiati dal descrittore del messaggio fornito dall'applicazione nella chiamata MQPUT o MQPUT1 .

  Il descrittore del messaggio separato è quello restituito all'applicazione nel parametro **MSGDSC** della chiamata MQGET quando il messaggio viene rimosso dalla coda di trasmissione.
- Un secondo descrittore del messaggio viene memorizzato all'interno della struttura MQXQH come parte dei dati del messaggio; viene denominato *descrittore del messaggio integrato*ed è una copia del descrittore del messaggio fornito dall'applicazione nella chiamata MQPUT o MQPUT1 (con variazioni minori).

  Il descrittore del messaggio incorporato è sempre un MQMD version-1 . Se il messaggio inserito dall'applicazione ha valori non predefiniti per uno o più campi version-2 in MQMD, una struttura MQMDE segue MQXQH ed è a sua volta seguita dai dati del messaggio dell'applicazione (se presenti). MQMDE è:

  – Generato dal gestore code (se l'applicazione utilizza un MQMD version-2 per inserire il messaggio) oppure
  – Già presente all'inizio dei dati del messaggio dell'applicazione (se l'applicazione utilizza un MQMD version-1 per inserire il messaggio).

  Il descrittore del messaggio incorporato è quello restituito all'applicazione nel parametro **MSGDSC** della chiamata MQGET quando il messaggio viene rimosso dalla coda di destinazione finale.

- "Campi nel descrittore del messaggio separato" a pagina 1282
- "Campi nel descrittore del messaggio incorporato" a pagina 1283
- "Inserimento di messaggi nelle code remote" a pagina 1284
- "Inserimento diretto dei messaggi nelle code di trasmissione" a pagina 1284
- "Richiamo dei messaggi dalle code di trasmissione" a pagina 1284
- "Campi" a pagina 1285
- "Valori iniziali" a pagina 1286
- "Dichiarazione RPG" a pagina 1286

Campi nel descrittore del messaggio separato

I campi nel descrittore del messaggio separato vengono impostati dal gestore code come mostrato nel seguente elenco. Se il gestore code non supporta l'MQMD version-2 , viene utilizzato un MQMD version-1 senza perdita di funzione.

Tabella 740. Campi nel descrittore del messaggio separato e valori utilizzati

Campo in MQMD separato	Valore utilizzato
MDSID	MSIDV
MDVER	MDVER2
MDREP	Copiato dal descrittore del messaggio incorporato, ma con i bit identificati da ROAUXM impostati su zero. Ciò impedisce che un messaggio di report COA o COD venga generato quando un messaggio viene inserito o rimosso da una coda di trasmissione.
MDMT	Copiato dal descrittore del messaggio incorporato.
MDEXP	Copiato dal descrittore del messaggio incorporato.
MDFB	Copiato dal descrittore del messaggio incorporato.

Tabella 740. Campi nel descrittore del messaggio separato e valori utilizzati (Continua)

Campo in MQMD separato	Valore utilizzato
MDENC	ENNAT
MDCSI	Attributo **CodedCharSetId** del gestore code.
MDFMT	FMXQH
MDPRI	Copiato dal descrittore del messaggio incorporato.
MDPER	Copiato dal descrittore del messaggio incorporato.
MDMID	Il gestore code genera un nuovo valore. Questo identificativo del messaggio è diverso da *MDMID* che il gestore code potrebbe aver generato per il descrittore del messaggio integrato (vedere descritto in precedenza).
MDCID	Il *MDMID* dal descrittore del messaggio incorporato.
MDBOC	0
MDRQ	Copiato dal descrittore del messaggio incorporato.
MDRM	Copiato dal descrittore del messaggio incorporato.
MDUID	Copiato dal descrittore del messaggio incorporato.
MDACC	Copiato dal descrittore del messaggio incorporato.
MDAID	Copiato dal descrittore del messaggio incorporato.
MDPAT	ATQM
MDPAN	Primi 28 byte del nome del gestore code.
MDPD	Data in cui il messaggio è stato inserito nella coda di trasmissione.
MDPT	L'ora in cui il messaggio è stato inserito nella coda di trasmissione.
MDAOD	Spazi
MDGID	GINONE
MDSEQ	1
MDOFF	0
MDMFL	MFNONE
MDOLN	OLUNDF

Campi nel descrittore del messaggio incorporato

I campi nel descrittore del messaggio integrato hanno gli stessi valori di quelli del parametro **MSGDSC** della chiamata MQPUT o MQPUT1 , tranne per quanto segue:

- Il campo *MDVER* ha sempre il valore MDVER1.
- Se il campo *MDPRI* ha il valore PRQDEF, viene sostituito dal valore dell'attributo **DefPriority** della coda.
- Se il campo *MDPER* ha il valore PEQDEF, viene sostituito dal valore dell'attributo **DefPersistence** della coda.
- Se il campo *MDMID* ha il valore MINONE, o se è stata specificata l'opzione PMNMID, o se il messaggio è un messaggio dell'elenco di distribuzione, *MDMID* viene sostituito da un nuovo identificativo del messaggio generato dal gestore code.

  Quando un messaggio dell'elenco di distribuzione viene suddiviso in messaggi dell'elenco di distribuzione più piccoli posizionati su code di trasmissione differenti, il campo *MDMID* in ciascuno dei

nuovi descrittori di messaggi incorporati è uguale a quello del messaggio dell'elenco di distribuzione originale.

- Se è stata specificata l'opzione PMNCID, *MDCID* viene sostituito da un nuovo identificativo di correlazione generato dal gestore code.
- I campi di contesto sono impostati come indicato dalle opzioni PM* specificate nel parametro **PMO** ; i campi di contesto sono:
  - *MDACC*
  - *MDAID*
  - *MDAOD*
  - *MDPAN*
  - *MDPAT*
  - *MDPD*
  - *MDPT*
  - *MDUID*
- I campi version-2 (se presenti) vengono rimossi da MQMD e spostati in una struttura MQMDE, se uno o più campi version-2 hanno un valore non predefinito.

Inserimento di messaggi nelle code remote

: quando un'applicazione inserisce un messaggio su una coda remota (specificando direttamente il nome della coda remota o utilizzando una definizione locale della coda remota), il gestore code locale:

- Crea una struttura MQXQH contenente il descrittore del messaggio incorporato
- Accoda un MQMDE se ne è necessario uno e non è già presente
- Accoda i dati del messaggio dell'applicazione
- Inserisce il messaggio in una coda di trasmissione appropriata

Inserimento diretto dei messaggi nelle code di trasmissione

È inoltre possibile per un'applicazione inserire un messaggio direttamente in una coda di trasmissione. In questo caso, l'applicazione deve anteporre ai dati del messaggio dell'applicazione una struttura MQXQH e inizializzare i campi con i valori appropriati. Inoltre, il campo *MDFMT* nel parametro **MSGDSC** della chiamata MQPUT o MQPUT1 deve avere il valore FMXQH.

I dati carattere nella struttura MQXQH creati dall'applicazione devono trovarsi nella serie di caratteri del gestore code locale (definito dall'attributo del gestore code **CodedCharSetId**) e i dati interi devono essere nella codifica della macchina nativa. Inoltre, i dati carattere nella struttura MQXQH devono essere riempiti con spazi vuoti fino alla lunghezza definita del campo; i dati non devono essere terminati in modo prematuro utilizzando un carattere null, poiché il gestore code non converte i caratteri null e successivi in spazi vuoti nella struttura MQXQH.

Tenere presente, tuttavia, che il gestore code non controlla la presenza di una struttura MQXQH o che sono stati specificati valori validi per i campi.

Richiamo dei messaggi dalle code di trasmissione

Le applicazioni che ricevono messaggi da una coda di trasmissione devono elaborare le informazioni nella struttura MQXQH in modo appropriato. La presenza della struttura di MQXQH all'inizio dei dati del messaggio dell'applicazione è indicata dal valore FMXQH restituito nel campo *MDFMT* nel parametro **MSGDSC** della chiamata MQGET. I valori restituiti nei campi *MDCSI* e *MDENC* nel parametro **MSGDSC** , indicano la serie di caratteri e la codifica dei dati carattere e numero intero nella struttura MQXQH. La serie di caratteri e la codifica dei dati del messaggio dell'applicazione sono definiti dai campi *MDCSI* e *MDENC* nel descrittore del messaggio incorporato.

Campi

La struttura MQXQH contiene i seguenti campi; i campi sono descritti in **ordine alfabetico**:

XQMD (MQMD1)

Descrittore messaggio originale.

Questo è il descrittore del messaggio incorporato ed è una copia di chiusura del descrittore del messaggio MQMD specificato come parametro **MSGDSC** nella chiamata MQPUT o MQPUT1 quando il messaggio è stato originariamente inserito sulla coda remota.

Nota: Questo è un MQMD version-1 .

I valori iniziali dei campi in questa struttura sono gli stessi della struttura MQMD.

XQRQ (stringa di caratteri a 48 byte)

Il nome della coda di destinazione.

Questo è il nome della coda messaggi che è la destinazione finale apparente per il messaggio (potrebbe non essere la destinazione finale effettiva se, ad esempio, questa coda è definita in *XQRQM* per essere una definizione locale di un'altra coda remota).

Se il messaggio è un messaggio dell'elenco di distribuzione (ovvero, il campo *MDFMT* nel descrittore del messaggio incorporato è FMDH), *XQRQ* è vuoto.

La lunghezza di questo campo è fornita da LNQN. Il valore iniziale di questo campo è di 48 caratteri vuoti.

XQRQM (stringa di caratteri a 48 byte)

Nome del gestore code di destinazione.

Questo è il nome del gestore code o del gruppo di condivisione code che possiede la coda che è la destinazione finale apparente per il messaggio.

Se il messaggio è un messaggio dell'elenco di distribuzione, *XQRQM* è vuoto.

La lunghezza di questo campo è fornita da LNQMN. Il valore iniziale di questo campo è di 48 caratteri vuoti.

XQSID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

XQSIDV
Identificativo per la struttura dell'intestazione della coda di trasmissione.

Il valore iniziale di questo campo è XQSIDV.

XQVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere:

XQVER1
Numero di versione per la struttura di intestazione della coda di trasmissione.

La seguente costante specifica il numero di versione della versione corrente:

XQVERC
Versione corrente della struttura dell'intestazione della coda di trasmissione.

Il valore iniziale di questo campo è XQVER1.

Valori iniziali

Tabella 741. Campi in MQXQH		
Nome campo	**Nome della costante**	**Valore della costante**
XQSID	XQSIDV	'XQH¬'
XQVER	XQVER1	1
XQRQ	Nessuna	Spazi
XQRQM	Nessuna	Spazi
XQMD	Stessi nomi e valori di MQMD; consultare Tabella 709 a pagina 1184	-
Note: 1. Il simbolo ¬ rappresenta un singolo carattere vuoto.		

Dichiarazione RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQXQH Structure
D*
D* Structure identifier
D  XQSID                  1      4    INZ('XQH ')
D* Structure version number
D  XQVER                  5      8I 0 INZ(1)
D* Name of destination queue
D  XQRQ                   9     56    INZ
D* Name of destination queue manager
D  XQRQM                 57    104    INZ
D* Original message descriptor
D  XQ1SID               105    108    INZ('MD  ')
D  XQ1VER               109    112I 0 INZ(1)
D  XQ1REP               113    116I 0 INZ(0)
D  XQ1MT                117    120I 0 INZ(8)
D  XQ1EXP               121    124I 0 INZ(-1)
D  XQ1FB                125    128I 0 INZ(0)
D  XQ1ENC               129    132I 0 INZ(273)
D  XQ1CSI               133    136I 0 INZ(0)
D  XQ1FMT               137    144    INZ('        ')
D  XQ1PRI               145    148I 0 INZ(-1)
D  XQ1PER               149    152I 0 INZ(2)
D  XQ1MID               153    176    INZ(X'00000000000000-
D                                     00000000000000000000-
D                                     000000000000')
D  XQ1CID               177    200    INZ(X'00000000000000-
D                                     00000000000000000000-
D                                     000000000000')
D  XQ1BOC               201    204I 0 INZ(0)
D  XQ1RQ                205    252    INZ
D  XQ1RM                253    300    INZ
D  XQ1UID               301    312    INZ
D  XQ1ACC               313    344    INZ(X'00000000000000-
D                                     00000000000000000000-
D                                     00000000000000000000-
D                                     000000')
D  XQ1AID               345    376    INZ
D  XQ1PAT               377    380I 0 INZ(0)
D  XQ1PAN               381    408    INZ
D  XQ1PD                409    416    INZ
D  XQ1PT                417    424    INZ
D  XQ1AOD               425    428    INZ
```

IBM i Chiamate di funzioni su IBM i

Utilizzare queste informazioni per informazioni sulle chiamate di funzione disponibili nella programmazione IBM i .

Convenzioni utilizzate nelle descrizioni delle chiamate su IBM i

Per ogni chiamata, questa raccolta di argomenti fornisce una descrizione dei parametri e dell'utilizzo della chiamata. Ciò è seguito da richiami tipici della chiamata e dichiarazioni tipiche dei parametri, nel linguaggio di programmazione RPG.

Importante: Durante la codifica delle chiamate API IBM MQ è necessario assicurarsi che siano forniti tutti i parametri rilevanti (come descritto nelle seguenti sezioni). In caso contrario, si potrebbero ottenere risultati imprevedibili.

La descrizione di ogni chiamata contiene le sezioni seguenti:

Nome chiamata
Il nome della chiamata, seguito da una breve descrizione dello scopo della chiamata.

Parametri
Per ogni parametro, il nome è seguito dal relativo tipo di dati tra parentesi () e la sua direzione; ad esempio:

CMPCOD (numero intero decimale a 9 cifre) - output

Sono disponibili ulteriori informazioni sui tipi di dati della struttura in "Tipi di dati elementari" a pagina 1024.

La direzione del parametro può essere:

Immissione
Il programmatore deve fornire questo parametro.

Output
La chiamata restituisce questo parametro.

Input/output
È necessario fornire questo parametro, ma viene modificato dalla chiamata.

C'è anche una breve descrizione dello scopo del parametro, insieme a un elenco di tutti i valori che il parametro può prendere.

Gli ultimi due parametri in ogni chiamata sono un codice di completamento e un codice motivo. Il codice di completamento indica se la chiamata è stata completata correttamente, parzialmente o per nulla. Ulteriori informazioni sull'esito positivo parziale o sull'esito negativo della chiamata sono fornite nel codice di errore.

Note d'utilizzo
Ulteriori informazioni sulla chiamata, che descrivono come utilizzarla e le eventuali limitazioni al suo utilizzo.

Richiamo RPG
Richiamo tipico della chiamata e dichiarazione dei suoi parametri in RPG.

Altre convenzioni notazionali sono:

Costanti
I nomi delle costanti vengono visualizzati in maiuscolo; ad esempio, OOOUT.

Array
In alcune chiamate, i parametri sono schiere di stringhe di caratteri con una dimensione non fissa. Nelle descrizioni di questi parametri, una *n* minuscola rappresenta una costante numerica. Quando si codifica la dichiarazione per tale parametro, sostituire *n* con il valore numerico richiesto.

IBM i MQBACK (modifiche di backout) su IBM i

La chiamata MQBACK indica al gestore code che è necessario eseguire il backout di tutti i messaggi e gli inserimenti che si sono verificati dall'ultimo punto di sincronizzazione. I messaggi inseriti come parte di un'unità di lavoro vengono eliminati; i messaggi richiamati come parte di un'unità di lavoro vengono reintegrati nella coda.

- Questa chiamata è supportata nei seguenti ambienti:
  -  AIX
  -  IBM i
  -  Windows
- “Sintassi” a pagina 1288
- “Note d'utilizzo” a pagina 1288
- “Parametri” a pagina 1289
- “Dichiarazione RPG” a pagina 1290

Sintassi

MQBACK *(Hconn, CompCode, Reason)*

Note d'utilizzo

Considerare queste note di utilizzo quando si utilizza MQBACK.

1. Questa chiamata può essere utilizzata solo quando il gestore code stesso coordina l'unità di lavoro. Si tratta di un'unità di lavoro locale, in cui le modifiche riguardano solo le risorse IBM MQ .
2. In ambienti in cui il gestore code non coordina l'unità di lavoro, è necessario utilizzare la chiamata di backout appropriata invece di MQBACK. L'ambiente può anche supportare un backout implicito causato dalla chiusura anomala dell'applicazione.
   - Su IBM i, questa chiamata può essere utilizzata per unità di lavoro locali coordinate dal gestore code. Ciò significa che una definizione di commit non deve esistere a livello di lavoro, ovvero il comando STRCMTCTL con il parametro **CMTSCOPE(*JOB)** non deve essere stato immesso per il lavoro.
3. Se un'applicazione termina con modifiche non sottoposte a commit in un'unità di lavoro, la disposizione di tali modifiche dipende dal fatto che l'applicazione termini normalmente o in modo anomalo. Consultare le note sull'utilizzo in “MQDISC (Disconnetti gestore code) su IBM i” a pagina 1327 per ulteriori dettagli.
4. Quando un'applicazione inserisce o richiama i messaggi in gruppi o segmenti di messaggi logici, il gestore code conserva le informazioni relative al gruppo di messaggi e al messaggio logico per le ultime chiamate MQPUT e MQGET riuscite. Queste informazioni sono associate all'handle della coda e includono:
   - I valore dei campi *MDGID*, *MDSEQ*, *MDOFF*e *MDMFL* in MQMD.
   - Se il messaggio fa parte di un'unità di lavoro.
   - Per la chiamata MQPUT: se il messaggio è persistente o non persistente.

   Il gestore code conserva *tre* serie di informazioni sui gruppi e sui segmenti, una serie per ciascuno dei seguenti:
   - L'ultima chiamata MQPUT riuscita (può far parte di un'unità di lavoro).
   - L'ultima chiamata MQGET riuscita che ha rimosso un messaggio dalla coda (può far parte di un'unità di lavoro).
   - L'ultima chiamata MQGET riuscita che ha visualizzato un messaggio sulla coda (non può far parte di un'unità di lavoro).

   Se l'applicazione inserisce o richiama i messaggi come parte di un'unità di lavoro e l'applicazione decide quindi di eseguire il backout dell'unità di lavoro, le informazioni sul gruppo e sul segmento vengono ripristinate sul valore che aveva in precedenza:
   - Le informazioni associate alla chiamata MQPUT vengono ripristinate al valore che aveva prima della prima chiamata MQPUT riuscita per tale handle di coda nell'unità di lavoro corrente.

- Le informazioni associate alla chiamata MQGET vengono ripristinate al valore che aveva prima della prima chiamata MQGET riuscita per tale handle di coda nell'unità di lavoro corrente.

Le code che sono state aggiornate dall'applicazione dopo l'avvio dell'unità di lavoro, ma al di fuori dell'ambito dell'unità di lavoro, non hanno le relative informazioni sul gruppo e sul segmento ripristinate se viene eseguito il backout dell'unità di lavoro.

Il ripristino delle informazioni sul gruppo e sul segmento al suo valore precedente quando viene eseguito il backout di un'unità di lavoro consente all'applicazione di distribuire un gruppo di messaggi di grandi dimensioni o un messaggio logico di grandi dimensioni costituito da molti segmenti in diverse unità di lavoro e di riavviare nel punto corretto nel gruppo di messaggi o nel messaggio logico in caso di errore di una delle unità di lavoro. L'utilizzo di diverse unità di lavoro potrebbe essere vantaggioso se il gestore code locale ha solo una memoria di coda limitata. Tuttavia, l'applicazione deve conservare informazioni sufficienti per essere in grado di riavviare l'inserimento o il richiamo dei messaggi nel punto corretto se si verifica un errore di sistema. Per dettagli su come riavviare il sistema nel punto corretto dopo un malfunzionamento del sistema, consultare l'opzione PMLOGO descritta in “MQPMO (Put - message options) su IBM i” a pagina 1206e l'opzione GMLOGO descritta in “MQGMO (opzioni Get - message) su IBM i” a pagina 1105.

Le note di uso rimanenti si applicano solo quando il gestore code coordina le unità di lavoro:

1. Un'unità di lavoro ha lo stesso ambito di un handle di connessione. Ciò significa che tutte le chiamate IBM MQ che interessano una particolare unità di lavoro devono essere eseguite utilizzando lo stesso handle di connessione. Le chiamate emesse utilizzando un handle di collegamento differente (ad esempio, le chiamate emesse da un'altra applicazione) influenzano un'unità di lavoro diversa. Per informazioni sull'ambito degli handle di connessione, consultare il parametro **HCONN** descritto in “MQCONN (gestore code di connessione) su IBM i” a pagina 1313 .
2. Solo i messaggi inseriti o richiamati come parte dell'unità di lavoro corrente vengono influenzati da questa chiamata.
3. Un'applicazione di lunga durata che emette chiamate MQGET, MQPUT o MQPUT1 all'interno di un'unità di lavoro, ma che non emette mai una chiamata di commit o di backout, può causare il riempimento delle code con messaggi non disponibili per altre applicazioni. Per evitare questa possibilità, l'amministratore deve impostare l'attributo del gestore code **MaxUncommittedMsgs** su un valore sufficientemente basso per evitare che le applicazioni runaway riempiano le code, ma sufficientemente alto per consentire il corretto funzionamento delle applicazioni di messaggistica previste.

Parametri

La chiamata MQBACK ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Handle di connessione.

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

CMPCOD (numero intero con segno a 10 cifre) - output

Codice di completamento.

Il valore è uno dei seguenti:

CCOK
Completamento con esito positivo.

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Codice di errore *COMCOD*.

Se *COMCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *COMCOD* è CCFAIL:

RC2219
(2219, X'8AB') Chiamata MQI reimmessa prima del completamento della chiamata precedente.

RC2009
(2009, X'7D9') Connessione al gestore code persa.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2101
(2101, X'835 ') Oggetto danneggiato.

RC2123
(2123, X'84B') Il risultato dell'operazione di commit o di backout è misto.

RC2162
(2162, X'872 ') Chiusura del gestore code.

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                   CALLP     MQBACK(HCONN : COMCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQBACK           PR                  EXTPROC('MQBACK')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Completion code
D COMCOD                        10I 0
D* Reason code qualifying COMCOD
D REASON                        10I 0
```

IBM i MQBEGIN (Inizio unità di lavoro) su IBM i

La chiamata MQBEGIN inizia con un'unità di lavoro coordinata dal gestore code e che può coinvolgere gestori risorse esterni.

- Questa chiamata è supportata nei seguenti ambienti:
  - AIX AIX
  - IBM i IBM i
  - Windows Windows
- “Sintassi” a pagina 1291
- “Note d'utilizzo” a pagina 1291
- “Parametri” a pagina 1292

- “Dichiarazione RPG” a pagina 1293

Sintassi

MQBEGIN *(HCONN, BEGOP, CMPCOD, REASON)*

Note d'utilizzo

1. La chiamata MQBEGIN può essere utilizzata per avviare un'unità di lavoro coordinata dal gestore code e che potrebbe comportare modifiche alle risorse di proprietà di altri gestori risorse. Il gestore code supporta tre tipi di unità di lavoro:

   Unità di lavoro locale coordinata dal gestore code
   Questa è un'unità di lavoro in cui il gestore code è l'unico gestore risorse partecipante e quindi il gestore code agisce come coordinatore dell'unità di lavoro.

   - Per avviare questo tipo di unità di lavoro, l'opzione PMSYP o GMSYP deve essere specificata sulla prima chiamata MQPUT, MQPUT1o MQGET nell'unità di lavoro.

     Non è necessario che l'applicazione emani la chiamata MQBEGIN per avviare l'unità di lavoro, ma se viene utilizzato MQBEGIN, la chiamata viene completata con CCWARN e codice motivo RC2121.
   - Per eseguire il commit o il backout di questo tipo di unità di lavoro, è necessario utilizzare la chiamata MQCMIT o MQBACK.

   Unità di lavoro globale coordinata dal gestore code
   Si tratta di un'unità di lavoro in cui il gestore code funge da coordinatore dell'unità di lavoro, sia per le IBM MQ risorse *che per* per le risorse appartenenti ad altri gestori risorse. Questi gestori risorse cooperano con il gestore code per garantire che tutte le modifiche alle risorse nell'unità di lavoro siano sottoposte a commit o a backout insieme.

   - Per avviare questo tipo di unità di lavoro, è necessario utilizzare la chiamata MQBEGIN.
   - Per eseguire il commit o il backout di questo tipo di unità di lavoro, è necessario utilizzare le chiamate MQCMIT e MQBACK.

   Unità di lavoro globale coordinata esternamente
   Questa è un'unità di lavoro in cui il gestore code è un partecipante, ma il gestore code non agisce come coordinatore dell'unità di lavoro. Esiste invece un coordinatore dell'unità di lavoro esterna con cui il gestore code collabora.

   - Per avviare questo tipo di unità di lavoro, deve essere utilizzata la chiamata pertinente fornita dal coordinatore esterno dell'unità di lavoro.

     Se la chiamata MQBEGIN viene utilizzata per tentare di avviare l'unità di lavoro, la chiamata ha esito negativo con il codice di errore RC2012.
   - Per eseguire il commit o il backout di questo tipo di unità di lavoro, è necessario utilizzare le chiamate di commit e backout fornite dal coordinatore dell'unità di lavoro esterno.

     Se si utilizza la chiamata MQCMIT o MQBACK per provare a eseguire il commit o il backout dell'unità di lavoro, la chiamata ha esito negativo con codice motivo RC2012.
2. Se l'applicazione termina con modifiche non sottoposte a commit in un'unità di lavoro, la disposizione di tali modifiche dipende dal fatto che l'applicazione termini normalmente o in modo anomalo. Consultare le note sull'utilizzo in “MQDISC (Disconnetti gestore code) su IBM i” a pagina 1327 per ulteriori dettagli.
3. Un'applicazione può partecipare a una sola unità di lavoro alla volta. La chiamata MQBEGIN ha esito negativo con codice di errore RC2128 se esiste già un'unità di lavoro per l'applicazione, indipendentemente dal tipo di unità di lavoro.
4. La chiamata MQBEGIN non è valida in un ambiente client IBM MQ . Un tentativo di utilizzare la chiamata ha esito negativo con codice di errore RC2012.

5. Quando il gestore code agisce come coordinatore dell'unità di lavoro per le unità di lavoro globali, i gestori risorse che possono partecipare all'unità di lavoro vengono definiti nel file di configurazione del gestore code.
6. Su IBM i, i tre tipi di unità di lavoro sono supportati come segue:
   - **Le unità di lavoro locali coordinate dal gestore code** possono essere utilizzate solo quando non esiste una definizione di commit a livello di lavoro, ossia, il comando STRCMTCTL con il parametro **CMTSCOPE(*JOB)** non deve essere stato immesso per il lavoro.
   - Le **Unità di lavoro globali coordinate dal gestore code** non sono supportate.
   - **Le unità di lavoro globali coordinate esternamente** possono essere utilizzate solo quando esiste una definizione di commit a livello di lavoro, vale a dire, il comando STRCMTCTL con il parametro **CMTSCOPE(*JOB)** deve essere stato emesso per il lavoro. Se questa operazione è stata eseguita, le operazioni di IBM i COMMIT e ROLLBACK si applicano alle risorse IBM MQ e alle risorse appartenenti ad altri gestori risorse partecipanti.

Parametri

La chiamata MQBEGIN ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Handle di connessione.

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

BEGOP (MQBO) - input/output

Opzioni che controllano l'azione di MQBEGIN.

Vedi “MQBO (Opzioni di inizio) su IBM i” a pagina 1045 per i dettagli.

Se non è richiesta alcuna opzione, i programmi scritti nell'assembler C o S/390 possono specificare un indirizzo di parametro null, invece di specificare l'indirizzo di una struttura MQBO.

CMPCOD (numero intero con segno a 10 cifre) - output

Codice di completamento.

Il valore è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Codice di errore *CMPCOD*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCWARN:

RC2121
(2121, X'849 ') Nessun gestore risorse partecipante registrato.

RC2122
(2122, X'84A') Il gestore risorse partecipante non è disponibile.

Se *CMPCOD* è CCFAIL:

RC2134
(2134, X'856 ') Struttura opzioni di inizio non valida.

RC2219
(2219, X'8AB') Chiamata MQI reimmessa prima del completamento della chiamata precedente.

RC2009
(2009, X'7D9') Connessione al gestore code persa.

RC2012
(2012, X'7DC') Chiamata non valida nell'ambiente.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2046
(2046, X'7FE') Opzioni non valide o non congruenti.

RC2162
(2162, X'872 ') Chiusura del gestore code.

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

RC2128
(2128, X'850 ') Unità di lavoro già avviata.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                    CALLP     MQBEGIN(HCONN : BEGOP : CMPCOD :
C                                       REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQBEGIN          PR                  EXTPROC('MQBEGIN')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Options that control the action of MQBEGIN
D BEGOP                         12A
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CMPCOD
D REASON                        10I 0
```

IBM i MQBUFMH (Converti buffer in handle del messaggio) su IBM i

La chiamata alla funzione MQBUFMH converte un buffer in un handle del messaggio ed è l'inverso della chiamata MQMHBUF.

Questa chiamata acquisisce un descrittore di messaggi e le proprietà MQRFH2 nel buffer e li rende disponibili tramite un handle del messaggio. Le proprietà MQRFH2 nei dati del messaggio vengono, facoltativamente, rimosse. I campi *Encoding*, *CodedCharSetId*e *Format* del descrittore del messaggio vengono aggiornati, se necessario, per descrivere correttamente il contenuto del buffer dopo la rimozione delle proprietà.

- "Sintassi" a pagina 1294
- "Note d'utilizzo" a pagina 1294
- "Parametri" a pagina 1294

- “Dichiarazione RPG” a pagina 1296

Sintassi

MQBUFMH *(Hconn, Hmsg, BufMsgHOpts, MsgDesc, Buffer, BufferLength, DataLength, CompCode, Reason)*

Note d'utilizzo

Le chiamate MQBUFMH non possono essere intercettate dalle uscite API - un buffer viene convertito in un handle del messaggio nello spazio dell'applicazione; la chiamata non raggiunge il gestore code.

Parametri

La chiamata MQBUFMH ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* deve corrispondere all'handle di connessione utilizzato per creare l'handle del messaggio specificato nel parametro **Hmsg** .

Se l'handle del messaggio è stato creato utilizzando HCUNAS, è necessario stabilire una connessione valida sul thread convertendo un buffer in un handle del messaggio. Se non viene stabilita una connessione valida, la chiamata non riesce con RC2009.

HMSG (numero intero con segno a 20 cifre) - input

Questo handle è l'handle del messaggio per cui è richiesto un buffer. Il valore è stato restituito da una precedente chiamata MQCRTMH.

BMHOPT (MQBMHO) - input

La struttura MQBMHO consente alle applicazioni di specificare le opzioni che controllano il modo in cui i gestori dei messaggi vengono prodotti dai buffer.

Vedi “MQBMHO (Buffer per le opzioni di gestione dei messaggi) su IBM i” a pagina 1044 per i dettagli.

MSGDSC (MQMD) - input/output

La struttura *MSGDSC* contiene le proprietà descrittore del messaggio e descrive il contenuto dell'area di buffer.

In fase di output dalla chiamata, le proprietà vengono facoltativamente rimosse dall'area del buffer e, in questo caso, il descrittore del messaggio viene aggiornato per descrivere correttamente l'area del buffer.

I dati in questa struttura devono essere nella serie di caratteri e nella codifica dell'applicazione.

BUFLEN (numero intero con segno a 10 cifre) - input

BUFLEN è la lunghezza dell'area Buffer, in byte.

Un *BUFLEN* di zero byte è valido e indica che l'area buffer non contiene dati.

BUFFER (stringa di bit a 1 byte x BUFLEN) - input/output

BUFFER definisce l'area contenente il buffer di messaggi. Per la maggior parte dei dati, è necessario allineare il buffer su un limite di 4 byte.

Se *BUFFER* contiene dati numerici o di caratteri, impostare i campi *CodedCharSetId* e *Encoding* nel parametro **MSGDSC** sui valori appropriati per i dati; ciò consente la conversione dei dati, se necessario.

Se le proprietà vengono trovate nel buffer del messaggio, vengono facoltativamente rimosse; in seguito diventano disponibili dall'handle del messaggio al ritorno dalla chiamata.

Nel linguaggio di programmazione C, il parametro viene dichiarato come un puntatore a void, il che significa che l'indirizzo di qualsiasi tipo di dati può essere specificato come parametro.

Se il parametro **BUFLEN** è zero, non si fa riferimento a *BUFFER* . In questo caso, l'indirizzo del parametro inoltrato dai programmi scritti nell'assembler C o System/390 può essere null.

DATLEN (numero intero con segno a 10 cifre) - emissione

DATLEN è la lunghezza, in byte, del buffer per cui potrebbero essere rimosse le proprietà.

CMPCOD (numero intero con segno a 10 cifre) - output

CCOK
Completamento con esito positivo.

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Il codice di errore che qualifica *CMPCOD*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCFAIL:

RC2204
(2204, X'089C') Adattatore non disponibile.

RC2130
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

RC2157
(2157, X'86D') Gli ASID principale e home differiscono.

RC2489
(2489, X'09B9') Buffer per la struttura delle opzioni di gestione messaggi non valida.

RC2004
Parametro buffer (2004, X'07D4') non valido.

RC2005
Parametro di lunghezza buffer (2005, X'07D5') non valido.

RC2219
(2219, X'08AB') Chiamata MQI immessa prima del completamento della chiamata precedente.

RC2009
(2009, X'07D9') Connessione al gestore code persa.

RC2460
(2460, X'099C') Gestione messaggio non valida.

RC2026
(2026, X'07EA') Descrittore messaggio non valido.

RC2499
(2499, X'09C3') handle del messaggio già in uso.

RC2046
(2046, X'07FE') Opzioni non valide o non congruenti.

RC2334
(2334, X'091E') Struttura MQRFH2 non valida.

RC2421
(2421, X'0975 ') Impossibile analizzare una cartella MQRFH2 contenente le proprietà.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQBUFMH(HCONN : HMSG : BMHOPT :
                                        MSGDSC : BUFLEN : BUFFER :
                                        DATLEN : CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
DMQBUFMH          PR                  EXTPROC('MQBUFMH')
D* Connection handle
D HCONN                         10I 0
D* Message handle
D HMSG                          10I 0
D* Options that control the action of MQBUFMH
D BMHOPT                        12A   VALUE
D* Message descriptor
D MSGDSC                       364A
D* Length in bytes of the Buffer area
D BUFLEN                        10I 0
D* Area to contain the message buffer
D BUFFER                          *   VALUE
D* Length of the output buffer
D DATLEN                        10I 0
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CompCode
D REASON                        10I 0
```

IBM i MQCB (Manage callback) su IBM i

La chiamata MQCB registra nuovamente un callback per l'handle dell'oggetto specificato e controlla l'attivazione e le modifiche al callback.

Un callback è una parte di codice (specificato come il nome di una funzione che può essere collegata dinamicamente o come un puntatore di funzione) che viene richiamato da IBM MQ quando si verificano determinati eventi.

Per utilizzare MQCB e MQCTL su un client V7 , è necessario essere connessi ad un server V7 e il parametro **SHARECNV** del canale deve avere un valore diverso da zero.

I tipi di callback che possono essere definiti sono:

consumatore di messaggi
Una funzione di callback del destinatario del messaggio viene chiamata quando un messaggio, che soddisfa i criteri di selezione specificati, è disponibile su un handle dell'oggetto.

È possibile registrare una sola funzione di callback per ciascun handle di oggetto. Se una singola coda deve essere letta con più criteri di selezione, la coda deve essere aperta più volte e una funzione consumer deve essere registrata su ciascun handle.

Gestore eventi
Il gestore eventi viene richiamato per condizioni che influenzano l'intero ambiente di callback.

La funzione viene richiamata quando si verifica una condizione di evento, ad esempio, un gestore code o una connessione in fase di arresto o di sospensione.

La funzione non viene richiamata per condizioni specifiche di un singolo consumatore di messaggi, ad esempio RC2016; viene richiamata tuttavia se una funzione di callback non termina normalmente.

- “Sintassi” a pagina 1297
- “Note di utilizzo per MQCB” a pagina 1297
- “Parametri per MQCB” a pagina 1298
- “Dichiarazione RPG” a pagina 1304

Sintassi

(HCONN, OPERATN, HOBJ, CBDSC, MSGDSC, GMO, CMPCOD, REASON) MQCB

Note di utilizzo per MQCB

1. MQCB viene utilizzato per definire l'azione da richiamare per ciascun messaggio, corrispondente ai criteri specificati, disponibile sulla coda. Quando l'azione viene elaborata, il messaggio viene rimosso dalla coda e inoltrato al destinatario del messaggio definito oppure viene fornito un token del messaggio, che viene utilizzato per richiamare il messaggio.
2. MQCB può essere utilizzato per definire le routine di callback prima di iniziare l'utilizzo con MQCTL oppure può essere utilizzato dall'interno di una routine di callback.
3. Per utilizzare MQCB dall'esterno di una routine di callback, è necessario prima sospendere il consumo dei messaggi utilizzando MQCTL e riprendere il consumo in seguito.

Sequenza di callback del destinatario del messaggio

È possibile configurare un consumer per richiamare il callback in punti chiave durante il ciclo di vita del consumer. Ad esempio:

- quando il consumatore è registrato per la prima volta,
- quando viene avviata la connessione,
- quando la connessione viene arrestata e
- quando la registrazione del consumer viene annullata, esplicitamente o implicitamente da un MQCLOSE.

Tabella 742. Definizioni di verbo MQCTL

Verbo	**Significato**
MQCTL (START)	Chiamata MQCTL utilizzando l'operazione CTLSR
MQCTL (STOP)	Chiamata MQCTL utilizzando l'operazione CTLSP
MQCTL (ATTESA)	Chiamata MQCTL utilizzando l'operazione CTLSW

Consente al consumatore di mantenere lo stato associato al consumatore. Quando un'applicazione richiede un callback, le regole per il richiamo del consumer sono le seguenti:

REGISTRAZIONE
È sempre il primo tipo di richiamo del callback.

Viene sempre richiamato sullo stesso thread della chiamata MQCB (CBREG).

INIZIO
Viene sempre richiamato in modo sincrono con il verbo MQCTL (START).

- Tutte le richiamate START vengono completate prima della restituzione del verbo MQCTL (START).

Si trova sullo stesso thread della consegna del messaggio se è richiesto CTLTHR.

La chiamata con avvio non è garantita se, ad esempio, un callback precedente emette MQCTL (STOP) durante MQCTL (START).

ARRESTA
Non vengono consegnati ulteriori messaggi o eventi dopo questa chiamata fino a quando la connessione non viene riavviata.

Un STOP è garantito se l'applicazione è stata precedentemente richiamata per START, un messaggio o un evento.

DERISTER
È sempre l'ultimo tipo di richiamo del callback.

Verificare che l'applicazione esegua l'inizializzazione e la ripulitura basata sui thread nei callback START e STOP. È possibile eseguire l'inizializzazione e la ripulitura non basate su thread con callback REGISTER e DEREGISTER.

Non fare alcuna supposizione sulla vita e la disponibilità del thread diverso da quello che viene dichiarato. Ad esempio, non fare affidamento su un thread che rimane attivo oltre l'ultima chiamata a DEREGISTER. Allo stesso modo, quando si sceglie di non utilizzare CTLTHR, non presumere che il thread esista ogni volta che viene avviata la connessione.

Se l'applicazione ha particolari requisiti per le caratteristiche del thread, può sempre creare un thread di conseguenza, quindi utilizzare MQCTL (WAIT). Questo passo *dona* il thread a IBM MQ per la consegna asincrona dei messaggi.

Utilizzo della connessione dell'utente del messaggio

Di solito, quando un'applicazione emette un'altra chiamata MQI mentre una è in sospeso, la chiamata ha esito negativo con codice di errore RC2219.

Ci sono casi speciali, tuttavia, in cui l'applicazione deve emettere un'altra chiamata MQI prima che la chiamata precedente sia stata completata. Ad esempio, il consumer può essere richiamato durante una chiamata MQCB con CBRE.

In tale istanza, quando come risultato dell'applicazione che emette un comando MQCB o MQCTL, l'applicazione viene richiamata, all'applicazione è consentito emettere un'ulteriore chiamata MQI. Questa istanza significa che è possibile emettere, ad esempio, una chiamata MQOPEN, nella funzione consumer quando viene richiamata con un tipo CBCCALLT di CBCTRC. È consentita qualsiasi chiamata MQI, ad eccezione di MQDISC.

Parametri per MQCB

La chiamata MQCB ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Gestione della funzione di richiamata - parametro HCONN.

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

OPERATN (numero intero con segno a 10 cifre) - input

Gestisci funzione di callback - parametro OPERATN.

L'operazione in fase di elaborazione sul callback definito per l'handle oggetto specificato. È necessario specificare una delle seguenti opzioni; se è richiesta più di una opzione, i valori possono essere aggiunti (non aggiungere la stessa costante più di una volta) o combinati utilizzando l'operazione OR bitwise (se il linguaggio di programmazione supporta le operazioni bit).

Le combinazioni non valide vengono annotate; tutte le altre combinazioni sono valide.

CBREG
Definire la funzione di callback per l'handle oggetto specificato. Questa operazione definisce la funzione da chiamare e i criteri di selezione da utilizzare.

Se una funzione di callback è già definita per la gestione dell'oggetto, la definizione viene sostituita. Se viene rilevato un errore durante la sostituzione del callback, la registrazione della funzione viene annullata.

Se un callback è registrato nella stessa funzione di callback in cui è stata precedentemente annullata la registrazione, viene considerato come un'operazione di sostituzione; tutte le chiamate iniziali o finali non vengono richiamate.

È possibile utilizzare CBREG con CTLSU o CTLRE.

CBUNR
Arresta l'utilizzo dei messaggi per l'handle dell'oggetto e rimuove l'handle da quelli idonei per una richiamata.

La registrazione di un callback viene annullata automaticamente se l'handle associato è chiuso.

Se CBUNR viene chiamato dall'interno di un consumer e il callback ha una chiamata di arresto definita, viene richiamato al ritorno dal consumer.

Se questa operazione viene emessa rispetto a un *Hobj* senza un consumer registrato, la chiamata viene restituita con RC2448.

CTLSU
Sospende l'utilizzo dei messaggi per l'handle dell'oggetto.

Se questa operazione viene applicata a un gestore eventi, il gestore eventi non riceve gli eventi mentre è sospeso e gli eventi mancati mentre si trovano nello stato sospeso non vengono forniti all'operazione quando viene ripresa.

Mentre è sospesa, la funzionalità consumer continua a richiamare i callback di tipo controllo.

CTLRE
Riprendere l'utilizzo dei messaggi per l'handle dell'oggetto.

Se questa operazione viene applicata a un gestore eventi, il gestore eventi non riceve gli eventi mentre è sospeso e gli eventi mancati mentre si trovano nello stato sospeso non vengono forniti all'operazione quando viene ripresa.

CBDSC (MQCBD) - input

Gestione funzione di callback - parametro CBDSC.

Questa è una struttura che identifica la funzione di callback che viene registrata dall'applicazione e le opzioni utilizzate durante la registrazione.

Consultare "MQCBD - Descrittore di callback" a pagina 293 per i dettagli della struttura.

Il descrittore di callback è richiesto solo per l'opzione CBREG; se il descrittore non è richiesto, l'indirizzo del parametro passato può essere null.

HOBJ (numero intero con segno a 10 cifre) - immissione

Gestione funzione di callback - parametro HOBJ.

Questo handle rappresenta l'accesso stabilito all'oggetto da cui deve essere utilizzato un messaggio. Questo è un handle restituito da una chiamata MQOPEN o MQSUB precedente (nel parametro **HOBJ**).

HOBJ non è richiesto quando si definisce una routine del gestore eventi (CBTEH) e deve essere specificato come HONONE.

Se questo *Hobj* è stato restituito da una chiamata MQOPEN, la coda deve essere stata aperta con una o più delle seguenti opzioni:

- OOINPS
- OOINPX
- OOINPQ
- OOBRW

MSGDSC (MQMD) - immissione

Gestire il parametro -MSGDSC della funzione callback.

Questa struttura descrive gli attributi del messaggio richiesto e gli attributi del messaggio richiamato.

Il parametro **MsgDesc** definisce gli attributi dei messaggi richiesti dal consumer e la versione di MQMD da trasmettere al consumer del messaggio.

MsgId, *CorrelId*, *GroupId*, *MsgSeqNumbere* *Offset* in MQMD vengono utilizzati per selezionare i messaggi, in base alle opzioni specificate nel parametro **GetMsgOpts** .

Encoding e *CodedCharSetId* vengono utilizzati per la conversione del messaggio se si specifica l'opzione GMCONV.

Per i dettagli, consultare MQMD .

MsgDesc viene utilizzato solo per CBREG e, se si richiedono valori diversi da quelli predefiniti per qualsiasi campo. *MsgDesc* non viene utilizzato per un gestore eventi.

Se il descrittore non è richiesto, l'indirizzo del parametro passato può essere null.

Notare che se più consumer sono registrati nella stessa coda con selettori che si sovrappongono, il consumer scelto per ogni messaggio non è definito.

GMO (MQGMO) - input

Gestisci funzione di callback - parametro GMO.

Opzioni che controllano il modo in cui il consumatore di messaggi riceve i messaggi.

Tutte le opzioni hanno il significato descritto in "MQGMO (opzioni Get - message) su IBM i" a pagina 1105, quando vengono utilizzate su una chiamata MQGET, tranne:

GMSSIG
Questa opzione non è consentita.

GMBRWF, GMBRWN, GMMBH, GMMBC
L'ordine dei messaggi consegnati a un utente di navigazione è dettato dalle combinazioni di queste opzioni. Le combinazioni significative sono:

GMBRWF
Il primo messaggio sulla coda viene consegnato ripetutamente al consumer. Ciò è utile quando il consumer utilizza in modo distruttivo il messaggio nel callback. Utilizzare questa opzione con attenzione.

GMBRWN
Al consumer viene assegnato ogni messaggio sulla coda, dalla posizione corrente del cursore fino al raggiungimento della fine della coda.

GMBRWF + GMBRWN
Il cursore viene reimpostato all'inizio della coda. Al consumer viene quindi fornito ogni messaggio fino a quando il cursore non raggiunge la fine della coda.

GMBRWF + GMMBH o GMMBC
A partire dall'inizio della coda, il consumer riceve il primo messaggio non contrassegnato sulla coda, che viene quindi contrassegnato per questo consumer. Questa combinazione garantisce che il consumer possa ricevere nuovi messaggi aggiunti dietro il punto cursore corrente.

GMBRWN + GMMBH o GMMBC
Iniziando dalla posizione del cursore, il consumer riceve il successivo messaggio non contrassegnato sulla coda, che viene quindi contrassegnato per questo consumer. Utilizzare questa combinazione con attenzione perché i messaggi possono essere aggiunti alla coda dietro la posizione del cursore corrente.

GMBRWF + GMBRWN + GMMBH o GMMBC
Questa combinazione non è consentita, se utilizzata, la chiamata restituisce RC2046.

GMNWT, GMWT e GMWI
Queste opzioni controllano la modalità di richiamo del consumer.

GMNWT
Il consumer non viene mai richiamato con RC2033. Il consumer viene richiamato solo per messaggi ed eventi

GMWT con zero GMWI
Il codice RC2033 viene inoltrato al consumer solo quando non sono presenti messaggi e

- il consumer è stato avviato
- il destinatario ha ricevuto almeno un messaggio dall'ultimo codice di errore di nessun messaggio.

Ciò impedisce al consumer di eseguire il polling in un loop occupato quando viene specificato un intervallo di attesa zero.

GMWT e un GMWI positivo
L'utente viene richiamato dopo l'intervallo di attesa specificato con codice di errore RC2033. Questa chiamata viene effettuata indipendentemente dal fatto che i messaggi siano stati consegnati al consumer. Ciò consente all'utente di eseguire l'elaborazione di tipo heartbeat o batch.

GMWT e GMWI di WIULIM
Ciò specifica un'attesa infinita prima di restituire RC2033. Il consumer non viene mai richiamato con RC2033.

GMO viene utilizzato solo per CBREG e, se si richiedono valori diversi da quelli predefiniti per qualsiasi campo. *GMO* non viene utilizzato per un gestore eventi.

Se le opzioni non sono richieste, l'indirizzo del parametro passato può essere null.

Se nella struttura MQGMO viene fornito un handle delle proprietà del messaggio, viene fornita una copia nella struttura MQGMO che viene passata al callback del consumer. Al ritorno dalla chiamata MQCB, è possibile che l'applicazione elimini l'handle delle proprietà del messaggio.

CMPCOD (numero intero con segno a 10 cifre) - output

Gestione funzione di callback - parametro CMPCOD.

Il codice di completamento; è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Gestisci funzione di callback - parametro REASON.

I seguenti codici motivo sono i codici che il gestore code può restituire per il parametro **REASON** .

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è CCFAIL:

RC2204
(2204, X'89C') Adattatore non disponibile.

RC2133
(2133, X'855 ') Impossibile caricare i moduli dei servizi di conversione dati.

RC2130
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

RC2374
(2374, X' 946 ') Uscita API non riuscita.

RC2183
(2183, X'887 ') Impossibile caricare l'uscita API.

RC2157
(2157, X'86D') Gli ASID principale e home differiscono.

RC2005
(2005, X'7D5') Parametro di lunghezza del buffer non valido.

RC2219
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

RC2487
(2487, X'9B7') Campo di tipo callback non corretto.

RC2448
(2448, X' 990 ') Impossibile annullare la registrazione, sospendere o riprendere poiché non è presente alcun callback registrato.

RC2486
(2486, X'9B6') È necessario specificare *CallbackFunction* o *CallbackName* , ma non entrambi.

RC2483
(2483, X'9B3') Campo di tipo callback non corretto.

RC2484
(2484, X'9B4') Campo di opzioni MQCBD non corretto.

RC2140
(2140, X'85C') Richiesta di attesa rifiutata da CICS.

RC2009
(2009, X'7D9') Connessione al gestore code persa.

RC2217
(2217, X'8A9') Non autorizzato per la connessione.

RC2202
(2202, X'89A') Connessione in fase di sospensione.

RC2203
(2203, X'89B') Chiusura della connessione.

RC2207
(2207, X'89F') Errore identificativo di correlazione.

RC2010
(2010, X'7DA') Parametro di lunghezza dati non valido.

RC2016
(2016, X'7E0') Ottiene inibiti per la coda.

RC2351
(2351, X'92F') Unità globali di conflitto di lavoro.

RC2186
(2186, X'88A') Struttura delle opzioni Get - message non valida.

RC2353
(2353, X' 931 ') Handle in uso per l'unità di lavoro globale.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2019
(2019, X'7E3') Handle oggetto non valido.

RC2259
(2259, X'8D3') Specifica di ricerca incongruente.

RC2245
(2245, X'8C5') Specifica dell'unità di lavoro non congruente.

RC2246
(2246, X'8C6') Messaggio sotto il cursore non valido per il recupero.

RC2352
(2352, X' 930 ') L'unità di lavoro globale è in conflitto con l'unità di lavoro locale.

RC2247
(2247, X'8C7') Opzioni di corrispondenza non valide.

RC2485
(2485, X'9B4') Campo *MaxMsgLength* non corretto.

RC2026
(2026, X'7EA') Descrittore messaggio non valido.

RC2497
(2497, X'9C1') Il punto di ingresso della funzione specificato non è stato trovato nel modulo.

RC2496
(2496, X'9C0') È stato trovato un modulo, tuttavia è del tipo errato; non a 32 bit, 64 bit o una DLL (dynamic link library) valida.

RC2495
(2495, X'9BF') Modulo non trovato nel percorso di ricerca o non autorizzato al caricamento.

RC2250
(2250, X'8CA') Numero di sequenza messaggio non valido.

RC2331
(2331, X'91B') L'utilizzo del token del messaggio non è valido.

RC2033
(2033, X'7F1') Nessun messaggio disponibile.

RC2034
(2034, X'7F2') Il cursore di ricerca non è posizionato sul messaggio.

RC2036
(2036, X'7F4') Coda non aperta per la ricerca.

RC2037
(2037, X'7F5') Coda non aperta per l'input.

RC2041
(2041, X'7F9') Definizione oggetto modificata dall'apertura.

RC2101
(2101, X'835 ') Oggetto danneggiato.

RC2206
(2206, X'89E') Codice di operazione non corretto nella chiamata API.

RC2046
(2046, X'7FE') Opzioni non valide o non congruenti.

RC2193
(2193, X'891 ') Errore durante l'accesso al dataset della serie di pagine.

RC2052
(2052, X'804 ') La coda è stata eliminata.

RC2394
(2394, X'95A') La coda ha un tipo di indice errato.

RC2058
(2058, X'80A') Nome gestore code non valido o sconosciuto.

RC2059
(2059, X'80B') Gestore code non disponibile per la connessione.

RC2161
(2161, X'871 ') Gestore code in fase di sospensione.

RC2162
(2162, X'872 ') Chiusura del gestore code.

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2069
(2069, X'815 ') Segnale eccezionale per questa maniglia.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2109
(2109, X'83D') Chiamata eliminata dal programma di uscita.

RC2024
(2024, X'7E8') Non è possibile gestire ulteriori messaggi all'interno dell'unità di lavoro corrente.

RC2072
(2072, X'818 ') Supporto punto di sincronizzazione non disponibile.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

RC2354
(2354, X' 932 ') L'inserimento nell'unità di lavoro globale non è riuscito.

RC2355
(2355, X' 933 ') La miscelazione delle chiamate UOW non è supportata.

RC2255
(2255, X'8CF') Unità di lavoro non disponibile per il gestore code.

RC2090
(2090, X'82A') Intervallo di attesa in MQGMO non valido.

RC2256
(2256, X'8D0') Versione errata di MQGMO fornita.

RC2257
(2257, X'8D1') Versione non corretta di MQMD fornita.

RC2298
(2298, X'8FA') La funzione richiesta non è disponibile nell'ambiente corrente.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCB(HCONN : OPERATN : CBDSC :
                                HOBJ : MSGDSC : GMO :
                                DATLEN : CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
DMQCB             PR                  EXTPROC('MQCB')
D* Connection handle
D HCONN                          10I 0 VALUE
D* Operation
D OPERATN                        10I 0 VALUE
D* Callback descriptor
D CBDSC                         180A
D* Object handle
D HOBJ                           10I 0 VALUE
D* Message Descriptor
D MSGDSC                        364A
D* Get options
D GMO                           112A
D* Completion code
D CMPCOD                         10I 0
* Reason code qualifying CompCode
D REASON                         10I 0
```

IBM i MQCLOSE (Chiudi oggetto) su IBM i

La chiamata MQCLOSE rinuncia all'accesso ad un oggetto ed è l'inverso della chiamata MQOPEN.

- “Dichiarazione RPG” a pagina 1310

Sintassi

MQCLOSE *(HCONN, HOBJ, OPTS, CMPCOD, REASON)*

Note d'utilizzo

1. Quando un'applicazione emette la chiamata MQDISC o termina normalmente o in modo anomalo, tutti gli oggetti aperti dall'applicazione e ancora aperti vengono chiusi automaticamente con l'opzione CONONE.
2. I seguenti punti si applicano se l'oggetto da chiudere è una *coda*:
   - Se le operazioni sulla coda vengono eseguite come parte di un'unità di lavoro, la coda può essere chiusa prima o dopo che si verifichi il punto di sincronizzazione senza influenzare il risultato del punto di sincronizzazione.
   - Se la coda è stata aperta con l'opzione OOBRW, il cursore di ricerca viene distrutto. Se la coda viene successivamente riaperta con l'opzione OOBRW, viene creato un nuovo cursore di ricerca (consultare l'opzione OOBRW descritta in MQOPEN).
   - Se un messaggio è attualmente bloccato per questo handle al momento della chiamata MQCLOSE, il blocco viene rilasciato (consultare l'opzione GMLK descritta in “MQGMO (opzioni Get - message) su IBM i” a pagina 1105).
3. I seguenti punti si applicano se l'oggetto che si sta chiudendo è una *coda dinamica* (permanente o temporanea):
   - Per una coda dinamica, le opzioni CODEL o COPURG possono essere specificate indipendentemente dalle opzioni specificate nella chiamata MQOPEN corrispondente.
   - Quando una coda dinamica viene eliminata, tutte le chiamate MQGET con l'opzione GMWT in sospeso rispetto alla coda vengono annullate e viene restituito il codice motivo RC2052 . Consultare l'opzione GMWT descritta in “MQGMO (opzioni Get - message) su IBM i” a pagina 1105.

     Dopo che una coda dinamica è stata eliminata, qualsiasi chiamata (diversa da MQCLOSE) che tenta di fare riferimento alla coda utilizzando un handle *HOBJ* precedentemente acquisito ha esito negativo con codice motivo RC2052.

     Tenere presente che, sebbene le applicazioni non possano accedere a una coda eliminata, la coda non viene rimossa dal sistema e le risorse associate non vengono liberate, fino a quando tutti gli handle che fanno riferimento alla coda non sono stati chiusi e tutte le unità di lavoro che influiscono sulla coda non sono state sottoposte a commit o a backout.
   - Quando una coda dinamica permanente viene eliminata, se l'handle *HOBJ* specificato nella chiamata MQCLOSE non è quello restituito dalla chiamata MQOPEN che ha creato la coda, viene eseguito un controllo che l'identificativo utente utilizzato per convalidare la chiamata MQOPEN sia autorizzato ad eliminare la coda. Se l'opzione OOALTU è stata specificata nella chiamata MQOPEN, l'identificativo utente selezionato è *ODAU*.

     Questo controllo non viene eseguito se:

     – L'handle specificato è quello restituito dalla chiamata MQOPEN che ha creato la coda.

     – La coda che si sta eliminando è una coda dinamica temporanea.
   - Quando una coda dinamica temporanea viene chiusa, se l'handle *HOBJ* specificato nella chiamata MQCLOSE è quello restituito dalla chiamata MQOPEN che ha creato la coda, la coda viene eliminata. Ciò si verifica indipendentemente dalle opzioni di chiusura specificate nella chiamata MQCLOSE. Se ci sono messaggi nella coda, vengono eliminati; non viene generato alcun messaggio di report.

     Se ci sono unità di lavoro non sottoposte a commit che influiscono sulla coda, la coda e i relativi messaggi vengono ancora eliminati, ma ciò non causa l'errore delle unità di lavoro. Tuttavia, come descritto in precedenza, le risorse associate alle unità di lavoro non vengono liberate fino a quando non viene eseguito il commit o il backout di ciascuna unità di lavoro.

4. I seguenti punti si applicano se l'oggetto da chiudere è un *elenco di distribuzione*:
   - L'unica opzione di chiusura valida per un elenco di distribuzione è CONONE; la chiamata ha esito negativo con codice di errore RC2046 o RC2045 se vengono specificate altre opzioni.
   - Quando un elenco di distribuzione viene chiuso, i singoli codici di completamento e di errore non vengono restituiti per le code nell'elenco - solo i parametri **CMPCOD** e **REASON** della chiamata sono disponibili per scopi diagnostici.

     Se si verifica un errore durante la chiusura di una delle code, il gestore code continua l'elaborazione e tenta di chiudere le code rimanenti nell'elenco di distribuzione. I parametri **CMPCOD** e **REASON** della chiamata vengono quindi impostati per restituire le informazioni che descrivono l'errore. Pertanto, è possibile che il codice di completamento sia CCFAIL, anche se la maggior parte delle code è stata chiusa correttamente. La coda che ha rilevato l'errore non è stata identificata.

     Se si verifica un errore su più di una coda, non è definito quale errore viene riportato nei parametri **CMPCOD** e **REASON** .

Parametri

La chiamata MQCLOSE ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Handle di connessione.

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

HOBJ (numero intero con segno a 10 cifre) - input/output

Handle oggetto.

Questo handle rappresenta l'oggetto in fase di chiusura. L'oggetto può essere di qualsiasi tipo. Il valore di *HOBJ* è stato restituito da una chiamata MQOPEN precedente.

Al corretto completamento della chiamata, il gestore code imposta questo parametro su un valore che non è un handle valido per l'ambiente. Questo valore è:

HOUNUH
Gestione oggetto non utilizzabile.

OPTS (numero intero con segno a 10 cifre) - immissione

Opzioni che controllano l'azione di MQCLOSE.

Il parametro **OPTS** controlla come viene chiuso l'oggetto. Solo le sottoscrizioni e le code dinamiche permanenti possono essere chiuse in più modi. Le code dinamiche permanenti possono essere conservate o eliminate; si tratta di code con un attributo **DefinitionType** che ha il valore QDPERM (consultare l'attributo **DefinitionType** descritto in “Attributi per le code” a pagina 1406). Le opzioni di chiusura sono riepilogate in una tabella più avanti in questo argomento.

Le sottoscrizioni durevoli possono essere conservate o rimosse; queste vengono create utilizzando la chiamata MQSUB con l'opzione SODUR.

Quando si chiude l'handle su una destinazione gestita (ovvero il parametro **Hobj** restituito su una chiamata MQSUB che ha utilizzato l'opzione SOMAN), il gestore code ripulirà tutte le pubblicazioni non richiamate quando anche la sottoscrizione associata è stata rimossa. Questa operazione viene eseguita utilizzando l'opzione CORMSB sul parametro **Hsub** restituito su una chiamata MQSUB. Si noti che CORMSB è il funzionamento predefinito su MQCLOSE per una sottoscrizione non durevole.

Quando si chiude un handle per una destinazione non gestita, l'utente è responsabile della ripulitura della coda in cui vengono inviate le pubblicazioni. Si consiglia di chiudere prima la sottoscrizione utilizzando CORMSB ed elaborare i messaggi dalla coda fino a quando non ne rimane nessuno.

È necessario specificare uno (e solo uno) dei seguenti:

Opzioni di chiusura della coda dinamica

Queste opzioni controllano la modalità di chiusura delle code dinamiche permanenti:

CODEL
Eliminare la coda.

La coda viene eliminata se si verifica una delle seguenti condizioni:

- Si tratta di una coda dinamica permanente, creata da una precedente chiamata MQOPEN, e non ci sono messaggi sulla coda e non ci sono richieste get o put non sottoposte a commit in attesa per la coda (sia per l'attività corrente che per qualsiasi altra attività).
- È la coda dinamica temporanea creata dalla chiamata MQOPEN che ha restituito *H0BJ*. In questo caso, tutti i messaggi sulla coda vengono eliminati.

In tutti gli altri casi, incluso il caso in cui *Hobj* è stato restituito su una chiamata MQSUB, la chiamata ha esito negativo con codice di errore RC2045e l'oggetto non viene eliminato.

COPURG
Eliminare la coda, eliminando tutti i messaggi su di essa.

La coda viene eliminata se si verifica una delle seguenti condizioni:

- Si tratta di una coda dinamica permanente, creata da una precedente chiamata MQOPEN, e non vi sono richieste get o put non sottoposte a commit in sospeso per la coda (per l'attività corrente o per qualsiasi altra attività).
- È la coda dinamica temporanea creata dalla chiamata MQOPEN che ha restituito *H0BJ*.

In tutti gli altri casi, incluso il caso in cui *Hobj* è stato restituito su una chiamata MQSUB, la chiamata ha esito negativo con codice di errore RC2045e l'oggetto non viene eliminato.

La tabella successiva mostra quali opzioni di chiusura sono valide e se l'oggetto viene conservato o eliminato.

Tabella 743. Opzioni di chiusura valide per l'utilizzo con oggetti conservati o eliminati

Tipo di oggetto o coda	**CONONE**	**CODEL**	**COPURG**
Oggetto diverso da una coda	Conservato	Non valido	Non valido
Coda predefinita	Conservato	Non valido	Non valido
coda dinamica permanente	Conservato	Eliminato se vuoto e nessun aggiornamento in sospeso	Messaggi eliminati; coda eliminata se non vi sono aggiornamenti in sospeso
Coda dinamica temporanea (chiamata emessa dal creatore della coda)	Eliminato	Eliminato	Eliminato
Coda dinamica temporanea (chiamata non emessa dal creatore della coda)	Conservato	Non valido	Non valido
Elenco di distribuzione	Conservato	Non valido	Non valido
Destinazione sottoscrizione gestita	Conservato	Non valido	Non valido
Elenco di distribuzione (la sottoscrizione è stata rimossa)	Messaggi eliminati; coda eliminata	Non valido	Non valido

Opzioni di chiusura sottoscrizione

Queste opzioni controllano se le sottoscrizioni durevoli vengono rimosse quando l'handle viene chiuso e se le pubblicazioni ancora in attesa di essere lette dall'applicazione vengono ripulite. Queste

opzioni sono valide solo per l'utilizzo con un handle dell'oggetto restituito nel parametro **HSUB** di una chiamata MQSUB.

COKPSB
L'handle per la sottoscrizione è chiuso ma la sottoscrizione effettuata viene mantenuta. Le pubblicazioni continueranno ad essere inviate alla destinazione specificata nella sottoscrizione. Questa opzione è valida solo se la sottoscrizione è stata effettuata con l'opzione SODUR. COKPSB è il valore predefinito se la sottoscrizione è durevole

CORMSB
La sottoscrizione viene rimossa e l'handle della sottoscrizione viene chiuso.

Il parametro **Hobj** della chiamata MQSUB non è invalidato dalla chiusura del parametro **Hsub** e può continuare ad essere utilizzato per MQGET o MQCB per ricevere le pubblicazioni rimanenti. Quando viene chiuso anche il parametro **Hobj** della chiamata MQSUB, se si trattava di una destinazione gestita tutte le pubblicazioni non richiamate verranno rimosse.

CORMSB è il valore predefinito se la sottoscrizione non è durevole.

Queste opzioni di chiusura della sottoscrizione sono riepilogate nelle tabelle seguenti:

Per chiudere un handle di sottoscrizione durevole ma lasciare la sottoscrizione, utilizzare le seguenti opzioni di chiusura della sottoscrizione:

Tabella 744. Opzioni dell'attività per chiudere un handle di sottoscrizione durevole e lasciare la sottoscrizione

Attività	**Opzione di chiusura sottoscrizione**
Conserva pubblicazioni su un handle MQOPENed	COKPSB
Rimuovi pubblicazioni su un handle MQOPENed	Azione non consentita
Conserva le pubblicazioni su un handle con SOMAN	COKPSB
Rimuovere le pubblicazioni su un handle con SOMAN	Azione non consentita

Per annullare la sottoscrizione, chiudendo un handle di sottoscrizione durevole e annullando la sottoscrizione o chiudendo un handle di sottoscrizione non durevole, utilizzare le seguenti opzioni di chiusura della sottoscrizione:

Tabella 745. Opzioni attività per l'annullamento della sottoscrizione

Attività	**Opzione di chiusura sottoscrizione**
Conserva pubblicazioni su un handle MQOPENed	CORMSB
Rimuovi pubblicazioni su un handle MQOPENed	Azione non consentita
Conserva le pubblicazioni su un handle con SOMAN	CORMSB
Rimuovere le pubblicazioni su un handle con SOMAN	COPGSB

Opzioni di lettura anticipata

Le seguenti opzioni controllano cosa accade ai messaggi non persistenti che sono stati inviati al client prima che un'applicazione li richiedesse e non sono stati ancora utilizzati dall'applicazione. Questi messaggi vengono memorizzati nel buffer di lettura anticipata del client in attesa di essere richiesti dall'applicazione e possono essere eliminati o consumati dalla coda prima del completamento di MQCLOSE.

COIMM
L'oggetto viene chiuso immediatamente e tutti i messaggi che sono stati inviati al client prima che un'applicazione li richiedesse vengono eliminati e non sono disponibili per essere utilizzati da alcuna applicazione. Questo è il valore predefinito.

COQSC
Viene effettuata una richiesta di chiusura dell'oggetto, ma se i messaggi che sono stati inviati al client prima che un'applicazione li richiedesse, si trovano ancora nel buffer di lettura anticipata del client, la chiamata MQCLOSE restituirà un codice di avvertenza di RC2458e l'handle dell'oggetto rimarrà valido.

L'applicazione può quindi continuare a utilizzare l'handle dell'oggetto per richiamare i messaggi fino a quando non sono più disponibili e quindi chiudere nuovamente l'oggetto. Non verranno inviati ulteriori messaggi al client prima di una richiesta di applicazione, quindi, la lettura anticipata è ora disattivata.

Si consiglia alle applicazioni di utilizzare COQSC piuttosto che tentare di raggiungere un punto in cui non ci sono più messaggi nel buffer di lettura anticipata del client, poiché un messaggio potrebbe arrivare tra l'ultima chiamata MQGET e il seguente MQCLOSE che sarebbe stato eliminato se fosse stato utilizzato COIMM.

Se un MQCLOSE con COQSC viene emesso dall'interno di una funzione di callback asincrona, si applica lo stesso comportamento dei messaggi di lettura anticipata. Se viene restituito il codice di avvertenza RC2458 , la funzione di callback verrà richiamata almeno un'altra volta. Quando l'ultimo messaggio rimanente che era stato letto in anticipo è stato passato alla funzione di callback, il campo CBCFLG è impostato su CBCFBE.

Opzione predefinita

Se non si richiede alcuna delle opzioni descritte in precedenza, è possibile utilizzare la seguente opzione:

CONONE
Non è richiesta alcuna elaborazione di chiusura facoltativa.

Deve essere specificato per:

- Oggetti diversi dalle code
- Code predefinite
- Code dinamiche temporanee (ma solo nei casi in cui *HOBJ* non è l'handle restituito dalla chiamata MQOPEN che ha creato la coda).
- Liste di distribuzione

In tutti i casi precedenti, l'oggetto viene conservato e non eliminato.

Se questa opzione viene specificata per una coda dinamica temporanea:

- La coda viene eliminata, se è stata creata dalla chiamata MQOPEN che ha restituito *HOBJ* ; tutti i messaggi presenti nella coda vengono eliminati.
- In tutti gli altri casi, la coda (e gli eventuali messaggi su di essa) vengono conservati.

Se questa opzione viene specificata per una coda dinamica permanente, la coda viene conservata e non eliminata.

CMPCOD (numero intero con segno a 10 cifre) - output

Codice di completamento.

Il valore è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Codice di errore *CMPCOD*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCWARN:

RC2241
(2241, X'8C1') Gruppo di messaggi non completo.

RC2242
(2242, X'8C2') Messaggio logico non completo.

Se *CMPCOD* è CCFAIL:

RC2219
(2219, X'8AB') Chiamata MQI reimmessa prima del completamento della chiamata precedente.

RC2009
(2009, X'7D9') Connessione al gestore code persa.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2019
(2019, X'7E3') Handle oggetto non valido.

RC2035
(2035, X'7F3') Non autorizzato per l'accesso.

RC2101
(2101, X'835 ') Oggetto danneggiato.

RC2045
(2045, X'7FD') Opzione non valida per il tipo di oggetto.

RC2046
(2046, X'7FE') Opzioni non valide o non congruenti.

RC2058
(2058, X'80A') Nome gestore code non valido o sconosciuto.

RC2059
(2059, X'80B') Gestore code non disponibile per la connessione.

RC2162
(2162, X'872 ') Chiusura del gestore code.

RC2055
(2055, X'807 ') La coda contiene uno o più messaggi o richieste put o get senza commit.

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2063
(2063, X'80F') Si è verificato un errore di sicurezza.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCLOSE(HCONN : HOBJ : OPTS :
C                                       CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQCLOSE          PR                  EXTPROC('MQCLOSE')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Object handle
D HOBJ                          10I 0
D* Options that control the action of MQCLOSE
D OPTS                          10I 0 VALUE
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CMPCOD
D REASON                        10I 0
```

IBM i MQCMIT (Commit modifiche) su IBM i

La chiamata MQCMIT indica al gestore code che l'applicazione ha raggiunto un punto di sincronizzazione e che tutti i richiami e gli inserimenti di messaggi che si sono verificati dall'ultimo punto di sincronizzazione devono essere resi permanenti. I messaggi inseriti come parte di un'unità di lavoro vengono resi disponibili ad altre applicazioni; i messaggi richiamati come parte di un'unità di lavoro vengono eliminati.

Sintassi

MQCMIT *(HCONN, COMCOD, REASON)*

Note d'utilizzo

Considerare queste note di utilizzo quando si utilizza MQCMIT.

1. Questa chiamata può essere utilizzata solo quando il gestore code stesso coordina l'unità di lavoro. Si tratta di un'unità di lavoro locale, in cui le modifiche riguardano solo le risorse IBM MQ .
2. In ambienti in cui il gestore code non coordina l'unità di lavoro, è necessario utilizzare la chiamata di commit appropriata al posto di MQCMIT. L'ambiente può anche supportare un commit implicito causato dalla chiusura normale dell'applicazione.
   - Su IBM i, questa chiamata può essere utilizzata per unità di lavoro locali coordinate dal gestore code. Ciò significa che una definizione di commit non deve esistere a livello di lavoro, ovvero il comando STRCMTCTL con il parametro **CMTSCOPE(*JOB)** non deve essere stato immesso per il lavoro.
3. Se un'applicazione termina con modifiche non sottoposte a commit in un'unità di lavoro, la disposizione di tali modifiche dipende dal fatto che l'applicazione termini normalmente o in modo anomalo. Consultare le note sull'utilizzo in "MQDISC (Disconnetti gestore code) su IBM i" a pagina 1327 per ulteriori dettagli.
4. Quando un'applicazione inserisce o richiama i messaggi in gruppi o segmenti di messaggi logici, il gestore code conserva le informazioni relative al gruppo di messaggi e al messaggio logico per le ultime chiamate MQPUT e MQGET riuscite. Queste informazioni sono associate all'handle della coda e includono:
   - I valore dei campi *MDGID*, *MDSEQ*, *MDOFF*e *MDMFL* in MQMD.
   - Se il messaggio fa parte di un'unità di lavoro.
   - Per la chiamata MQPUT: se il messaggio è persistente o non persistente.

   Quando viene eseguito il commit di un'unità di lavoro, il gestore code conserva le informazioni sul gruppo e sul segmento e l'applicazione può continuare a inserire o richiamare i messaggi nel gruppo di messaggi corrente o nel messaggio logico.

La conservazione delle informazioni sul gruppo e sul segmento quando viene eseguito il commit di un'unità di lavoro consente all'applicazione di distribuire un gruppo di messaggi di grandi dimensioni o un messaggio logico di grandi dimensioni costituito da molti segmenti su diverse unità di lavoro. L'utilizzo di diverse unità di lavoro potrebbe essere vantaggioso se il gestore code locale ha solo una memoria di coda limitata. Tuttavia, l'applicazione deve conservare informazioni sufficienti per essere in grado di riavviare l'inserimento o il richiamo dei messaggi nel punto corretto se si verifica un errore di sistema. Per dettagli su come riavviare il sistema nel punto corretto dopo un malfunzionamento del sistema, consultare l'opzione PMLOGO descritta in “MQPMO (Put - message options) su IBM i” a pagina 1206e l'opzione GMLOGO descritta in “MQGMO (opzioni Get - message) su IBM i” a pagina 1105.

Le note di uso rimanenti si applicano solo quando il gestore code coordina le unità di lavoro:

1. Un'unità di lavoro ha lo stesso ambito di un handle di connessione. Ciò significa che tutte le chiamate IBM MQ che interessano una particolare unità di lavoro devono essere eseguite utilizzando lo stesso handle di connessione. Le chiamate emesse utilizzando un handle di collegamento differente (ad esempio, le chiamate emesse da un'altra applicazione) influenzano un'unità di lavoro diversa. Consultare il parametro **HCONN** descritto in MQCONN per informazioni sull'ambito degli handle di connessione.
2. Solo i messaggi inseriti o richiamati come parte dell'unità di lavoro corrente vengono influenzati da questa chiamata.
3. Un'applicazione di lunga durata che emette chiamate MQGET, MQPUT o MQPUT1 all'interno di un'unità di lavoro, ma che non emette mai una chiamata di commit o di back-out, può causare il riempimento delle code con messaggi che non sono disponibili per altre applicazioni. Per evitare questa possibilità, l'amministratore deve impostare l'attributo del gestore code **MaxUncommittedMsgs** su un valore sufficientemente basso per evitare che le applicazioni runaway riempiano le code, ma sufficientemente alto per consentire il corretto funzionamento delle applicazioni di messaggistica previste.

Parametri

La chiamata MQCMIT ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Handle di connessione.

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

COMCOD (numero intero con segno a 10 cifre) - output

Codice di completamento.

Il valore è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Codice di errore *COMCOD*.

Se *COMCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *COMCOD* è CCWARN:

RC2003
(2003, X'7D3') Unità di lavoro ripristinata.

RC2124
(2124, X'84C') Il risultato dell'operazione di commit è in sospeso.

Se *COMCOD* è CCFAIL:

RC2219
(2219, X'8AB') Chiamata MQI reimmessa prima del completamento della chiamata precedente.

RC2009
(2009, X'7D9') Connessione al gestore code persa.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2101
(2101, X'835 ') Oggetto danneggiato.

RC2123
(2123, X'84B') Il risultato dell'operazione di commit o di backout è misto.

RC2162
(2162, X'872 ') Chiusura del gestore code.

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                   CALLP     MQCMIT(HCONN : COMCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQCMIT           PR                  EXTPROC('MQCMIT')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Completion code
D COMCOD                        10I 0
D* Reason code qualifying COMCOD
D REASON                        10I 0
```

IBM i MQCONN (gestore code di connessione) su IBM i

La chiamata MQCONN collega un programma applicativo a un gestore code. Fornisce un handle di connessione del gestore code, che è utilizzato dall'applicazione nelle successive chiamate di accodamento messaggi.

- Le applicazioni devono utilizzare la chiamata MQCONN o MQCONNX per connettersi al gestore code e la chiamata MQDISC per disconnettersi dal gestore code.

Su IBM MQ for Multiplatforms, ogni thread in un'applicazione può connettersi a gestori code differenti. Su altri sistemi, tutte le connessioni simultanee in un processo devono essere allo stesso gestore code.

- "Sintassi" a pagina 1314
- "Note d'utilizzo" a pagina 1314

- “Parametri” a pagina 1314
- “Dichiarazione RPG” a pagina 1317

Sintassi

MQCONN *(QMNAME, HCONN, CMPCOD, REASON)*

Note d'utilizzo

1. Il gestore code a cui viene effettuata una connessione utilizzando la chiamata MQCONN è denominato *gestore code locale*.
2. Le code di proprietà del gestore code locale vengono visualizzate dall'applicazione come code locali. È possibile inserire e richiamare messaggi da queste code.

   Le code condivise di proprietà del gruppo di condivisione code a cui appartiene il gestore code locale appaiono all'applicazione come code locali. È possibile inserire e richiamare messaggi da queste code.

   Le code di proprietà di gestori code remoti vengono visualizzate come code remote. È possibile inserire messaggi su queste code, ma non è possibile ottenere messaggi da queste code.
3. Se il gestore code non riesce mentre un'applicazione è in esecuzione, l'applicazione deve emettere nuovamente la chiamata MQCONN per ottenere un nuovo handle di connessione da utilizzare nelle chiamate IBM MQ successive. L'applicazione può emettere la chiamata MQCONN periodicamente fino a quando la chiamata ha esito positivo.

   Se un'applicazione non è sicura di essere connessa al gestore code, può tranquillamente emettere una chiamata MQCONN per ottenere un handle di connessione. Se l'applicazione è già connessa, l'handle restituito è lo stesso restituito dalla precedente chiamata MQCONN, ma con codice di completamento CCWARN e codice motivo RC2002.
4. Quando l'applicazione ha terminato di utilizzare le chiamate IBM MQ , deve utilizzare la chiamata MQDISC per disconnettersi dal gestore code.
5. Su IBM i, i programmi che terminano in modo anomalo non vengono automaticamente disconnessi dal gestore code. Pertanto, le applicazioni devono essere scritte per consentire la possibilità che la chiamata MQCONN o MQCONNX restituisca il codice di completamento CCWARN e il codice motivo RC2002. L'handle di collegamento restituito in questa situazione può essere utilizzato normalmente.

Parametri

La chiamata MQCONN ha i parametri seguenti:

QMNAME (stringa di caratteri a 48 byte) - input

Il nome del gestore code.

Questo è il nome del gestore code a cui l'applicazione desidera connettersi. Il nome può contenere i seguenti caratteri:

- Caratteri alfabetici maiuscoli (da A a Z)
- Caratteri alfabetici minuscoli (da a a z)
- Cifre numeriche (da 0 a 9)
- Punto (.), barra (/), sottolineatura (_), percentuale (%)

Il nome non deve contenere spazi iniziali o incorporati, ma può contenere spazi finali. Un carattere null può essere utilizzato per indicare la fine dei dati significativi nel nome; il valore null e i caratteri che lo seguono vengono trattati come spazi vuoti. Le seguenti limitazioni si applicano agli ambienti indicati:

- Su IBM i, i nomi contenenti caratteri minuscoli, barra o percentuale devono essere racchiusi tra virgolette quando vengono specificati nei comandi. Queste virgolette non devono essere specificate nel parametro **QMNAME** .

Se il nome è composto interamente da spazi vuoti, viene utilizzato il nome del gestore code *predefinito* .

Il nome specificato per *QMNAME* deve essere il nome di un gestore code *collegabile* .

Gruppi di condivisione code:Sui sistemi in cui esistono diversi gestori code e sono configurati per formare un gruppo di condivisione code, il nome del gruppo di condivisione code può essere specificato per *QMNAME* invece del nome di un gestore code. Ciò consente all'applicazione di connettersi a *qualsiasi* gestore code disponibile nel gruppo di condivisione code. Il sistema può essere configurato anche in modo che un *QMNAME* vuoto provochi la connessione al gruppo di condivisione code anziché al gestore code predefinito.

Se *QMNAME* specifica il nome del gruppo di condivisione code, ma è presente anche un gestore code con tale nome sul sistema, la connessione viene effettuata al secondo in preferenza al primo. Solo se la connessione non riesce, viene tentata la connessione a uno dei gestori code nel gruppo di condivisione code.

Se la connessione ha esito positivo, l'handle restituito dalla chiamata MQCONN o MQCONNX può essere utilizzato per accedere a *tutte* le risorse (condivise e non condivise) che appartengono al particolare gestore code a cui è stata effettuata la connessione. L'accesso a queste risorse è soggetto ai tipici controlli di autorizzazione.

Se l'applicazione emette due chiamate MQCONN o MQCONNX per stabilire connessioni simultanee e una o entrambe le chiamate specificano il nome del gruppo di condivisione code, la seconda chiamata potrebbe restituire il codice di completamento CCWARN e il codice di errore RC2002. Ciò si verifica quando la seconda chiamata si connette allo stesso gestore code della prima chiamata.

I gruppi di condivisione code sono supportati solo su z/OS. La connessione a un gruppo di condivisione code è supportata solo in ambienti batch, batch RRS e TSO.

applicazioni client IBM MQ: per applicazioni IBM MQ MQI client , viene tentata una connessione per ogni definizione di canale di connessione client con il nome gestore code specificato, fino a quando non ne viene eseguita una. Il gestore code, tuttavia, deve avere lo stesso nome del nome specificato. Se viene specificato un nome completamente vuoto, ogni canale di connessione client con un nome gestore code completamente vuoto viene tentato fino a quando non viene eseguito correttamente; in questo caso, non viene eseguito alcun controllo rispetto al nome effettivo del gestore code.

IBM MQ Gruppi di gestori code client: se il nome specificato inizia con un asterisco (*), il gestore code effettivo a cui viene effettuata la connessione potrebbe avere un nome diverso da quello specificato dall'applicazione. Il nome specificato (senza l'asterisco) definisce un *gruppo* di gestori code idonei per la connessione. L'implementazione ne seleziona uno dal gruppo provandone uno a turno, in ordine alfabetico, fino a quando non ne trova uno a cui è possibile stabilire una connessione. Se nessuno dei gestori code del gruppo è disponibile per la connessione, la chiamata non riesce. Ogni gestore code viene tentato una sola volta. Se per il nome viene specificato un asterisco, viene utilizzato un gruppo di gestori code predefinito definito dall'implementazione.

I gruppi di gestori code sono supportati solo per le applicazioni in esecuzione in un ambiente client MQ; la chiamata ha esito negativo se un'applicazione non client specifica un nome gestore code che inizia con un asterisco. Un gruppo viene definito fornendo diverse definizioni di canale di connessione client con lo stesso nome gestore code (il nome specificato senza l'asterisco), per comunicare con ciascuno dei gestori code del gruppo. Il gruppo predefinito viene definito fornendo una o più definizioni di canale di connessione client, ciascuna con un nome gestore code vuoto (specificare un nome tutto vuoto ha quindi lo stesso effetto di specificare un singolo asterisco per il nome di un'applicazione client).

Dopo la connessione a un gestore code di un gruppo, un'applicazione può specificare spazi vuoti nel modo tipico nei campi del nome del gestore code nei descrittori del messaggio e dell'oggetto per indicare il nome del gestore code a cui l'applicazione si è effettivamente connessa (il *gestore code locale*). Se l'applicazione deve conoscere questo nome, è possibile emettere la chiamata MQINQ per analizzare l'attributo del gestore code **QMgrName** .

Il prefisso di un asterisco al nome della connessione implica che l'applicazione non dipende dalla connessione a un determinato gestore code nel gruppo. Le applicazioni adatte sarebbero:

- Applicazioni che immettono messaggi ma non li ricevono.
- Le applicazioni che immettono i messaggi di richiesta e quindi ricevono i messaggi di risposta da una coda *dinamica temporanea* .

Le applicazioni non adatte sono quelle che devono richiamare i messaggi da una particolare coda in un determinato gestore code; tali applicazioni non devono anteporre un asterisco al nome.

Notare che se viene specificato un asterisco, la lunghezza massima del resto del nome è di 47 caratteri.

La lunghezza di questo parametro è fornita da LNQMN.

HCONN (numero intero con segno a 10 cifre) - output

Handle di connessione.

Questo handle rappresenta la connessione al gestore code. Deve essere specificato su tutte le successive chiamate di accodamento messaggi emesse dall'applicazione. Cessa di essere valida quando viene emessa la chiamata MQDISC o quando termina l'unità di elaborazione che definisce l'ambito dell'handle.

L'ambito della maniglia è limitato alla più piccola unità di l'elaborazione parallela supportata dalla piattaforma su cui è in esecuzione l'applicazione; l'handle non è valido all'esterno dell'unità di elaborazione parallela da cui è stata emessa la chiamata MQCONN.

- Su IBM i, l'ambito dell'handle è il lavoro che emette la chiamata.

CMPCOD (numero intero con segno a 10 cifre) - output

Codice di completamento.

Il valore è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Codice di errore *CMPCOD*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCWARN:

RC2002
(2002, X'7D2') Applicazione già connessa.

Se *CMPCOD* è CCFAIL:

RC2219
(2219, X'8AB') Chiamata MQI reimmessa prima del completamento della chiamata precedente.

RC2267
(2267, X'8DB') Impossibile caricare l'uscita del carico di lavoro del cluster.

RC2009
(2009, X'7D9') Connessione al gestore code persa.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2035
(2035, X'7F3') Non autorizzato per l'accesso.

RC2137
(2137, X'859 ') Oggetto non aperto correttamente.

RC2058
(2058, X'80A') Nome gestore code non valido o sconosciuto.

RC2059
(2059, X'80B') Gestore code non disponibile per la connessione.

RC2161
(2161, X'871 ') Gestore code in fase di sospensione.

RC2162
(2162, X'872 ') Chiusura del gestore code.

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2063
(2063, X'80F') Si è verificato un errore di sicurezza.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCONN(QMNAME : HCONN : CMPCOD :
C                                      REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQCONN           PR                  EXTPROC('MQCONN')
D* Name of queue manager
D QMNAME                        48A
D* Connection handle
D HCONN                         10I 0
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CMPCOD
D REASON                        10I 0
```

IBM i MQCONNX (Connetti gestore code (esteso)) su IBM i

La chiamata MQCONNX connette un programma applicativo a un gestore code. Fornisce un handle di connessione del gestore code, utilizzato dall'applicazione nelle successive chiamate IBM MQ .

La chiamata MQCONNX è simile alla chiamata MQCONN, ma MQCONNX consente di specificare opzioni per controllare il funzionamento della chiamata.

Su IBM MQ for Multiplatforms, ogni thread in un'applicazione può connettersi a gestori code differenti. Su altri sistemi, tutte le connessioni simultanee in un processo devono essere allo stesso gestore code.

- "Sintassi" a pagina 1317
- "Parametri" a pagina 1318
- "Dichiarazione RPG" a pagina 1318

Sintassi

(QMNAME, CNOPT, HCONN, CMPCOD, REASON) MQCONNX

Parametri

La chiamata MQCONNX presenta i parametri seguenti:

QMNAME (stringa di caratteri a 48 byte) - input

Il nome del gestore code.

Per i dettagli, consultare il parametro **QMNAME** descritto in “MQCONN (gestore code di connessione) su IBM i” a pagina 1313 .

CNOPT (MQCNO) - input/output

Opzioni che controllano l'azione di MQCONNX.

Vedi “MQCNO (Opzioni di connessione) su IBM i” a pagina 1075 per i dettagli.

HCONN (numero intero con segno a 10 cifre) - output

Handle di connessione.

Per i dettagli, consultare il parametro **HCONN** descritto in “MQCONN (gestore code di connessione) su IBM i” a pagina 1313 .

CMPCOD (numero intero con segno a 10 cifre) - output

Codice di completamento.

Per i dettagli, consultare il parametro **CMPCOD** descritto in “MQCONN (gestore code di connessione) su IBM i” a pagina 1313 .

REASON (numero intero con segno a 10 cifre) - output

Codice di errore *CMPCOD*.

Consultare il parametro **REASON** descritto in “MQCONN (gestore code di connessione) su IBM i” a pagina 1313 per dettagli sui possibili codici di errore.

I seguenti codici motivo aggiuntivi possono essere restituiti dalla chiamata MQCONNX:

Se *CMPCOD* è CCFAIL:

RC2278
(2278, X'8E6') Campi di collegamento client non validi.

RC2139
(2139, X'85B') Struttura delle opzioni di connessione non valida.

RC2046
(2046, X'7FE') Opzioni non valide o non congruenti.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                   CALLP     MQCONN(QMNAME : HCONN : CMPCOD :
C                                    REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQCONN           PR                  EXTPROC('MQCONN')
D* Name of queue manager
D QMNAME                         48A
D* Options that control the action of MQCONNX
D HCONN                         224A
D* Connection handle
D HCONN                          10I 0
D* Completion code
D CMPCOD                         10I 0
```

```
     D* Reason code qualifying CMPCOD
     D REASON                        10I 0
```

IBM i MQCRTMH (Create message handle) su IBM i

La chiamata MQCRTMH restituisce un handle del messaggio.

Un'applicazione può utilizzarlo nelle successive chiamate di accodamento dei messaggi:

- Utilizzare la chiamata MQSETMP per impostare una proprietà dell'handle del messaggio.
- Utilizzare la chiamata MQINQMP per esaminare il valore di una proprietà dell'handle del messaggio.
- Utilizzare la chiamata MQDLTMP per eliminare una proprietà dell'handle del messaggio.

L'handle del messaggio può essere utilizzato nelle chiamate MQPUT e MQPUT1 per associare le proprietà dell'handle del messaggio alle proprietà del messaggio da inserire. Allo stesso modo, specificando un handle del messaggio sulla chiamata MQGET, è possibile accedere alle proprietà del messaggio richiamato utilizzando l'handle del messaggio quando la chiamata MQGET viene completata.

Utilizzare MQDLTMH per eliminare l'handle del messaggio.

- "Sintassi" a pagina 1319
- "Parametri" a pagina 1319
- "Dichiarazione RPG" a pagina 1321

Sintassi

MQCRTMH *(Hconn, CrtMsgHOpts, Hmsg, CompCode, Reason)*

Parametri

La chiamata MQCRTMH ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* è stato restituito da una chiamata MQCONN o MQCONNX precedente. Se la connessione al gestore code cessa di essere valida e nessuna chiamata IBM MQ è in esecuzione sull'handle del messaggio, MQDLTMH viene richiamato implicitamente per eliminare il messaggio.

In alternativa, è possibile specificare il valore seguente:

HCUNAS
L'handle di connessione non rappresenta una connessione ad alcun particolare gestore code.

Quando si utilizza questo valore, l'handle del messaggio deve essere eliminato con una chiamata esplicita a MQDLTMH per rilasciare qualsiasi memoria ad esso assegnata; IBM MQ non elimina mai implicitamente l'handle del messaggio.

Deve essere presente almeno una connessione valida a un gestore code stabilito sul thread che crea l'handle del messaggio, altrimenti la chiamata non riesce con RC2018.

CRTOPT (MQCMHO) - immissione

Le opzioni che controllano l'azione di MQCRTMH. Consultare MQCMHO per i dettagli.

HMSG (numero intero con segno a 20 cifre) - output

Nell'output viene restituito un handle del messaggio che può essere utilizzato per impostare, analizzare ed eliminare le proprietà dell'handle del messaggio. Inizialmente l'handle del messaggio non contiene proprietà.

Un handle del messaggio ha anche un descrittore del messaggio associato. Inizialmente questo descrittore di messaggi contiene i valori predefiniti. I valori dei campi del descrittore del messaggio

associato possono essere impostati e interrogati utilizzando le chiamate MQSETMP e MQINQMP. La chiamata MQDLTMP reimposta un campo del descrittore del messaggio sul valore predefinito.

Se il parametro *HCONN* viene specificato come valore HCUNAS, l'handle del messaggio restituito può essere utilizzato sulle chiamate MQGET, MQPUT o MQPUT1 con qualsiasi connessione all'interno dell'unità di elaborazione, ma può essere utilizzato da una sola chiamata IBM MQ alla volta. Se l'handle è in uso quando una seconda chiamata IBM MQ tenta di utilizzare lo stesso handle del messaggio, la seconda chiamata IBM MQ ha esito negativo con codice di errore RC2499.

Se il parametro *HCONN* non è HCUNAS, l'handle del messaggio restituito può essere utilizzato solo sulla connessione specificata.

Lo stesso valore del parametro *HCONN* deve essere utilizzato nelle successive chiamate MQI in cui viene utilizzato questo handle del messaggio:

- MQDLTMH
- MQSETMP
- MQINQMP
- MQDLTMP
- MQMHBUF
- MQBUFMH

L'handle del messaggio restituito cessa di essere valido quando viene emessa la chiamata MQDLTMH per l'handle del messaggio o quando termina l'unità di elaborazione che definisce l'ambito dell'handle. MQDLTMH viene richiamato implicitamente se viene fornita una connessione specifica quando viene creato l'handle del messaggio e la connessione al gestore code cessa di essere valida, ad esempio, se viene richiamato MQDBC.

CMPCOD (numero intero con segno a 10 cifre) - output

Il codice di completamento; è uno dei seguenti:

CCOK
Completamento con esito positivo.

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Il codice di errore che qualifica *CMPCOD*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCFAIL:

RC2204
(2204, X'089C') Adattatore non disponibile.

RC2130
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

RC2157
(2157, X'86D') Gli ASID principale e home differiscono.

RC2219
(2219, X'08AB') Chiamata MQI immessa prima del completamento della chiamata precedente.

RC2461
(2461, X'099D') Struttura delle opzioni di gestione del messaggio non valida.

RC2273
(2273, X'7D9') Connessione al gestore code persa.

RC2017
(2017, X'07E1') Nessun ulteriore handle disponibile.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2460
(2460, X'099C') Il puntatore della gestione messaggi non è valido.

RC2046
(2046, X'07FE') Opzioni non valide o non congruenti.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

Per ulteriori dettagli, vedere “Codici di ritorno per IBM i (ILE RPG)” a pagina 1467.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCRTMH(HCONN : CRTOPT : HMSG :
                                        CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
DMQCRTMH          PR                  EXTPROC('MQCRTMH')
D* Connection handle
D HCONN                           10I 0 VALUE
D* Options that control the action of MQCRTMH
D CRTOPT                          12A
D* Message handle
D HMSG                            20I 0
D* Completion code
D CMPCOD                          10I 0
D* Reason code qualifying CompCode
D REASON                          10I 0
```

IBM i MQCTL (Controllo callback) su IBM i

La chiamata MQCTL esegue azioni di controllo sugli handle di oggetto aperti per una connessione.

Sintassi

(Hconn, Operation, ControlOpts, CompCode, Reason) MQCTL

Note d'utilizzo

1. Le routine di callback devono controllare le risposte da tutti i servizi richiamati e, se la routine rileva una condizione che non può essere risolta, deve emettere un comando MQCB (CBREG) per impedire chiamate ripetute alla routine di callback.

Parametri

La chiamata MQCTL presenta i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

OPERATN (numero intero con segno a 10 cifre) - input

L'operazione in fase di elaborazione sul callback definito per l'handle oggetto specificato. È necessario specificare solo una delle seguenti opzioni:

RCTL

Avviare l'utilizzo dei messaggi per tutte le funzioni del consumatore di messaggi definite per l'handle di connessione specificato.

I callback vengono eseguiti su un thread avviato dal sistema, che è diverso da qualsiasi thread dell'applicazione.

Questa operazione fornisce il controllo dell'handle di connessione fornito al sistema. Le uniche chiamate MQI che possono essere emesse da un thread diverso da quello consumer sono:

- MQCTL con operazione CTLSP
- MQCTL con operazione CTLSU
- MQDISC - Esegue MQCTL con l'operazione CTLSP prima di disconnettere HConn.

RC2500 viene restituito se viene emessa una chiamata API IBM MQ durante l'avvio dell'handle di connessione e la chiamata non ha origine da una funzione del consumer del messaggio.

Se una connessione non riesce, questa operazione interrompe la conversazione il più presto possibile. È quindi possibile che una chiamata API IBM MQ emessa sul thread principale riceva il codice di ritorno RC2500 per un certo periodo di tempo, seguito dal codice di ritorno RC2009 quando la connessione ritorna allo stato arrestato.

Questo può essere emesso in una funzione consumer. Per la stessa connessione della routine di callback, il solo scopo è quello di annullare un'operazione CTLSP precedentemente emessa.

Questa opzione non è supportata se l'applicazione è collegata a una libreria IBM MQ senza sottoprocessi.

CTLSW

Avviare l'utilizzo dei messaggi per tutte le funzioni del consumatore di messaggi definite per l'handle di connessione specificato.

I destinatari dei messaggi vengono eseguiti sullo stesso thread e il controllo non viene restituito al chiamante di MQCTL fino a quando:

- Rilasciato dall'utilizzo delle operazioni MQCTL CTLSP o CTLSU, oppure
- Tutte le routine consumer sono state annullate o sospese.

Se tutti i consumer vengono annullati o sospesi, viene emessa un'operazione CTLSP implicita.

Questa opzione non può essere utilizzata dall'interno di una routine di callback, per l'handle di connessione corrente o per qualsiasi altro handle di connessione. Se la chiamata viene tentata, viene restituita con RC2012.

Se, in qualsiasi momento durante un'operazione CTLSW, non vi sono consumer registrati e non sospesi, la chiamata ha esito negativo con un codice di errore RC2446.

Se, durante un'operazione CTLSW, la connessione viene sospesa, la chiamata MQCTL restituisce un codice motivo di avvertenza di RC2521; la connessione rimane 'avviata'.

L'applicazione può scegliere di emettere CTLSP o CTLRE. In questa istanza, l'operazione CTLRE si blocca.

Questa opzione non è supportata in un client a thread singolo.

CTLSP
Arrestare l'utilizzo dei messaggi e attendere che tutti i consumer completino le operazioni prima del completamento di questa opzione. Questa operazione rilascia l'handle di connessione.

Se emessa dall'interno di una routine di callback, questa opzione non diventa effettiva fino a quando la routine non termina. Non vengono richiamate ulteriori routine del destinatario del messaggio dopo che sono state completate le routine del destinatario per i messaggi già letti e dopo che sono state effettuate le chiamate di arresto (se richieste) alle routine di richiamata.

Se emesso al di fuori di una routine di callback, il controllo non ritorna al chiamante fino a quando non sono state completate le routine consumer per i messaggi già letti e dopo che sono state effettuate le chiamate di arresto (se richieste) ai callback. I callback stessi, tuttavia, rimangono registrati.

Questa funzione non ha alcun effetto sui messaggi di lettura anticipata. È necessario assicurarsi che i consumer eseguano MQCLOSE (COQSC), dall'interno della funzione di callback, per determinare se sono disponibili ulteriori messaggi da consegnare.

CTLSU
Sospendere l'utilizzo dei messaggi. Questa operazione rilascia l'handle di connessione.

Ciò non influisce sulla lettura anticipata dei messaggi per l'applicazione. Se si intende interrompere l'utilizzo dei messaggi per un lungo periodo, considerare la chiusura della coda e riaprirla quando il consumo deve continuare.

Se emesso dall'interno di una routine di callback, non diventa effettivo fino a quando la routine non esce. Non verranno richiamate ulteriori routine del destinatario del messaggio dopo l'uscita della routine corrente.

Se emesso all'esterno di un callback, il controllo non ritorna al chiamante fino a quando non viene completata la routine consumer corrente e non vengono richiamate altre.

CTLRE
Riprendere l'utilizzo dei messaggi.

Questa opzione viene normalmente emessa dal thread dell'applicazione principale, ma può essere utilizzata anche dall'interno di una routine di callback per annullare una precedente richiesta di sospensione emessa nella stessa routine.

Se CTLRE viene utilizzato per riprendere un CTLSW, l'operazione si blocca.

PCTLOP (MQCTLO) - input

Opzioni che controllano l'azione di MQCTL

Consultare MQCTLO per i dettagli della struttura.

CMPCOD (numero intero con segno a 10 cifre) - output

Il codice di completamento; è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

I seguenti codici motivo sono quelli che il gestore code può restituire per il parametro **Reason** .

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCFAIL:

RC2133
(2133, X'855 ') Impossibile caricare i moduli dei servizi di conversione dati.

RC2204
(2204, X'89C') Adattatore non disponibile.

RC2130
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

RC2374
(2374, X' 946 ') Uscita API non riuscita.

RC2183
(2183, X'887 ') Impossibile caricare l'uscita API.

RC2157
(2157, X'86D') Gli ASID principale e home differiscono.

RC2005
(2005, X'7D5') Parametro di lunghezza del buffer non valido.

RC2487
(2487, X'9B7') Impossibile richiamare la routine di callback

RC2448
(2448, X' 990 ') Impossibile annullare la registrazione, sospendere o riprendere perché non è presente alcun callback registrato

RC2486
(2486, X'9B6') Sia CallbackFunction che CallbackName sono stati specificati in una chiamata CBREG oppure è stato specificato uno tra CallbackFunction o CallbackName ma non corrisponde alla funzione di callback attualmente registrata.

RC2483
(2483, X'9B3') Campo tipo CallBacknon corretto.

RC2219
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

RC2444
(2444, X'98C') Il blocco di opzione è errato.

RC2484
(2484, X'9B4') Campo di opzioni MQCBD non corretto.

RC2140
(2140, X'85C') Richiesta di attesa rifiutata da CICS.

RC2009
(2009, X'7D9') Connessione al gestore code persa.

RC2217
(2217, X'8A9') Non autorizzato per la connessione.

RC2202
(2202, X'89A') Connessione in fase di sospensione.

RC2203
(2203, X'89B') Chiusura della connessione.

RC2207
(2207, X'89F') Errore identificativo di correlazione.

RC2016
(2016, X'7E0') Ottiene inibiti per la coda.

RC2351
(2351, X'92F') Unità globali di conflitto di lavoro.

RC2186
(2186, X'88A') Struttura delle opzioni Get - message non valida.

RC2353
(2353, X' 931 ') Handle in uso per l'unità di lavoro globale.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2019
(2019, X'7E3') Handle oggetto non valido.

RC2259
(2259, X'8D3') Specifica di ricerca incongruente.

RC2245
(2245, X'8C5') Specifica dell'unità di lavoro non congruente.

RC2246
(2246, X'8C6') Messaggio sotto il cursore non valido per il recupero.

RC2352
(2352, X' 930 ') L'unità di lavoro globale è in conflitto con l'unità di lavoro locale.

RC2247
(2247, X'8C7') Opzioni di corrispondenza non valide.

RC2485
(2485, X'9B5') Campo di lunghezza MaxMsgnon corretto

RC2026
(2026, X'7EA') Descrittore messaggio non valido.

RC2497
(2497, X'9C1') Il punto di ingresso funzione specificato non è stato trovato nel modulo.

RC2496
(2496, X'9C0') Il modulo è stato trovato ma è del tipo errato (32 bit o 64 bit) o non è una dll valida.

RC2495
(2495, X'9BF') Modulo non trovato nel percorso di ricerca o non autorizzato al caricamento.

RC2206
(2206, X'89E') Errore identificativo messaggio.

RC2250
(2250, X'8CA') Numero di sequenza messaggio non valido.

RC2331
(2331, X'91B') L'utilizzo del token del messaggio non è valido.

RC2036
(2036, X'7F4') Coda non aperta per la ricerca.

RC2037
(2037, X'7F5') Coda non aperta per l'input.

RC2041
(2041, X'7F9') Definizione oggetto modificata dall'apertura.

RC2101
(2101, X'835 ') Oggetto danneggiato.

RC2488
(2488, X'9B8') Codice operazione non corretto nella chiamata API

RC2046
(2046, X'7FE') Opzioni non valide o non congruenti.

RC2193
(2193, X'891 ') Errore durante l'accesso al dataset della serie di pagine.

RC2052
(2052, X'804 ') La coda è stata eliminata.

RC2394
(2394, X'95A') La coda ha un tipo di indice errato.

RC2058
(2058, X'80A') Nome gestore code non valido o sconosciuto.

RC2059
(2059, X'80B') Gestore code non disponibile per la connessione.

RC2161
(2161, X'871 ') Gestore code in fase di sospensione.

RC2162
(2162, X'872 ') Chiusura del gestore code.

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2069
(2069, X'815 ') Segnale eccezionale per questa maniglia.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2109
(2109, X'83D') Chiamata eliminata dal programma di uscita.

RC2072
(2072, X'818 ') Supporto punto di sincronizzazione non disponibile.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

RC2354
(2354, X' 932 ') L'inserimento nell'unità di lavoro globale non è riuscito.

RC2355
(2355, X' 933 ') La miscelazione delle chiamate UOW non è supportata.

RC2255
(2255, X'8CF') Unità di lavoro non disponibile per il gestore code.

RC2090
(2090, X'82A') Intervallo di attesa in MQGMO non valido.

RC2256
(2256, X'8D0') Versione errata di MQGMO fornita.

RC2257
(2257, X'8D1') Versione non corretta di MQMD fornita.

RC2298
(2298, X'8FA') La funzione richiesta non è disponibile nell'ambiente corrente.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCTL(HCONN : OPERATN : PCTLOP :
                                      CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
DMQCTL            PR                  EXTPROC('MQCTL')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Operation
D OPERATN                       10I 0 VALUE
D* Control options
D PCTLOP                        32A
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CompCode
D REASON                        10I 0
```

IBM i MQDISC (Disconnetti gestore code) su IBM i

La chiamata MQDISC interrompe la connessione tra il gestore code e il programma applicativo ed è l'inverso della chiamata MQCONN o MQCONNX.

Sintassi

MQDISC *(HCONN, CMPCOD, REASON)*

Note d'utilizzo

1. Se viene emessa una chiamata MQDISC quando l'applicazione ha ancora oggetti aperti, tali oggetti vengono chiusi dal gestore code, con le opzioni di chiusura impostate su CONONE.
2. Se l'applicazione termina con modifiche di cui non è stato eseguito il commit in un'unità di lavoro, la disposizione di tali modifiche dipende dal modo in cui termina l'applicazione:
   a. Se l'applicazione emette la chiamata MQDISC prima di terminare:
      - Per un'unità di lavoro coordinata del gestore code, il gestore code emette la chiamata MQCMIT per conto dell'applicazione. Se possibile, viene eseguito il commit dell'unità di lavoro e, in caso contrario, viene eseguito il backout.
      - Per un'unità di lavoro coordinata esternamente, non c'è alcuna variazione nello stato dell'unità di lavoro; tuttavia, il gestore code indicherà che l'unità di lavoro deve essere sottoposta a commit, quando richiesto dal coordinatore dell'unità di lavoro.
   b. Se l'applicazione termina normalmente ma non emette la chiamata MQDISC, viene eseguito il backout dell'unità di lavoro.
   c. Se l'applicazione termina *in modo anomalo* senza emettere la chiamata MQDISC, viene eseguito il backout dell'unità di lavoro.

Parametri

La chiamata MQDISC ha i seguenti parametri:

HCONN (numero intero con segno a 10 cifre) - input/output

Handle di connessione.

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

Una volta completata correttamente la chiamata, il gestore code imposta *HCONN* su un valore che non è un handle valido per l'ambiente. Questo valore è:

HCUNUH
Handle di connessione inutilizzabile.

CMPCOD (numero intero con segno a 10 cifre) - output

Codice di completamento.

Il valore è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Codice di errore *CMPCOD*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCFAIL:

RC2219
(2219, X'8AB') Chiamata MQI reimmessa prima del completamento della chiamata precedente.

RC2009
(2009, X'7D9') Connessione al gestore code persa.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2058
(2058, X'80A') Nome gestore code non valido o sconosciuto.

RC2059
(2059, X'80B') Gestore code non disponibile per la connessione.

RC2162
(2162, X'872 ') Chiusura del gestore code.

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                   CALLP     MQDISC(HCONN : CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQDISC           PR                  EXTPROC('MQDISC')
D* Connection handle
D HCONN                         10I 0
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CMPCOD
D REASON                        10I 0
```

IBM i MQDLTMH (Elimina handle del messaggio) su IBM i

La chiamata MQDLTMH elimina un handle del messaggio ed è l'inverso della chiamata MQCRTMH.

- “Sintassi” a pagina 1329
- “Note d'utilizzo” a pagina 1329
- “Parametri” a pagina 1330

- “Dichiarazione RPG” a pagina 1331

Sintassi

MQDLTMH *((Hconn, Hmsg, DltMsgHOpts, CompCode, Reason)*

Note d'utilizzo

1. È possibile utilizzare questa chiamata solo quando il gestore code stesso coordina l'unità di lavoro. Questo può essere:
   - Un'unità di lavoro locale, in cui le modifiche influenzano solo le risorse IBM MQ .
   - Un'unità di lavoro globale, in cui le modifiche possono influire sulle risorse appartenenti ad altri gestori risorse, nonché sulle risorse IBM MQ .

   Per ulteriori dettagli sulle unità di lavoro locali e globali, consultare “MQBEGIN (Inizio unità di lavoro) su IBM i” a pagina 1290.
2. In ambienti in cui il gestore code non coordina l'unità di lavoro, utilizzare la chiamata di backout appropriata invece di MQBACK. L'ambiente potrebbe anche supportare un backout implicito causato dalla chiusura anomala dell'applicazione.
   - Su z/OS, utilizzare le seguenti chiamate:
     - I programmi batch (inclusi i programmi IMS batch DL/I) possono utilizzare la chiamata MQBACK se l'unità di lavoro influenza solo le risorse IBM MQ . Tuttavia, se l'unità di lavoro influisce sia sulle risorse IBM MQ che sulle risorse appartenenti ad altri gestori risorse (ad esempio, Db2), utilizzare la chiamata SRRBACK fornita da RRS (Recoverable Resource Service) z/OS . La chiamata SRRBACK ripristina le modifiche alle risorse appartenenti ai gestori risorse che sono state abilitate per il coordinamento RRS.
     - Le applicazioni CICS devono utilizzare il comando `EXEC CICS SYNCPOINT ROLLBACK` per eseguire il backout dell'unità di lavoro. Non utilizzare la chiamata MQBACK per applicazioni CICS .
     - Le applicazioni IMS (diverse dai programmi DL/I batch) devono utilizzare chiamate IMS come `ROLB` per eseguire il backout dell'unità di lavoro. Non utilizzare la chiamata MQBACK per applicazioni IMS (diverse dai programmi DL/I batch).
   - Su IBM i, utilizzare questa chiamata per le unità di lavoro locali coordinate dal gestore code. Ciò significa che una definizione di commit non deve esistere a livello di lavoro, ovvero il comando `STRCMTCTL` con il parametro **`CMTSCOPE(*JOB)`** non deve essere stato immesso per il lavoro.
3. Se un'applicazione termina con modifiche non sottoposte a commit in un'unità di lavoro, la disposizione di tali modifiche dipende dal fatto che l'applicazione termini normalmente o in modo anomalo. Consultare le note sull'utilizzo in “MQDISC (Disconnetti gestore code) su IBM i” a pagina 1327 per ulteriori dettagli.
4. Quando un'applicazione inserisce o richiama i messaggi in gruppi o segmenti di messaggi logici, il gestore code conserva le informazioni relative al gruppo di messaggi e al messaggio logico per le ultime chiamate MQPUT e MQGET riuscite. Queste informazioni sono associate all'handle della coda e includono:
   - I valore dei campi *GroupId*, *MsgSeqNumber*, *Offset*e *MsgFlags* in MQMD.
   - Se il messaggio fa parte di un'unità di lavoro.
   - Per la chiamata MQPUT: se il messaggio è persistente o non persistente.

   Il gestore code conserva tre serie di informazioni di gruppi e segmenti, una per ciascuno dei seguenti:
   - L'ultima chiamata MQPUT riuscita (può far parte di un'unità di lavoro).
   - L'ultima chiamata MQGET riuscita che ha rimosso un messaggio dalla coda (può far parte di un'unità di lavoro).
   - L'ultima chiamata MQGET riuscita che ha visualizzato un messaggio sulla coda (non può far parte di un'unità di lavoro).

Se l'applicazione inserisce o richiama i messaggi come parte di un'unità di lavoro, e l'applicazione esegue il backout dell'unità di lavoro, le informazioni sul gruppo e sul segmento vengono ripristinate sul valore che aveva in precedenza:

- Le informazioni associate alla chiamata MQPUT vengono ripristinate al valore che aveva prima della prima chiamata MQPUT riuscita per tale handle di coda nell'unità di lavoro corrente.
- Le informazioni associate alla chiamata MQGET vengono ripristinate al valore che aveva prima della prima chiamata MQGET riuscita per tale handle di coda nell'unità di lavoro corrente.

Le code che sono state aggiornate dall'applicazione dopo l'avvio dell'unità di lavoro, ma al di fuori dell'ambito dell'unità di lavoro, non hanno le relative informazioni sul gruppo e sul segmento ripristinate se viene eseguito il backout dell'unità di lavoro.

Il ripristino delle informazioni sul gruppo e sul segmento al suo valore precedente quando viene eseguito il backout di un'unità di lavoro consente all'applicazione di distribuire un gruppo di messaggi di grandi dimensioni o un messaggio logico di grandi dimensioni costituito da molti segmenti in diverse unità di lavoro e di riavviare nel punto corretto nel gruppo di messaggi o nel messaggio logico in caso di errore di una delle unità di lavoro. L'utilizzo di diverse unità di lavoro potrebbe essere vantaggioso se il gestore code locale ha solo una memoria di coda limitata. Tuttavia, l'applicazione deve conservare informazioni sufficienti per essere in grado di riavviare l'inserimento o il richiamo dei messaggi nel punto corretto se si verifica un errore di sistema.

Per i dettagli su come riavviare il sistema nel punto corretto dopo un malfunzionamento del sistema, consultare l'opzione PMLOGO descritta in PMOPT (10 cifre intere con segno)e l'opzione GMLOGO descritta in GMOPT (10 cifre intere con segno).

Le note di uso rimanenti si applicano solo quando il gestore code coordina le unità di lavoro:

5. Un'unità di lavoro ha lo stesso ambito di un handle di connessione. Tutte le chiamate IBM MQ che interessano una particolare unità di lavoro devono essere eseguite utilizzando lo stesso handle di connessione. Le chiamate emesse utilizzando un handle di collegamento differente (ad esempio, le chiamate emesse da un'altra applicazione) influenzano un'unità di lavoro diversa. Consultare HCONN (numero intero con segno a 10 cifre) - output per informazioni sull'ambito degli handle di connessione.
6. Solo i messaggi inseriti o richiamati come parte dell'unità di lavoro corrente vengono influenzati da questa chiamata.
7. Un'applicazione di lunga durata che emette chiamate MQGET, MQPUT o MQPUT1 all'interno di un'unità di lavoro, ma che non emette mai una chiamata di commit o di backout, può riempire le code con messaggi che non sono disponibili per altre applicazioni. Per evitare questa possibilità, l'amministratore deve impostare l'attributo del gestore code **MaxUncommittedMsgs** su un valore sufficientemente basso per evitare che le applicazioni runaway riempiano le code, ma abbastanza alto per consentire il corretto funzionamento delle applicazioni di messaggistica previste.

Parametri

La chiamata MQDLTMH presenta i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Questo handle rappresenta la connessione al gestore code.

Il valore deve corrispondere all'handle di connessione utilizzato per creare l'handle del messaggio specificato nel parametro **HMSG** .

Se l'handle del messaggio è stato creato utilizzando HCUNAS, è necessario stabilire una connessione valida sul thread che elimina l'handle del messaggio, altrimenti la chiamata non riesce con RC2009 .

HMSG (numero intero con segno a 20 cifre) - input/output

Questo è l'handle del messaggio da eliminare. Il valore è stato restituito da una precedente chiamata MQCRTMH.

Una volta completata correttamente la chiamata, l'handle viene impostato su un valore non valido per l'ambiente. Questo valore è:

HMUNUH
Gestore messaggi inutilizzabile.

L'handle del messaggio non può essere eliminato se è in corso un'altra chiamata IBM MQ che ha passato lo stesso handle del messaggio.

DLTOPT (MQDMHO) - immissione

Consultare MQDMHO per i dettagli.

CMPCOD (numero intero con segno a 10 cifre) - output

Il codice di completamento; è uno dei seguenti:

CCOK
Completamento con esito positivo.

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Il codice di errore che qualifica *CMPCOD*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCFAIL:

RC2204
(2204, X'089C') Adattatore non disponibile.

RC2130
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

RC2157
(2157, X'86D') Gli ASID principale e home differiscono.

RC2219
(2219, X'08AB') Chiamata MQI immessa prima del completamento della chiamata precedente.

RC2009
(2009, X'07D9') Connessione al gestore code persa.

RC2462
(2462, X'099E') Struttura delle opzioni di eliminazione dell'handle del messaggio non valida.

RC2460
(2460, X'099C') Il puntatore della gestione messaggi non è valido.

RC2499
(2499, X'09C3') handle del messaggio già in uso.

RC2046
(2046, X'07FE') Opzioni non valide o non congruenti.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

Per ulteriori dettagli, vedere “Codici di ritorno per IBM i (ILE RPG)” a pagina 1467.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQDLTMH(HCONN : HMSG : DLTOPT :
                                   CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
     DMQDLTMH          PR                  EXTPROC('MQDLTMH')
     D* Connection handle
     D HCONN                        10I 0 VALUE
     D* Message handle
     D HMSG                         20I 0
     D* Options that control the action of MQDLTMH
     D DLTOPT                       12A
     D* Completion code
     D CMPCOD                       10I 0
     D* Reason code qualifying CompCode
     D REASON                       10I 0
```

MQDLTMP - Proprietà Elimina messaggio

La chiamata MQDLTMP elimina una proprietà da un handle del messaggio ed è l'inverso della chiamata MQSETMP.

Sintassi

MQDLTMP *(Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason)*

Parametri

La chiamata MQDLTMP ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - Input

Questo handle rappresenta la connessione al gestore code. Il valore deve corrispondere all'handle di connessione utilizzato per creare l'handle del messaggio specificato nel parametro **HMSG** .

Se la gestione del messaggio è stata creata utilizzando HCUNAS, è necessario stabilire una connessione valida sul thread che elimina la gestione del messaggio, altrimenti la chiamata non riesce con RC2009.

HMSG (numero intero con segno a 20 cifre) - input

Questo è l'handle del messaggio contenente la proprietà da eliminare. Il valore è stato restituito da una precedente chiamata MQCRTMH.

DLTOPT (MQDMPO) - Immissione

Consultare il tipo di dati MQDMPO per i dettagli.

PRNAME (MQCHARV) - input

Il nome della proprietà da eliminare. Consultare Nomi proprietà per ulteriori informazioni sui nomi proprietà.

I caratteri jolly non sono consentiti nel nome proprietà.

CMPCOD (numero intero con segno a 10 cifre) - output

Il codice di completamento; è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Il codice di errore che qualifica *CMPCOD*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCWARN:

RC2471
(2471, X'09A7') Proprietà non disponibile.

RC2421
(2421, X'0975 ') Impossibile analizzare una cartella MQRFH2 contenente le proprietà.

Se *CMPCOD* è CCFAIL:

RC2204
(2204, X'089C') Adattatore non disponibile.

RC2130
(2130, X'0852 ') Impossibile caricare il modulo di servizio adattatore.

RC2157
(2157, X'086D') Gli ASID principale e principale differiscono.

RC2219
(2219, X'08AB') Chiamata MQI immessa prima del completamento della chiamata precedente.

RC2009
(2009, X'07D9') Connessione al gestore code persa.

RC2481
(2481, X'09B1') Struttura delle opzioni di eliminazione della proprietà del messaggio non valida.

RC2460
(2460, X'099C') Gestione messaggio non valida.

RC2499
(2499, X'09C3') handle del messaggio già in uso.

RC2046
(2046, X'07FE') Opzioni non valide o non congruenti.

RC2442
(2442, X'098A') Nome proprietà non valido.

RC2111
(2111, X'083F') Identificativo serie di caratteri codificato del nome proprietà non valido.

RC2195
(2195, X'0893 ') Si è verificato un errore non previsto.

Per ulteriori informazioni su questi codici, vedi Codici di motivo e di completamento API.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQDLTMP(HCONN : HMSG : DLTOPT :
                                   PRNAME : CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
DMQDLTMP          PR                  EXTPROC('MQDLTMP')
D* Connection handle
D HCONN                           10I 0 VALUE
D* Message handle
D HMSG                            20I 0 VALUE
D* Options that control the action of MQDLTMP
```

```
     D DLTOPT                        12A
     D* Property name
     D PRNAME                        32A
     D* Completion code
     D CMPCOD                        10I 0
     D* Reason code qualifying CompCode
     D REASON                        10I 0
```

IBM i MQGET (Richiama messaggio) su IBM i

La chiamata MQGET richiama un messaggio da una coda locale che è stata aperta utilizzando la chiamata MQOPEN.

Sintassi

MQGET *(HCONN, HOBJ, MSGDSC, GMO, BUFLEN, BUFFER, DATLEN, CMPCOD, REASON)*

Note d'utilizzo

1. Il messaggio richiamato viene normalmente eliminato dalla coda. Questa eliminazione può verificarsi come parte della chiamata MQGET stessa o come parte di un punto di sincronizzazione. L'eliminazione del messaggio non si verifica se viene specificata un'opzione GMBRWF o GMBRWN sul parametro **GMO** (consultare il campo *GMOPT* descritto in "MQGMO (opzioni Get - message) su IBM i" a pagina 1105).
2. Se l'opzione GMLK viene specificata con una delle opzioni di ricerca, il messaggio visualizzato viene bloccato in modo che sia visibile solo a questo handle.

   Se viene specificata l'opzione GMUNLK, viene sbloccato un messaggio precedentemente bloccato. In questo caso, non viene richiamato alcun messaggio e i parametri **MSGDSC**, **BUFLEN**, **BUFFER** e **DATLEN** non vengono controllati o modificati.
3. Se l'applicazione che emette la chiamata MQGET è in esecuzione come un IBM MQ MQI client, è possibile che il messaggio richiamato venga perso se durante l'elaborazione della chiamata MQGET il IBM MQ MQI client termina in modo anomalo o la connessione client viene interrotta. Ciò si verifica perché il surrogato in esecuzione sulla piattaforma del gestore code e che emette la chiamata MQGET per conto del client non può rilevare la perdita del client fino a quando il surrogato sta per restituire il messaggio al client; ciò avviene dopo che il messaggio è stato rimosso dalla coda. Ciò può verificarsi sia per i messaggi persistenti che per quelli non persistenti.

   Il rischio di perdere i messaggi in questo modo può essere eliminato richiamando sempre i messaggi all'interno delle unità di lavoro (ossia, specificando l'opzione GMSYP nella chiamata MQGET e utilizzando le chiamate MQCMIT o MQBACK per eseguire il commit o il backout dell'unità di lavoro quando l'elaborazione del messaggio è completa). Se viene specificato GMSYP e il client termina in maniera anomala o la connessione viene interrotta, il surrogato esegue il backout dell'unità di lavoro sul gestore code e il messaggio viene reintegrato nella coda.

   In linea di principio, la stessa situazione può verificarsi con le applicazioni in esecuzione sulla piattaforma del gestore code, ma in questo caso la finestra durante la quale un messaggio può essere perso è piccola. Tuttavia, come con IBM MQ MQI clients , il rischio può essere eliminato recuperando il messaggio all'interno di un'unità di lavoro.
4. Se un'applicazione inserisce una sequenza di messaggi su un particolareall'interno di una singola unità di lavoro e quindi esegue il commit di tale unità di lavoro correttamente, i messaggi diventano disponibili per il richiamo nel modo seguente:

- Se la coda è una *coda non condivisa* (ossia, una coda locale), tutti i messaggi all'interno dell'unità di lavoro diventano disponibili contemporaneamente.
- Se la coda è una *coda condivisa*, i messaggi all'interno dell'unità di lavoro diventano disponibili nell'ordine in cui sono stati inseriti, ma non tutti contemporaneamente. Quando il sistema ha un carico elevato, è possibile che il primo messaggio nell'unità di lavoro venga richiamato correttamente, ma che la chiamata MQGET per il secondo o il successivo messaggio nell'unità di lavoro abbia esito negativo con RC2033. Se ciò si verifica, l'applicazione deve attendere un breve periodo e riprovare l'operazione.

5. Se un'applicazione inserisce una sequenza di messaggi nella stessa coda senza utilizzare i gruppi di messaggi, l'ordine di tali messaggi viene conservato se sono soddisfatte determinate condizioni. Per i dettagli, consultare le note di utilizzo nella descrizione della chiamata MQPUT. Se le condizioni sono soddisfatte, i messaggi vengono presentati all'applicazione ricevente nell'ordine in cui sono stati inviati, se:
   - Solo un destinatario riceve i messaggi dalla coda.

     Se ci sono due o più applicazioni che ricevono i messaggi dalla coda, devono concordare con il mittente il meccanismo da utilizzare per identificare i messaggi che appartengono a una sequenza. Ad esempio, il mittente potrebbe impostare tutti i campi MDCID nei messaggi in una sequenza su un valore univoco per tale sequenza di messaggi.
   - Il destinatario non modifica deliberatamente l'ordine di richiamo, ad esempio specificando un particolare MDMID o MDCID.

   Se l'applicazione mittente inserisce i messaggi come un gruppo di messaggi, i messaggi vengono presentati all'applicazione ricevente nell'ordine corretto se l'applicazione ricevente specifica l'opzione GMLOGO sulla chiamata MQGET. Per ulteriori informazioni sui gruppi di messaggi, consultare:
   - Campo MDMFL in MQMD
   - Opzione PMLOGO in MQPMO
   - Opzione GMLOGO in MQGMO
6. Test delle applicazioni per il codice di feedback FBQUIT nel campo MDFB del parametro **MSGDSC** . Se questo valore viene trovato, l'applicazione termina. Per ulteriori informazioni, consultare il campo MDFB descritto in “MQMD (Message Descriptor) su IBM i” a pagina 1140 .
7. Se la coda identificata da HOBJ è stata aperta con l'opzione OOSAVA e il codice di completamento dalla chiamata MQGET è CCOK o CCWARN, il contesto associato all'handle della coda HOBJ è impostato sul contesto del messaggio che è stato richiamato (a meno che non sia impostata l'opzione GMBRWF o GMBRWN, nel qual caso il contesto è contrassegnato come non disponibile). Questo contesto può essere utilizzato su una chiamata MQPUT o MQPUT1 successiva specificando le opzioni PMPASI o PMPASA. Ciò consente al contesto del messaggio ricevuto di essere trasferito in tutto o in parte a un altro messaggio (ad esempio, quando il messaggio viene inoltrato a un'altra coda). Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.
8. Se l'opzione GMCONV è inclusa nel parametro **GMO** , i dati del messaggio dell'applicazione vengono convertiti nella rappresentazione richiesta dall'applicazione ricevente, prima che i dati vengano inseriti nel parametro **BUFFER** :
   - Il campo MDFMT nelle informazioni di controllo nel messaggio identifica la struttura dei dati dell'applicazione e i campi MDCSI e MDENC nelle informazioni di controllo nel messaggio specificano l'identificativo della serie di caratteri e la codifica.
   - L'applicazione che emette la chiamata MQGET specifica nei campi MDCSI e MDENC nel parametro **MSGDSC** l'identificativo della serie di caratteri e la codifica in cui devono essere convertiti i dati del messaggio dell'applicazione.

   Quando la conversione dei dati del messaggio è necessaria, la conversione viene eseguita dal gestore code stesso o da un'uscita scritta dall'utente, a seconda del valore del campo MDFMT nelle informazioni di controllo nel messaggio:

- I seguenti formati vengono convertiti automaticamente dal gestore code; questi formati sono denominati formati "integrati":

MN FMADM	FMMDE
FMCICS	FMPCF
FMCMD1	MMRM
FMCMD2	FMRF
FMDLH	FMRFH2
FMDH	FMSTR
FMEVNT	FMTM
FMIMS	FMXQH
FMIMVS	

- Il nome formato FMNONE è un valore speciale che indica che la natura dei dati nel messaggio non è definita. Di conseguenza, il gestore code non tenta la conversione quando il messaggio viene richiamato dalla coda.

  Nota: Se GMCONV viene specificato sulla chiamata MQGET per un messaggio con un nome formato FMNONE e la serie di caratteri o la codifica del messaggio differiscono da quella specificata nel parametro **MSGDSC** , il messaggio viene ancora restituito nel parametro **BUFFER** (supponendo che non vi siano altri errori), ma la chiamata viene completata con il codice di completamento CCWARN e il codice di errore RC2110.

  FMNONE può essere utilizzato quando la natura dei dati del messaggio significa che non richiede la conversione o quando le applicazioni di invio e ricezione hanno concordato tra loro il formato in cui devono essere inviati i dati del messaggio.
- Tutti gli altri nomi di formato fanno sì che il messaggio venga passato ad un'uscita scritta dall'utente per la conversione. L'uscita ha lo stesso nome del formato, oltre alle aggiunte specifiche dell'ambiente. I nomi dei formati specificati dall'utente non devono iniziare con le lettere "MQ", poiché tali nomi potrebbero essere in conflitto con i nomi dei formati supportati in futuro.

I dati utente nel messaggio possono essere convertiti tra tutte le serie di caratteri e le codifiche supportate. Tuttavia, tenere presente che se il messaggio contiene una o più strutture di intestazione IBM MQ , il messaggio non può essere convertito da o in una serie di caratteri con caratteri a doppio byte o a più byte per uno qualsiasi dei caratteri validi nei nomi delle code. Il codice di errore RC2111 o RC2115 risulta se questo viene tentato e il messaggio viene restituito non convertito. La serie di caratteri Unicode UTF-16 è un esempio di tale serie di caratteri.

Al ritorno da MQGET, il seguente codice di errore indica che il messaggio è stato convertito correttamente:

- RCNONE

Il seguente codice di errore indica che il messaggio potrebbe essere stato convertito correttamente; l'applicazione deve controllare i campi `MDCSI` e `MDENC` nel parametro **MSGDSC** per scoprire:

- RC2079

Tutti gli altri codici di errore indicano che il messaggio non è stato convertito.

Nota: L'interpretazione del codice di errore descritto in questo esempio è vera per le conversioni eseguite dalle uscite scritte dall'utente solo se l'uscita è conforme alle linee guida di elaborazione.

9. Per i formati integrati elencati precedentemente, il gestore code potrebbe eseguire la conversione predefinita delle stringhe di caratteri nel messaggio quando viene specificata l'opzione GMCONV. La conversione predefinita consente al gestore code di utilizzare una serie di caratteri predefinita specificata dall'installazione che si avvicina alla serie di caratteri effettiva, durante la conversione dei dati stringa. Come risultato, la chiamata MQGET può avere esito positivo con codice di completamento CCOK, invece di completare con CCWARN e codice motivo RC2111 o RC2115.

Nota: Il risultato dell'utilizzo di una serie di caratteri approssimativa per convertire i dati stringa è che alcuni caratteri potrebbero essere convertiti in modo non corretto. Ciò può essere evitato utilizzando nella stringa solo i caratteri che sono comuni sia alla serie di caratteri effettiva che alla serie di caratteri predefinita.

La conversione predefinita si applica ai dati del messaggio dell'applicazione e ai campi carattere nelle strutture MQMD e MQMDE:

- La conversione predefinita dei dati del messaggio dell'applicazione si verifica solo quando tutte le seguenti istruzioni sono vere:
  - L'applicazione specifica GMCONV.
  - Il messaggio contiene dati che devono essere convertiti da o in una serie di caratteri non supportata.
  - La conversione predefinita è stata abilitata quando il gestore code è stato installato o riavviato.
- La conversione predefinita dei campi di caratteri nelle strutture MQMD e MQMDE si verifica come necessario, se la conversione predefinita è abilitata per il gestore code. La conversione viene eseguita anche se l'opzione GMCONV non è specificata dall'applicazione nella chiamata MQGET.

10. Il parametro **BUFFER** mostrato nell'esempio di programmazione RPG viene dichiarato come stringa; ciò limita la lunghezza massima del parametro a 256 byte. Se è richiesto un buffer più grande, il parametro deve essere dichiarato invece come una struttura o come un campo in un file fisico.

    Dichiarare il parametro come una struttura aumenta la lunghezza massima possibile a 9999 byte, mentre dichiarare il parametro come un campo in un file fisico aumenta la lunghezza massima possibile a circa 32 KB.

Parametri

La chiamata MQGET ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Handle di connessione.

Questo handle rappresenta la connessione al gestore code. Il valore di HCONN è stato restituito da una chiamata MQCONN o MQCONNX precedente.

HOBJ (numero intero con segno a 10 cifre) - immissione

Handle oggetto.

Questo handle rappresenta la coda da cui deve essere richiamato un messaggio. Il valore di HOBJ è stato restituito da una chiamata MQOPEN precedente. La coda deve essere stata aperta con una o più delle seguenti opzioni (consultare "MQOPEN (Open object) su IBM i" a pagina 1359 per i dettagli):

- OOINPS
- OOINPX
- OOINPQ
- OOBRW

MSGDSC (MQMD) - input/output

Descrittore del messaggio.

Questa struttura descrive gli attributi del messaggio richiesto e gli attributi del messaggio richiamato. Vedi "MQMD (Message Descriptor) su IBM i" a pagina 1140 per i dettagli.

Se BUFLEN è inferiore alla lunghezza del messaggio, MSGDSC viene ancora immesso dal gestore code, se GMATM è specificato nel parametro **GMO** (vedere il campo GMOPT descritto in "MQGMO (opzioni Get - message) su IBM i" a pagina 1105).

Se l'applicazione fornisce un MQMD version-1 , il messaggio restituito ha un prefisso MQMDE ai dati del messaggio dell'applicazione, ma solo se uno o più campi in MQMDE hanno un valore non

predefinito. Se tutti i campi in MQMDE hanno valori predefiniti, MQMDE viene omesso. Un nome formato FMMDE nel campo MDFMT in MQMD indica che è presente un MQMDE.

GMO (MQGMO) - input/output

Opzioni che controllano l'azione di MQGET.

Vedi "MQGMO (opzioni Get - message) su IBM i" a pagina 1105 per i dettagli.

BUFLEN (numero intero con segno a 10 cifre) - input

Lunghezza in byte dell'area BUFFER .

È possibile specificare zero per i messaggi che non hanno dati o se il messaggio deve essere rimosso dalla coda e i dati eliminati (GMATM deve essere specificato in questo caso).

Nota: La lunghezza del messaggio più lungo che è possibile leggere dalla coda viene fornita dall'attributo coda **MaxMsgLength** ; consultare "Attributi per le code" a pagina 1406.

BUFFER (stringa bit a 1 byte x BUFLEN) - output

Area per contenere i dati del messaggio.

Il buffer deve essere allineato su un limite appropriato alla natura dei dati nel messaggio. L'allineamento a 4 byte deve essere adatto per la maggior parte dei messaggi (inclusi i messaggi contenenti strutture di intestazione IBM MQ), ma alcuni messaggi potrebbero richiedere un allineamento più rigoroso. Ad esempio, un messaggio contenente un numero intero binario a 64 bit potrebbe richiedere un allineamento a 8 byte.

Se BUFLEN è inferiore alla lunghezza del messaggio, la maggior parte del messaggio possibile viene spostata in BUFFER ; Ciò si verifica se GMATM viene specificato nel parametro **GMO** (per ulteriori informazioni, consultare il campo GMOPT descritto in "MQGMO (opzioni Get - message) su IBM i" a pagina 1105).

La serie di caratteri e la codifica dei dati in **BUFFER** vengono forniti dai campi MDCSI e MDENC restituiti nel parametro **MSGDSC** . Se questi valori sono diversi da quelli richiesti dal destinatario, il destinatario deve convertire i dati del messaggio dell'applicazione nella serie di caratteri e nella codifica richiesti. L'opzione GMCONV può essere utilizzata con un'uscita scritta da un utente per eseguire la conversione dei dati del messaggio (consultare "MQGMO (opzioni Get - message) su IBM i" a pagina 1105 per dettagli su questa opzione).

Nota: Tutti gli altri parametri sulla chiamata MQGET si trovano nella serie di caratteri e nella codifica del gestore code locale (forniti dall'attributo del gestore code **CodedCharSetId** e ENNAT).

Se la chiamata ha esito negativo, è possibile che il contenuto del buffer sia stato ancora modificato.

DATLEN (numero intero con segno a 10 cifre) - emissione

Lunghezza del messaggio.

Questa è la lunghezza in byte dei dati dell'applicazione nel messaggio. Se la lunghezza di questo messaggio è maggiore di BUFLEN, nel parametro **BUFFER** vengono restituiti solo BUFLEN byte (ossia, il messaggio viene troncato). Se il valore è zero, significa che il messaggio non contiene dati dell'applicazione.

Se BUFLEN è inferiore alla lunghezza del messaggio, DATLEN viene ancora immesso dal gestore code, se GMATM è specificato nel parametro **GMO** (per ulteriori informazioni, consultare il campo GMOPT descritto in "MQGMO (opzioni Get - message) su IBM i" a pagina 1105). Ciò consente all'applicazione di determinare la dimensione del buffer richiesto per contenere i dati del messaggio e quindi emettere nuovamente la chiamata con un buffer della dimensione appropriata.

Tuttavia, se viene specificata l'opzione GMCONV e i dati del messaggio convertito sono troppo lunghi per rientrare in BUFFER, il valore restituito per DATLEN è:

- La lunghezza dei dati non convertiti, per i formati definiti del gestore code.

  In questo caso, se la natura dei dati causa l'espansione durante la conversione, l'applicazione deve assegnare un buffer più grande del valore restituito dal gestore code per DATLEN.

- Il valore restituito dall'uscita di conversione dati, per formati definiti dall'applicazione.

CMPCOD (numero intero con segno a 10 cifre) - output

Codice di completamento.

Il valore è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Codice di errore CMPCOD.

I seguenti codici motivo sono quelli che il gestore code può restituire per il parametro **REASON** .
Se l'applicazione specifica l'opzione GMCONV e viene richiamata un'uscita scritta dall'utente per convertire alcuni o tutti i dati del messaggio, è l'uscita che decide quale valore viene restituito per il parametro **REASON** . Di conseguenza, sono possibili valori diversi da quelli documentati più avanti in questa sezione.

Se CMPCOD è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se CMPCOD è CCWARN:

RC2120
(2120, X'848 ') Dati convertiti troppo grandi per il buffer.

RC2190
(2190, X'88E') Stringa convertita troppo grande per il campo.

RC2150
(2150, X'866 ') Stringa DBCS non valida.

RC2110
(2110, X'83E') Formato del messaggio non valido.

RC2243
(2243, X'8C3') I segmenti di messaggi hanno CCSID differenti.

RC2244
(2244, X'8C4') I segmenti dei messaggi hanno codifiche differenti.

RC2209
(2209, X'8A1') Nessun messaggio bloccato.

RC2119
(2119, X'847 ') Dati del messaggio non convertiti.

RC2272
(2272, X'8E0') Dati del messaggio parzialmente convertiti.

RC2145
(2145, X'861 ') Parametro del buffer di origine non valido.

RC2111
(2111, X'83F') Identificativo serie di caratteri codificati origine non valido.

RC2113
(2113, X'841 ') Codifica decimale compresso nel messaggio non riconosciuta.

RC2114
(2114, X'842 ') La codifica a virgola mobile nel messaggio non è stata riconosciuta.

RC2112
(2112, X'840 ') Numero intero di origine non riconosciuto.

RC2143
(2143, X'85F') Parametro di lunghezza origine non valido.

RC2146
(2146, X'862 ') Parametro buffer di destinazione non valido.

RC2115
(2115, X'843 ') Identificativo serie di caratteri codificati di destinazione non valido.

RC2117
(2117, X'845 ') La codifica decimale compresso specificata dal ricevitore non è stata riconosciuta.

RC2118
(2118, X'846 ') La codifica a virgola mobile specificata dal ricevitore non è stata riconosciuta.

RC2116
(2116, X'844 ') Codifica numero intero di destinazione non riconosciuta.

RC2079
(2079, X'81F') Messaggio troncato restituito (elaborazione completata).

RC2080
(2080, X'820 ') Messaggio troncato restituito (elaborazione non completata).

Se CMPCOD è CCFAIL:

RC2004
(2004, X'7D4') Parametro del buffer non valido.

RC2005
(2005, X'7D5') Parametro di lunghezza del buffer non valido.

RC2219
(2219, X'8AB') Chiamata MQI reimmessa prima del completamento della chiamata precedente.

RC2009
(2009, X'7D9') Connessione al gestore code persa.

RC2010
(2010, X'7DA') Parametro di lunghezza dati non valido.

RC2016
(2016, X'7E0') Ottiene inibiti per la coda.

RC2186
(2186, X'88A') Struttura delle opzioni Get - message non valida.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2019
(2019, X'7E3') Handle oggetto non valido.

RC2241
(2241, X'8C1') Gruppo di messaggi non completo.

RC2242
(2242, X'8C2') Messaggio logico non completo.

RC2259
(2259, X'8D3') Specifica di ricerca incongruente.

RC2245
(2245, X'8C5') Specifica dell'unità di lavoro non congruente.

RC2246
(2246, X'8C6') Messaggio sotto il cursore non valido per il recupero.

RC2247
(2247, X'8C7') Opzioni di corrispondenza non valide.

RC2026
(2026, X'7EA') Descrittore messaggio non valido.

RC2250
(2250, X'8CA') Numero di sequenza messaggio non valido.

RC2033
(2033, X'7F1') Nessun messaggio disponibile.

RC2034
(2034, X'7F2') Il cursore di ricerca non è posizionato sul messaggio.

RC2036
(2036, X'7F4') Coda non aperta per la ricerca.

RC2037
(2037, X'7F5') Coda non aperta per l'input.

RC2041
(2041, X'7F9') Definizione oggetto modificata dall'apertura.

RC2101
(2101, X'835 ') Oggetto danneggiato.

RC2046
(2046, X'7FE') Opzioni non valide o non congruenti.

RC2052
(2052, X'804 ') La coda è stata eliminata.

RC2058
(2058, X'80A') Nome gestore code non valido o sconosciuto.

RC2059
(2059, X'80B') Gestore code non disponibile per la connessione.

RC2161
(2161, X'871 ') Gestore code in fase di sospensione.

RC2162
(2162, X'872 ') Chiusura del gestore code.

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2024
(2024, X'7E8') Non è possibile gestire ulteriori messaggi all'interno dell'unità di lavoro corrente.

RC2072
(2072, X'818 ') Supporto punto di sincronizzazione non disponibile.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

RC2255
(2255, X'8CF') Unità di lavoro non disponibile per il gestore code.

RC2090
(2090, X'82A') Intervallo di attesa in MQGMO non valido.

RC2256
(2256, X'8D0') Versione errata di MQGMO fornita.

RC2257
(2257, X'8D1') Versione non corretta di MQMD fornita.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQGET(HCONN : HOBJ : MSGDSC : GMO :
C                                     BUFLEN : BUFFER : DATLEN :
C                                     CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQGET            PR                  EXTPROC('MQGET')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Object handle
D HOBJ                          10I 0 VALUE
D* Message descriptor
D MSGDSC                       364A
D* Options that control the action of MQGET
D GMO                          112A
D* Length in bytes of the Buffer area
D BUFLEN                        10I 0 VALUE
D* Area to contain the message data
D BUFFER                          *   VALUE
D* Length of the message
D DATLEN                        10I 0
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CMPCOD
D REASON                        10I 0
```

IBM i MQINQ (Interroga sugli attributi dell'oggetto) su IBM i

La chiamata MQINQ restituisce un array di numeri interi e una serie di stringhe di caratteri contenenti gli attributi di un oggetto.

Sono validi i seguenti tipi di oggetto:

- Coda
- Elenco nomi
- Definizione di processo
- Gestore code

- “Sintassi” a pagina 1342
- “Note d'utilizzo” a pagina 1342
- “Parametri” a pagina 1344
- “Dichiarazione RPG” a pagina 1350

Sintassi

MQINQ *(HCONN, HOBJ, SELCNT, SELS, IACNT, INTATR, CALEN, CHRATR, CMPCOD, REASON)*

Note d'utilizzo

1. I valori restituiti sono un'istantanea degli attributi selezionati. Non esiste alcuna garanzia che gli attributi non vengano modificati prima che l'applicazione possa agire sui valori restituiti.
2. Quando si apre una coda modello, viene creata una coda locale dinamica. Ciò è vero anche se si apre la coda del modello per interrogarsi sui relativi attributi.

   Gli attributi della coda dinamica (con alcune eccezioni) sono gli stessi della coda modello al momento della creazione della coda dinamica. Se, quindi, si utilizza la chiamata MQINQ su questa coda, il gestore code restituisce gli attributi della coda dinamica e non quelli della coda modello. Consultare Tabella 1 per dettagli su quali attributi della coda modello sono ereditati dalla coda dinamica.

3. Se l'oggetto interrogato è una coda alias, i valori di attributo restituiti dalla chiamata MQINQ sono quelli della coda alias e non quelli della coda di base in cui si risolve l'alias.
4. Se l'oggetto da interrogare è una coda cluster, gli attributi che possono essere interrogati dipendono dalla modalità di apertura della coda:
   - Se la coda cluster è aperta per l'interrogazione più uno o più di input, ricerca o impostazione, è necessario che sia presente un'istanza locale della coda cluster affinché l'apertura abbia esito positivo. In questo caso gli attributi che possono essere interrogati sono quelli validi per le code locali.
   - Se la coda del cluster è aperta per l'interrogazione da sola o per l'interrogazione e l'output, è possibile interrogare solo i seguenti attributi; l'attributo **QType** ha il valore QTCLUS in questo caso:
     - CAQD
     - CAQN
     - IDBND
     - IADPER
     - IADPRI
     - IAIPUT
     - TIPOIQ

     Se la coda cluster viene aperta senza un collegamento fisso (ovvero, OOBNDN specificato nella chiamata MQOPEN o OOBNDQ specificato quando l'attributo **DefBind** ha il valore BNDNOT), le successive chiamate MQINQ per la coda potrebbero interrogare istanze differenti della coda cluster, anche se di solito tutte le istanze hanno gli stessi valori di attributo.

   Per ulteriori informazioni sulle code del cluster, fare riferimento a Configurazione di un cluster di gestori code.
5. Se è necessario interrogare un certo numero di attributi e alcuni di essi devono essere impostati utilizzando la chiamata MQSET, potrebbe essere utile posizionare all'inizio degli array selettori gli attributi che devono essere impostati, in modo che gli stessi array (con conteggi ridotti) possano essere utilizzati per MQSET.
6. Se si verifica più di una delle situazioni di avvertenza (vedere il parametro **CMPCOD**), il codice motivo restituito è il *primo* nel seguente elenco che si applica:
   a. RC2068
   b. RC2022
   c. RC2008
7. Per ulteriori informazioni sugli attributi dell'oggetto, consultare:
   - “Attributi per le code” a pagina 1406
   - “Attributi per gli elenchi nomi” a pagina 1436
   - “Attributi per le definizioni di processo su IBM i .” a pagina 1437
   - “Attributi per il gestore code su IBM i” a pagina 1439
8. Una nuova coda locale SYSTEM.ADMIN.COMMAND.EVENT viene utilizzato per accodare i messaggi generati ogni volta che vengono emessi i comandi. I messaggi vengono inseriti in questa coda per la maggior parte dei comandi, a seconda di come è impostato l'attributo del gestore code CMDEV:
   - ENABLED - i messaggi di evento di comando vengono generati e inseriti nella coda per tutti i comandi riusciti.
   - I messaggi di evento del comando NODISPLAY vengono generati e inseriti nella coda per tutti i comandi riusciti diversi dal comando DISPLAY (MQSC) e dal comando Inquire (PCF).
   - DISABLED - i messaggi di eventi di comandi non vengono generati (questo è il valore predefinito iniziale del gestore code).

Parametri

La chiamata MQINQ ha i seguenti parametri:

HCONN (numero intero con segno a 10 cifre) - input

Handle di connessione.

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

HOBJ (numero intero con segno a 10 cifre) - immissione

Handle oggetto.

Questo handle rappresenta l'oggetto (di qualsiasi tipo) con gli attributi richiesti. La gestione deve essere stata restituita da una precedente chiamata MQOPEN che specificava l'opzione OOINQ.

SELCNT (numero intero con segno a 10 cifre) - immissione

Conteggio dei selettori.

Questo è il conteggio dei selettori forniti nell'array *SELS* . È il numero di attributi da restituire. Zero è un valore valido. Il numero massimo consentito è 256.

(numero intero con segno a 10 cifre x SELCNT) - immissione

Array di selettori di attributo.

Si tratta di un array di selettori di attributi **SELCNT** ; ogni selettore identifica un attributo (numero intero o carattere) con un valore richiesto.

Ogni selettore deve essere valido per il tipo di oggetto rappresentato da *HOBJ* , altrimenti la chiamata non riesce con codice di completamento CCFAIL e codice motivo RC2067.

Nel caso speciale delle code:

- Se il selettore non è valido per le code di *qualsiasi* tipo, la chiamata ha esito negativo con codice di completamento CCFAIL e codice motivo RC2067.
- Se il selettore è applicabile solo a code di tipo o tipi diversi da quello dell'oggetto, la chiamata ha esito positivo con codice di completamento CCWARN e codice motivo RC2068.
- Se la coda interrogata è una coda cluster, i selettori validi dipendono dal modo in cui la coda è stata risolta; consultare la nota di utilizzo 4 per ulteriori dettagli.

I selettori possono essere specificati in qualsiasi ordine. I valori di attributo che corrispondono ai selettori di attributi interi (selettori IA*) vengono restituiti in *INTATR* nello stesso ordine in cui tali selettori si verificano in *SELS*. I valori di attributo che corrispondono ai selettori di attributi di caratteri (selettori CA*) vengono restituiti in *CHRATR* nello stesso ordine in cui si verificano tali selettori. I selettori IA* possono essere intercalati con i selettori CA*; è importante solo l'ordine relativo all'interno di ciascun tipo.

Nota:

1. I selettori di attributi di caratteri e numeri interi vengono assegnati in due intervalli differenti; i selettori IA* si trovano nell'intervallo tra IAFRST e IALAST e i selettori CA* nell'intervallo tra CAFRST e CALAST.

   Per ogni intervallo, le costanti IALSTU e CALSTU definiscono il valore più alto accettato dal gestore code.
2. Se tutti i selettori IA* si verificano per primi, è possibile utilizzare gli stessi numeri di elemento per indirizzare gli elementi corrispondenti negli array *SELS* e *INTATR* .

Gli attributi che è possibile interrogare sono elencati nelle seguenti tabelle. Per i selettori CA*, la costante che definisce la lunghezza in byte della stringa risultante in *CHRATR* viene fornita tra parentesi.

Tabella 746. Selettori attributo MQINQ per code		
Selettore	**Descrizione**	**Nota**
CAALTD	Data della modifica più recente (LNDATE).	1
CAALTT	Ora della modifica più recente (LNTIME).	1
CABRQN	LNQN (backout - requeue name) eccessivo.	5
CABASQ	Nome della coda in cui l'alias si risolve (LNQN).	
CACFSN	LNCFSN (Coupling Facility Structure name).	3
CACLN	Nome cluster (LNCLUN).	1
CACLNL	Elenco nomi cluster (LNNLN).	1
CACRST	Data di creazione della coda (LNCRTD).	
RTT CAC	Ora di creazione della coda (LNCRTT).	
CAINIQ	Nome coda di iniziazione (LNQN).	
CAPRON	Nome della definizione del processo (LNPRON).	
CAQD	Descrizione coda (LNQD).	
CAQN	Nome coda (LNQN).	
CARQMN	Nome del gestore code remoto (LNQMN).	
CARQN	Il nome della coda remota come noto sul gestore code remoto (LNQN).	
CATRGG	Dati trigger (LNTRGD).	5
QQN CAX	Nome coda di trasmissione (LNQN).	
IABTHR	La soglia di ripristino.	5
IACDEP	Numero di messaggi in coda.	
IDBND	Collegamento predefinito.	1
IADINP	Opzione open - for - input predefinita.	5
IADPER	Persistenza del messaggio predefinita.	
IADPRI	La priorità messaggi predefinita.	5
VAL.UFF	Il tipo di definizione della coda.	
IADIST	Supporto elenco di distribuzione.	2
IAHGB	Indica se rafforzare il conteggio di backout.	5
IAIGET	Indica se le operazioni get sono consentite.	
IAIPUT	Se le operazioni di inserimento sono consentite.	
IAMLEN	la lunghezza massima del messaggio.	
IAMDEP	Numero massimo di messaggi consentiti nella coda.	
IAMDS	Se la priorità del messaggio è rilevante.	5
IAOIC	Numero di chiamate MQOPEN che hanno la coda aperta per l'input.	
OAOC	Numero di chiamate MQOPEN che hanno la coda aperta per l'output.	
DAQI	Attributo di controllo per eventi di grandezza della coda elevata.	4, 5

Tabella 746. Selettori attributo MQINQ per code (Continua)		
Selettore	**Descrizione**	**Nota**
DHL IAQ	Limite massimo per la profondità della coda.	4, 5
IQDLE	Attributo di controllo per gli eventi di profondità della coda bassa.	4, 5
DLL IAQ	Limite basso per la profondità della coda.	4, 5
IQDME	Attributo di controllo per il numero massimo di eventi di profondità coda.	4, 5
IQSI	Limite per l'intervallo di servizio della coda.	4, 5
IIAQSIE	Attributo di controllo per gli eventi di intervallo del servizio coda.	4, 5
TIPOIQ	Il tipo di coda.	
SSGD	Disposizione del gruppo di condivisione code.	3
IARINT	Intervallo di conservazione della coda.	5
IASCOP	Ambito definizione coda.	4, 5
IASHAR	Indica se la coda può essere condivisa per l'input.	
IATRGC	Controllo trigger.	
IATRGG	La lunghezza del trigger.	5
IATRGP	La priorità dei messaggi per i trigger.	5
IATRGT	Il tipo di trigger.	
IAUSAG	Utilizzo.	
CLWLUSEQ	Utilizzare le code remote.	

Nota:

1. Supportato sulle piattaforme seguenti:
   - AIX
   - IBM i
   - Windows
   - z/OS

   e per IBM MQ MQI clients collegati a questi sistemi.
2. Supportato sulle piattaforme seguenti:
   - AIX
   - IBM i
   - Windows

   e per i client IBM MQ connessi a questi sistemi.
3. z/OS Supportato su z/OS.
4. z/OS Non supportato su z/OS.
5. Non supportato su VSE/ESA.

Tabella 747. Selettori di attributi MQINQ per gli elenchi nomi

Selettore	Descrizione	Nota
CAALTD	Data della modifica più recente (LNDATE)	1
CAALTT	Ora della modifica più recente (LNTIME)	1
CALSTD	Descrizione elenco nomi (LNNLD)	1
CALSTN	Nome dell'oggetto elenco nomi (LNNLN)	1
CANAMS	Nomi nell'elenco nomi (LNQN x *Numero di nomi nell'elenco)*	1
IANAMC	Numero di nomi nell'elenco nomi	1
SSGD	Disposizione del gruppo di condivisione code	3

Tabella 748. Selettori di attributo MQINQ per le definizioni di processi

Selettore	Descrizione	Nota
CAALTD	Data della modifica più recente (LNDATE)	1
CAALTT	Ora della modifica più recente (LNTIME)	1
CAPPI	Identificativo applicazione (LNPROA)	5
CAENVD	Dati di ambiente (LNPROE)	5
CAPROD	Descrizione della definizione del processo (LNPROD)	5
CAPRON	Nome della definizione del processo (LNPRON)	5
CAUSRD	Dati utente (LNPROU)	5
IAPPT	Tipo di applicazione	5
SSGD	Disposizione del gruppo di condivisione code	3

Tabella 749. Selettori di attributo MQINQ per il gestore code

Selettore	Descrizione	Nota
CAALTD	Data della modifica più recente (LNDATE)	1
CAALTT	Ora della modifica più recente (LNTIME)	1
CACADX	Nome uscita definizione canale automatica (LNEXN)	1
CACLWD	Dati passati all'uscita del carico di lavoro del cluster (LNEXDA)	1
CACLWX	Nome dell'uscita del workload del cluster (LNEXN)	1
CACMDQ	Nome coda immissione comandi di sistema (LNQN)	5
CADLQ	Nome della coda di messaggi non recapitabili (LNQN)	5
CADXQN	Nome coda di trasmissione predefinita (LNQN)	5
MCAQ	Descrizione gestore code (LNQMD)	5
IDCAQM	Identificativo gestore code (LNQMID)	1
CQMN	Nome del gestore code locale (LNQMN)	5
SGNCAQ	Nome gruppo di condivisione code (LNQSGN)	3

Tabella 749. Selettori di attributo MQINQ per il gestore code (Continua)		
Selettore	**Descrizione**	**Nota**
CARPN	Il nome del cluster per il quale il gestore code fornisce i servizi di repository (LNQMN)	1
CARPNL	Nome dell'oggetto elenco nomi contenente i nomi dei cluster per i quali il gestore code fornisce i servizi del repository (LNNLN)	1
CMDEV	Attributo di controllo che determina se i messaggi generati quando vengono emessi i comandi vengono inseriti in una coda	8
AUTIAA	Attributo di controllo per gli eventi di autorizzazione	4, 5
IACAD	Attributo di controllo per la definizione di canale automatica	2
IACADE	Attributo di controllo per gli eventi di definizione canale automatici	2
IACLXQ	Tipo coda di trasmissione cluster predefinito	4
IACLWL	Lunghezza carico di lavoro cluster	1
IACCSI	CCSID (Coded character set identifier)	5
IACMDL	Livello di comando supportato dal gestore code	5
IACFGE	Attributo di controllo per gli eventi di configurazione	3
IADIST	Supporto elenco di distribuzione	2
IAINHE	Attributo di controllo per gli eventi di inibizione	4, 5
IALCLE	Attributo di controllo per gli eventi locali	4, 5
IAMHND	Il numero massimo di puntatori	5
IAMLEN	Lunghezza massima dei messaggi	5
IAMPRI	Priorità massima	5
IAMUNC	Numero massimo di messaggi di cui non è stato eseguito il commit in un'unità di lavoro	5
IAPFME	Attributo di controllo per gli eventi delle prestazioni	4, 5
IAPLAT	Piattaforma su cui risiede il gestore code	5
IARMTE	Attributo Controllo per eventi remoti	4, 5
IASSE	Attributo di controllo per gli eventi di avvio arresto	4, 5
IASYNC	Disponibilità punto di sincronizzazione	5
IATRLFT	Durata degli argomenti non amministrativi non utilizzati	
IATRGI	Intervallo trigger	5

IACNT (numero intero con segno a 10 cifre) - input

Conteggio degli attributi interi.

Questo è il numero di elementi nell'array INTATR . Zero è un valore valido.

Se questo è almeno il numero di selettori IA* nel parametro **SELS** , vengono restituiti tutti gli attributi integer richiesti.

INTATR (numero intero con segno a 10 cifre x IACNT) - emissione

Array di attributi integer.

Questo è un array di *IACNT* valori di attributo integer.

I valori dell'attributo integer vengono restituiti nello stesso ordine dei selettori IA* nel parametro **SELS** . Se l'array contiene più elementi del numero di selettori IA*, gli elementi in eccesso non vengono modificati.

Se HOBJ rappresenta una coda, ma un selettore di attributo non è applicabile a quel tipo di coda, viene restituito il valore specifico IAVNA per l'elemento corrispondente nell'array INTATR .

CALEN (numero intero con segno a 10 cifre) - immissione

Lunghezza del buffer degli attributi carattere.

È la lunghezza in byte del parametro **CHRATR** .

Deve essere almeno la somma delle lunghezze degli attributi carattere richiesti (consultare SELS). Zero è un valore valido.

CHRATR (stringa di caratteri a 1 byte x CALEN) - emissione

Attributi carattere.

Si tratta del buffer in cui vengono restituiti gli attributi carattere, concatenati insieme. La lunghezza del buffer viene fornita dal parametro **CALEN** .

Gli attributi carattere vengono restituiti nello stesso ordine dei selettori CA* nel parametro **SELS** . La lunghezza di ogni stringa di attributo è fissa per ogni attributo (consultare SELS) e il valore in esso contenuto viene riempito a destra con spazi vuoti, se necessario. Se il buffer è più grande di quello necessario per contenere tutti gli attributi di caratteri richiesti (incluso il riempimento), i byte oltre l'ultimo valore di attributo restituito non vengono modificati.

Se HOBJ rappresenta una coda, ma un selettore di attributo non è applicabile a tale tipo di coda, viene restituita una stringa di caratteri composta interamente da asterischi (*) come valore di tale attributo in CHRATR.

CMPCOD (numero intero con segno a 10 cifre) - output

Codice di completamento.

Il valore è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Codice di errore CMPCOD.

Se CMPCOD è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCWARN:

RC2008
(2008, X'7D8') Spazio non sufficiente consentito per gli attributi carattere.

RC2022
(2022, X'7E6') Spazio non sufficiente consentito per gli attributi integer.

RC2068
(2068, X'814 ') Selettore non applicabile al tipo di coda.

Se *CMPCOD* è CCFAIL:

RC2219
(2219, X'8AB') Chiamata MQI reimmessa prima del completamento della chiamata precedente.

RC2006
(2006, X'7D6') Lunghezza degli attributi dei caratteri non valida.

RC2007
(2007, X'7D7') Stringa di attributi carattere non valida.

RC2009
(2009, X'7D9') Connessione al gestore code persa.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2019
(2019, X'7E3') Handle oggetto non valido.

RC2021
(2021, X'7E5') Conteggio di attributi interi non valido.

RC2023
(2023, X'7E7') Array di attributi interi non valido.

RC2038
(2038, X'7F6') Coda non aperta per l'indagine.

RC2041
(2041, X'7F9') Definizione oggetto modificata dall'apertura.

RC2101
(2101, X'835 ') Oggetto danneggiato.

RC2052
(2052, X'804 ') La coda è stata eliminata.

RC2058
(2058, X'80A') Nome gestore code non valido o sconosciuto.

RC2059
(2059, X'80B') Gestore code non disponibile per la connessione.

RC2162
(2162, X'872 ') Chiusura del gestore code.

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2065
(2065, X'811 ') Conteggio dei selettori non valido.

RC2067
(2067, X'813 ') Selettore attributo non valido.

RC2066
(2066, X'812 ') Conteggio dei selettori troppo grande.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQINQ(HCONN : HOBJ : SELCNT :
C                                     SELS(1) : IACNT : INTATR(1) :
C                                     CALEN : CHRATR : CMPCOD :
C                                     REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQINQ            PR                  EXTPROC('MQINQ')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Object handle
D HOBJ                          10I 0 VALUE
D* Count of selectors
D SELCNT                        10I 0 VALUE
D* Array of attribute selectors
D SELS                          10I 0
D* Count of integer attributes
D IACNT                         10I 0 VALUE
D* Array of integer attributes
D INTATR                        10I 0
D* Length of character attributes buffer
D CALEN                         10I 0 VALUE
D* Character attributes
D CHRATR                          *   VALUE
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CMPCOD
D REASON                        10I 0
```

IBM i MQINQMP (proprietà Interroga messaggio) su IBM i

La chiamata a MQINQMP restituisce il valore di una proprietà di un messaggio.

- "Sintassi" a pagina 1351
- "Parametri" a pagina 1351
- "Dichiarazione RPG" a pagina 1355

Sintassi

MQINQMP *(Hconn, Hmsg, InqPropOpts, Name, PropDesc, Type, ValueLength, Value, DataLength, CompCode, Reason)*

Parametri

La chiamata MQINQMP ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* deve corrispondere all'handle di connessione utilizzato per creare l'handle del messaggio specificato nel parametro **Hmsg** .

Se l'handle del messaggio è stato creato utilizzando HCUNAS, è necessario stabilire una connessione valida sul thread che richiede una proprietà dell'handle del messaggio, altrimenti la chiamata non riesce con RC2009.

HMSG (numero intero con segno a 20 cifre) - input

Questo è l'handle del messaggio da interrogare. Il valore è stato restituito da una precedente chiamata **MQCRTMH** .

INQOPT (MQIMPO) - immissione

Consultare il tipo di dati MQIMPO per i dettagli.

PRNAME (MQCHARV) - input

Descrive il nome della proprietà da interrogare.

Se non è possibile trovare alcuna proprietà con questo nome, la chiamata ha esito negativo con motivo RC2471.

È possibile utilizzare il segno percentuale (%) alla fine del nome della proprietà. Il carattere jolly corrisponde a zero o più caratteri, incluso il carattere punto (.). Ciò consente a un'applicazione di analizzare il valore di molte proprietà. Richiamare MQINQMP con l'opzione IPINQF per ottenere la prima proprietà corrispondente e nuovamente con l'opzione IPINQN per ottenere la proprietà corrispondente successiva. Quando non sono disponibili ulteriori proprietà corrispondenti, la chiamata ha esito negativo con RC2471. Se il campo *ReturnedName* della struttura InqPropOpts viene inizializzato con un indirizzo o un offset per il nome restituito della proprietà, questo viene completato al ritorno da MQINQMP con il nome della proprietà corrispondente. Se il campo *VSBufSize* di *ReturnedName* nella struttura InqPropOpts è inferiore alla lunghezza del nome della proprietà restituito, il codice di completamento viene impostato su CCFAIL con motivo RC2465.

Le proprietà che hanno sinonimi noti vengono restituite come segue:

1. Proprietà con il prefisso "mqps." vengono restituiti con il nome della proprietà IBM MQ . Ad esempio, "MQTopicString" è il nome restituito invece di "mqps.Top".
2. Proprietà con il prefisso "jms." o "mcd." vengono restituiti come nome campo intestazione JMS . Ad esempio, "JMSExpiration" è il nome restituito anziché "jms.Exp".
3. Proprietà con il prefisso "usr." sono restituiti senza tale prefisso. Ad esempio, viene restituito "Colore" invece di "usr.Color".

Le proprietà con sinonimi vengono restituite una volta sola.

Nel linguaggio di programmazione RPG, vengono definite le seguenti variabili macro per l'interrogazione di tutte le proprietà e di tutte le proprietà che iniziano con "usr.":

INQALL
Analizzare tutte le proprietà del messaggio.

INQUSR
Interrogare tutte le proprietà del messaggio che avviano "usr.". Il nome restituito viene restituito senza "usr." .

Se viene specificato IPINQN ma il nome è stato modificato dalla chiamata precedente o questa è la prima chiamata, allora IPINQF è implicito.

Consultare Nomi proprietà e Limitazioni nome proprietà per ulteriori informazioni sull'utilizzo dei nomi proprietà.

PRPDSC (MQPD) - output

Questa struttura viene utilizzata per definire gli attributi di una proprietà, inclusi gli eventi che si verificano se la proprietà non è supportata, il contesto del messaggio a cui appartiene la proprietà e i messaggi in cui la proprietà deve essere copiata. Consultare MQPD per dettagli su questa struttura.

TYPE (numero intero con segno a 10 cifre) - input/output

Al ritorno dalla chiamata MQINQMP, questo parametro è impostato sul tipo di dati *Valore*. Il tipo di dati può essere uno dei seguenti:

TIPOnota di carico
Un valore booleano.

TYPBST
una stringa di byte.

TYPI8
Un numero intero con segno a 8 bit.

TYPI16
Un numero intero con segno a 16 bit.

TYPI32
Un numero intero con segno a 32 bit.

TYPI64
Un numero intero con segno a 64 bit.

TYPF32
Un numero a virgola mobile a 32 bit.

TYPF64
Un numero a virgola mobile a 64 bit.

TIPOSTR
Una stringa di caratteri.

TYPNUL
La proprietà esiste ma ha un valore null.

Se il tipo di dati del valore della proprietà non viene riconosciuto, viene restituito TYPSTR e una rappresentazione di stringa del valore viene inserita nell'area *Valore* . È possibile trovare una rappresentazione stringa del tipo di dati nel campo *IPTYP* del parametro *IPOPT* . Viene restituito un codice di completamento di avvertenza con motivo RC2467.

Inoltre, se viene specificata l'opzione IPCTYP, è richiesta la conversione del valore della proprietà. Utilizzare *Tipo* come input per specificare il tipo di dati per cui si desidera restituire la proprietà. Consultare la descrizione dell'opzione IPCTYP di "MQIMPO (Interroga opzioni proprietà messaggio) su IBM i" a pagina 1133 per dettagli sulla conversione del tipo di dati.

Se non si richiede la conversione del tipo, è possibile utilizzare il seguente valore nell'input:

TYPAST
Il valore della proprietà viene restituito senza convertirne il tipo di dati.

VALLEN (numero intero con segno a 10 cifre) - input

La lunghezza in byte dell'area Valore.

Specificare zero per le proprietà per cui non è richiesto il valore restituito. Queste potrebbero essere proprietà progettate da un'applicazione per avere un valore null o una stringa vuota. Specificare anche zero se è stata specificata l'opzione IPQLEN; in questo caso, non viene restituito alcun valore.

VALUE (bit a 1 byte stringxVALLEN) - output

Questa è l'area che deve contenere il valore della proprietà richiesta. Il buffer deve essere allineato su un limite appropriato per il valore restituito. In caso contrario, potrebbe verificarsi un errore quando si accede al valore in un secondo momento.

Se *VALLEN* è inferiore alla lunghezza del valore della proprietà, la maggior parte del valore della proprietà viene spostata in *VALUE* e la chiamata ha esito negativo con codice di completamento CCFAIL e motivo RC2469.

La serie di caratteri dei dati in *VALUE* viene fornita dal campo IPRETCSI nel parametro INQOPT. La codifica dei dati in *VALUE* viene fornita dal campo IPRETENC nel parametro INQOPT.

Se il parametro *VALLEN* è zero, VALUE non viene indicato.

DATLEN (numero intero con segno a 10 cifre) - emissione

Questa è la lunghezza in byte del valore della proprietà effettiva come restituito nell'area *Valore* .

Se *DataLength* è inferiore alla lunghezza del valore della proprietà, *DataLength* viene ancora immesso alla restituzione dalla chiamata MQINQMP. Ciò consente all'applicazione di determinare la dimensione del buffer richiesta per contenere il valore della proprietà e quindi emettere nuovamente la chiamata con un buffer della dimensione appropriata.

Possono essere restituiti anche i seguenti valori.

Se il parametro *Tipo* è impostato su TYPSTR o TYPBST:

VLEMP
La proprietà esiste ma non contiene caratteri o byte.

CMPCOD (numero intero con segno a 10 cifre) - output

Il codice di completamento; è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Il codice di errore che qualifica *CompCode*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è CCWARN:

RC2492
(2492, X'09BC') Nome proprietà restituito non convertito.

RC2466
(2466, X'09A2') Valore proprietà non convertito.

RC2467
(2467, X'09A3') Il tipo di dati della proprietà non è supportato.

RC2421
(2421, X'0975 ') Impossibile analizzare una cartella MQRFH2 contenente le proprietà.

Se *CMPCOD* è CCFAIL:

RC2204
(2204, X'089C') Adattatore non disponibile.

RC2130
(2130, X'0852 ') Impossibile caricare il modulo di servizio adattatore.

RC2157
(2157, X'086D') Gli ASID principale e principale differiscono.

RC2004
(2004, X'07D4') Parametro valore non valido.

RC2005
(2005, X'07D5') Parametro di lunghezza valore non valido.

RC2219
(2219, X'08AB') Chiamata MQI immessa prima del completamento della chiamata precedente.

RC2009
(2009, X'07D9') Connessione al gestore code persa.

RC2010
(2010, X'07DA') Parametro di lunghezza dati non valido.

RC2464
(2464, X'09A0') Struttura delle opzioni della proprietà del messaggio di interrogazione non valida.

RC2460
(2460, X'099C') Gestione messaggio non valida.

RC2499
(2499, X'09C3') handle del messaggio già in uso.

RC2064
(2046, X'07F8') Opzioni non valide o non congruenti.

RC2482
(2482, X'09B2') Struttura descrittore proprietà non valido.

RC2470
(2470, X'09A6') Conversione dal tipo di dati effettivo a quello richiesto non supportata.

RC2442
(2442, X'098A') Nome proprietà non valido.

RC2465
(2465, X'09A1') Nome proprietà troppo grande per il buffer dei nomi restituito.

RC2471
(2471, X'09A7) Proprietà non disponibile.

RC2469
(2469, X'09A5') Valore della proprietà troppo grande per l'area Valore.

RC2472
(2472, X'09A8') Errore di formato numero rilevato nei dati del valore.

RC2473
(2473, X'09A9') Tipo di proprietà richiesto non valido.

RC2111
(2111, X'083F') Identificativo serie di caratteri codificato del nome proprietà non valido.

RC2071
(2071, X'0871 ') Memoria disponibile insufficiente.

RC2195
(2195, X'0893 ') Si è verificato un errore non previsto.

Per informazioni dettagliate su questi codici, consultare:

- Messaggi IBM MQ for z/OS , codici di completamento e di errore per IBM MQ for z/OS
- Messaggi e codici di errore per tutte le altre piattaforme IBM MQ

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQINQMP(HCONN : HMSG : INQOPT :
                                        PRNAME : PRPDSC : TYPE :
                                        VALLEN : VALUE : DATLEN :
                                        CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
DMQINQMP          PR                  EXTPROC('MQINQMP')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Message handle
D HMSG                          20I 0 VALUE
D* Options that control the action of MQINQMP
D INQOPT                        72A
D* Property name
D PRNAME                        32A
D* Property descriptor
D PRPDSC                        24A
D* Property data type
D TYPE                          10I 0
D* Length in bytes of the Value area
D VALLEN                        10I 0 VALUE
D* Property value
D VALUE                           *   VALUE
D* Length of the property value
D DATLEN                        10I 0
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CompCode
D REASON                        10I 0
```

IBM i MQMHBUF (Conversione dell'handle del messaggio nel buffer) su IBM i

MQMHBUF converte un handle del messaggio in un buffer ed è l'inverso della chiamata MQBUFMH.

Sintassi

MQMHBUF *(Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer, DataLength, CompCode, Reason)*

Note d'utilizzo

MQMHBUF converte un handle del messaggio in un buffer.

È possibile utilizzarla con un'uscita API MQGET per accedere ad alcune proprietà, utilizzando le API delle proprietà del messaggio, e quindi passare queste proprietà in un buffer di nuovo a un'applicazione progettata per utilizzare le intestazioni MQRFH2 piuttosto che gli handle del messaggio.

Questa chiamata è l'inverso della chiamata MQBUFMH, che è possibile utilizzare per analizzare le proprietà del messaggio da un buffer in un handle del messaggio.

Parametri

La chiamata MQMHBUF ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Questo handle rappresenta la connessione al gestore code.

Il valore di *HCONN* deve corrispondere all'handle di connessione utilizzato per creare l'handle del messaggio specificato nel parametro **HMSG** .

Se l'handle del messaggio è stato creato utilizzando HCUNAS, è necessario stabilire una connessione valida sul thread che elimina l'handle del messaggio. Se non viene stabilita una connessione valida, la chiamata non riesce con RC2009.

HMSG (numero intero con segno a 20 cifre) - input

Questo handle è l'handle del messaggio per cui è richiesto un buffer.

Il valore è stato restituito da una precedente chiamata MQCRTMH.

MHBOPT (MQMHBO) - input

La struttura MQMHBO consente alle applicazioni di specificare le opzioni che controllano la modalità di produzione dei buffer dagli handle dei messaggi.

Vedi "MQBMHO (Buffer per le opzioni di gestione dei messaggi) su IBM i" a pagina 1044 per i dettagli.

PRNAME (MQCHARV) - input

Il nome della proprietà o delle proprietà da inserire nel buffer.

Se non è possibile trovare alcuna proprietà corrispondente al nome, la chiamata ha esito negativo con RC2471.

Caratteri jolly

È possibile utilizzare un carattere jolly per inserire più di una proprietà nel buffer. Per fare ciò, utilizzare il simbolo di percentuale (%) alla fine del nome della proprietà. Questo carattere jolly corrisponde a zero o più caratteri, incluso il carattere punto (.).

Consultare Nomi proprietà e Limitazioni nome proprietà per ulteriori informazioni sull'utilizzo dei nomi proprietà.

MSGDSC (MQMD) - input/output

La struttura *MSGDSC* descrive il contenuto dell'area buffer.

In fase di output, i campi *Encoding*, *CodedCharSetId* e *Format* sono impostati per descrivere correttamente la codifica, l'identificativo della serie di caratteri e il formato dei dati nell'area di buffer come scritto dalla chiamata.

I dati in questa struttura sono nella serie di caratteri e nella codifica dell'applicazione.

BUFLEN (numero intero con segno a 10 cifre) - input

BUFFLEN è la lunghezza dell'area Buffer, in byte.

BUFFER (stringa di bit a 1 byte x BUFLEN) - input/output

BUFFER definisce l'area contenente il buffer di messaggi. Per la maggior parte dei dati, è necessario allineare il buffer su un limite di 4 byte.

Se *BUFFER* contiene dati numerici o di caratteri, impostare i campi *CodedCharSetId* e *Encoding* nel parametro **MSGDSC** sui valori appropriati per i dati; ciò consente la conversione dei dati, se necessario.

Se le proprietà vengono trovate nel buffer del messaggio, vengono facoltativamente rimosse; in seguito diventano disponibili dall'handle del messaggio al ritorno dalla chiamata.

Nel linguaggio di programmazione C, il parametro viene dichiarato come un puntatore a void, il che significa che l'indirizzo di qualsiasi tipo di dati può essere specificato come parametro.

Se il parametro **BUFLEN** è zero, non si fa riferimento a *BUFFER* . In questo caso, l'indirizzo del parametro inoltrato dai programmi scritti nell'assembler C o System/390 può essere null.

DATLEN (numero intero con segno a 10 cifre) - emissione

DATLEN è la lunghezza, in byte, delle proprietà restituite nel buffer. Se il valore è zero, nessuna proprietà corrispondeva al valore fornito in *PRNAME* e la chiamata ha esito negativo con codice di errore RC2471.

Se *BUFLEN* è inferiore alla lunghezza richiesta per memorizzare le proprietà nel buffer, la chiamata MQMHBUF ha esito negativo con RC2469, ma viene ancora immesso un valore in *DATLEN*. Ciò consente all'applicazione di determinare la dimensione del buffer richiesto per contenere le proprietà e quindi emettere nuovamente la chiamata con il *BUFLEN*richiesto.

CMPCOD (numero intero con segno a 10 cifre) - output

Il codice di completamento; è uno dei seguenti:

CCOK
Completamento con esito positivo.

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Il codice di errore che qualifica *CMPCOD*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCFAIL:

RC2204
(2204, X'089C') Adattatore non disponibile.

RC2130
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

RC2157
(2157, X'86D') Gli ASID principale e home differiscono.

RC2501
(2501, X'095C') L'handle del messaggio nella struttura delle opzioni del buffer non è valida.

RC2004
Parametro buffer (2004, X'07D4') non valido.

RC2005
Parametro di lunghezza buffer (2005, X'07D5') non valido.

RC2219
(2219, X'08AB') Chiamata MQI immessa prima del completamento della chiamata precedente.

RC2009
(2009, X'07D9') Connessione al gestore code persa.

RC2010
(2010, X'07DA') Parametro di lunghezza dati non valido.

RC2460
(2460, X'099C') Gestione messaggio non valida.

RC2026
(2026, X'07EA') Descrittore messaggio non valido.

RC2499
(2499, X'09C3') handle del messaggio già in uso.

RC2046
(2046, X'07FE') Opzioni non valide o non congruenti.

RC2442
(2442, X'098A') Il nome proprietà non è valido.

RC2471
(2471, X'09A7') Proprietà non disponibile.

RC2469
(2469, X'09A5') Il valore BufferLength è troppo piccolo per contenere le proprietà specificate.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                   CALLP     MQMHBUF(HCONN : HMSG : MHBOPT :
                                      PRNAME : MSGDSC : BUFLEN :
                                      BUFFER : DATLEN :
                                      CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
DMQMHBUF          PR                  EXTPROC('MQMHBUF')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Message handle
D HMSG                          20I 0 VALUE
D* Options that control the action of MQMHBUF
D MHBOPT                        12A
D* Property name
D PRNAME                        32A
D* Message descriptor
```

```
     D MSGDSC                        364A
     D* Length in bytes of the Buffer area
     D BUFLEN                       10I 0 VALUE
     D* Area to contain the properties
     D BUFFER                         *   VALUE
     D* Length of the properties
     D DATLEN                       10I 0
     D* Completion code
     D CMPCOD                       10I 0
     D* Reason code qualifying CompCode
     D REASON                       10I 0
```

IBM i MQOPEN (Open object) su IBM i

La chiamata MQOPEN stabilisce l'accesso a un oggetto.

Sono validi i seguenti tipi di oggetto:

- Coda (inclusi elenchi di distribuzione)
- Elenco nomi
- Definizione di processo
- Gestore code
- Argomento

Indice analitico

Sintassi

MQOPEN *(HCONN, OBJDSC, OPTS, HOBJ, CMPCOD, REASON)*

Note d'utilizzo

1. L'oggetto aperto è uno dei seguenti:
   - Una coda, per:
     - Richiamare o sfogliare i messaggi (utilizzando la chiamata MQGET)
     - Inserire i messaggi (utilizzando la chiamata MQPUT)
     - Interrogare sugli attributi della coda (utilizzando la chiamata MQINQ)
     - Impostare gli attributi della coda (utilizzando la chiamata MQSET)

     Se la coda denominata è una coda modello, viene creata una coda locale dinamica.

     Un elenco di distribuzione è un tipo speciale di oggetto coda che contiene un elenco di code. Può essere aperto per inserire messaggi, ma non per ottenere o sfogliare messaggi o per interrogare o impostare attributi. Consultare la nota di utilizzo 8 per ulteriori dettagli.

     Una coda che ha `QSGDISP(GROUP)` è un tipo speciale di definizione della coda che non può essere utilizzato con le chiamate MQOPEN o MQPUT1 .
   - Un elenco nomi, per:
     - Verificare i nomi delle code nell'elenco (utilizzando la chiamata MQINQ).
   - Una definizione di processo, al fine di:
     - Interrogare gli attributi del processo (utilizzando la chiamata MQINQ).
   - Il gestore code, per:

– Richiedere informazioni sugli attributi del gestore code locale (utilizzando la chiamata MQINQ).

2. È valido per un'applicazione per aprire lo stesso oggetto più di una volta. Viene restituito un handle di oggetto diverso per ogni apertura. Ogni handle restituito può essere utilizzato per le funzioni per cui è stato eseguito il corrispondente open.
3. Se l'oggetto aperto è una coda ma non una coda cluster, tutta la risoluzione dei nomi all'interno del gestore code locale si verifica al momento della chiamata MQOPEN. Ciò potrebbe includere una o più delle seguenti operazioni per una particolare chiamata MQOPEN:

   • Risoluzione alias al nome di una coda di base
   • Risoluzione del nome di una definizione locale di una coda remota sul nome del gestore code remoto e il nome con cui la coda è nota sul gestore code remoto
   • Risoluzione del nome del gestore code remoto sul nome di una coda di trasmissione locale

   Tuttavia, tenere presente che le chiamate MQINQ o MQSET successive per l'handle si riferiscono esclusivamente al nome che è stato aperto e non all'oggetto risultante dopo la risoluzione del nome. Ad esempio, se l'oggetto aperto è un alias, gli attributi restituiti dalla chiamata MQINQ sono gli attributi dell'alias, non gli attributi della coda di base in cui si risolve l'alias. Tuttavia, il controllo della risoluzione dei nomi viene ancora eseguito indipendentemente da quanto specificato per il parametro **OPTS** sul MQOPEN corrispondente.

   Se l'oggetto che si sta aprendo è una coda cluster, la risoluzione del nome può verificarsi al momento della chiamata MQOPEN o essere rinviata fino a un momento successivo. Il punto in cui si verifica la risoluzione è controllato dalle opzioni OOBND* specificate nella chiamata MQOPEN:

   • OOBNDO
   • DN OBN
   • OOBNDQ

   Per ulteriori informazioni sulla risoluzione dei nomi per le code cluster, consultare Risoluzione dei nomi .
4. Gli attributi di un oggetto possono essere modificati mentre un'applicazione ha l'oggetto aperto. In molti casi, l'applicazione non lo nota, ma per alcuni attributi il gestore code contrassegna l'handle come non più valido. Essi sono:

   • Qualsiasi attributo che influisce sulla risoluzione del nome dell'oggetto. Ciò si applica indipendentemente dalle opzioni aperte utilizzate e include quanto segue:
     – Una modifica all'attributo **BaseQName** di una coda alias aperta.
     – Una modifica agli attributi della coda **RemoteQName** o **RemoteQMgrName** , per qualsiasi handle aperto per questa coda o per una coda che si risolve tramite questa definizione come alias del gestore code.
     – Qualsiasi modifica che fa sì che un handle attualmente aperto per una coda remota si risolva in una coda *di trasmissione* diversa o che non si risolva in una coda. Ad esempio, può includere:
       - Una modifica all'attributo **XmitQName** della definizione locale di una coda remota, se la definizione viene utilizzata per una coda o per un alias del gestore code.

       C'è un'eccezione a questo, ovvero la creazione di una nuova coda di trasmissione. Un handle che si sarebbe risolto in questa coda se fosse stato presente quando l'handle è stato aperto, ma invece risolto nella coda di trasmissione predefinita, non viene reso non valido.
     – Una modifica all'attributo del gestore code **DefXmitQName** . In questo caso, tutti gli handle aperti che si sono risolti nella coda precedentemente denominata (che si sono risolti solo perché era la coda di trasmissione predefinita) sono contrassegnati come non validi. Gli handle risolti in questa coda per altri motivi non vengono influenzati.
   • L'attributo della coda **Shareability** , se ci sono due o più handle che attualmente forniscono l'accesso OOINPS per questa coda o per una coda che si risolve in questa coda. In tal caso, *tutti* gli handle aperti per questa coda o per una coda che si risolve in questa coda, vengono contrassegnati come non validi, indipendentemente dalle opzioni di apertura.

- L'attributo della coda **Usage** , per tutti gli handle aperti per questa coda o per una coda che si risolve in questa coda, indipendentemente dalle opzioni di apertura.

Quando un handle viene contrassegnato come non valido, tutte le chiamate successive (diverse da MQCLOSE) che utilizzano questo handle hanno esito negativo con codice motivo RC2041; l'applicazione deve emettere una chiamata MQCLOSE (utilizzando l'handle originale) e quindi riaprire la coda. Qualsiasi aggiornamento non sottoposto a commit rispetto al vecchio handle delle precedenti chiamate riuscite può essere ancora sottoposto a commit o a backout, come richiesto dalla logica dell'applicazione.

Se la modifica di un attributo provoca tale situazione, è necessario utilizzare una versione speciale "force" del comando.

5. Il gestore code esegue i controlli di sicurezza quando viene emessa una chiamata MQOPEN, per verificare che l'identificativo utente con cui l'applicazione è in esecuzione disponga del livello di autorizzazione appropriato prima che sia consentito l'accesso. Il controllo dell'autorizzazione viene eseguito sul nome dell'oggetto che si sta aprendo e non sul nome o sui nomi, come risultato dopo che un nome è stato risolto.

   Se l'oggetto che si sta aprendo è una coda modello, il gestore code esegue un controllo di sicurezza completo rispetto sia al nome della coda modello che al nome della coda dinamica che viene creata. Se la coda dinamica risultante viene aperta esplicitamente, viene eseguito un ulteriore controllo di sicurezza della risorsa rispetto al nome della coda dinamica.
6. Una coda remota può essere specificata in due modi nel parametro **OBJDSC** di questa chiamata (consultare i campi *ODON* e *ODMN* descritti in "MQOD (Object descriptor) su IBM i" a pagina 1192):

   - Specificando per *ODON* il nome di una definizione locale della coda remota. In tal caso, *ODMN* fa riferimento al gestore code locale e può essere specificato come vuoto.

     La convalida di sicurezza eseguita dal gestore code locale verifica che l'utente sia autorizzato ad aprire la definizione locale della coda remota.
   - Specificando per *ODON* il nome della coda remota come è noto al gestore code remoto. In questo caso *ODMN* è il nome del gestore code remoto.

     La convalida di sicurezza eseguita dal gestore code locale verifica che l'utente sia autorizzato a inviare i messaggi alla coda di trasmissione risultante dal processo di risoluzione dei nomi.

   In entrambi i casi:

   - Nessun messaggio viene inviato dal gestore code locale al gestore code remoto per verificare che l'utente sia autorizzato a inserire messaggi nella coda.
   - Quando un messaggio arriva al gestore code remoto, il gestore code remoto potrebbe rifiutarlo perché l'utente che ha originato il messaggio non è autorizzato.
7. Una chiamata MQOPEN con l'opzione OOBRW stabilisce un cursore di ricerca da utilizzare con le chiamate MQGET che specificano l'handle dell'oggetto e una delle opzioni di esplorazione. Ciò consente di eseguire la scansione della coda senza modificarne il contenuto. Un messaggio che è stato trovato durante la navigazione può essere successivamente rimosso dalla coda utilizzando l'opzione GMMUC.

   Più cursori di esplorazione possono essere attivi per una singola applicazione emettendo diverse richieste MQOPEN per la stessa coda.
8. Le seguenti note si applicano all'uso degli elenchi di distribuzione.

   - I campi nella struttura MQOD devono essere impostati come segue quando si apre un elenco di distribuzione:
     - *ODVER* deve essere ODVER2 o superiore.
     - *ODOT* deve essere OTQ.
     - *ODON* deve essere vuoto o la stringa null.
     - *ODMN* deve essere vuoto o la stringa null.
     - *ODREC* Deve essere maggiore di zero.

- Uno tra *ODORO* e *ODORP* deve essere zero e l'altro diverso da zero.
- Non più di uno tra *ODRRO* e *ODRRP* può essere diverso da zero.
- Devono essere presenti *ODREC* record oggetto, indirizzati da *ODORO* o *ODORP*. I record oggetto devono essere impostati sui nomi delle code di destinazione da aprire.
- Se uno tra *ODRRO* e *ODRRP* è diverso da zero, devono essere presenti *ODREC* record di risposta. Essi vengono impostati dal gestore code se la chiamata viene completata con il codice motivo RC2136.

Un MQOD version-2 può essere utilizzato anche per aprire una singola coda che non si trova in un elenco di distribuzione, assicurandosi che *ODREC* sia zero.

- Nel parametro **OPTS** sono valide solo le seguenti opzioni di apertura:
  - OOOUT
  - OOPAS*
  - OSET*
  - OOALTU
  - OOFIQ
- Le code di destinazione nell'elenco di distribuzione possono essere code locali, alias o remote, ma non possono essere code modello. Se viene specificata una coda modello, tale coda non viene aperta, con codice di errore RC2057. Tuttavia, ciò non impedisce che altre code nell'elenco vengano aperte correttamente.
- I parametri del codice di completamento e del codice di errore sono impostati come segue:
  - Se le operazioni di apertura per le code nell'elenco di distribuzione hanno avuto esito positivo o negativo nello stesso modo, i parametri del codice di completamento e del codice motivo vengono impostati per descrivere il risultato comune. I record di risposta MQRR (se forniti dall'applicazione) non sono impostati in questo caso.

    Ad esempio, se ogni apertura ha esito positivo, il codice di completamento viene impostato su CCOK e il codice motivo è RCNONE; se ogni apertura ha esito negativo perché non esiste alcuna coda, i parametri vengono impostati su CCFAIL e RC2085.
  - Se le operazioni di apertura per le code nell'elenco di distribuzione non hanno esito positivo o negativo nello stesso modo:
    - Il parametro del codice di completamento è impostato su CCWARN se almeno un'apertura ha avuto esito positivo e su CCFAIL se tutti hanno avuto esito negativo.
    - Il parametro codice di errore è impostato su RC2136.
    - I record di risposta (se forniti dall'applicazione) sono impostati sui codici di completamento individuali e sui codici motivo per le code nell'elenco di distribuzione.
- Quando un elenco di distribuzione è stato aperto correttamente, l'handle *HOBJ* restituito dalla chiamata può essere utilizzato nelle chiamate MQPUT successive per inserire i messaggi nelle code nell'elenco di distribuzione e in una chiamata MQCLOSE per rinunciare all'accesso all'elenco di distribuzione. L'unica opzione di chiusura valida per un elenco di distribuzione è CONONE.

  La chiamata MQPUT1 può essere utilizzata anche per inserire un messaggio in un elenco di distribuzione; la struttura MQOD che definisce le code nell'elenco viene specificata come parametro su tale chiamata.
- Ogni destinazione aperta correttamente nell'elenco di distribuzione viene contata come un handle *separato* quando si verifica se l'applicazione ha superato il numero massimo consentito di handle (consultare l'attributo del gestore code **MaxHandles**). Ciò è vero anche quando due o più destinazioni nell'elenco di distribuzione vengono effettivamente risolte nella stessa coda fisica. Se la chiamata MQOPEN o MQPUT1 per un elenco di distribuzione fa sì che il numero di handle utilizzati dall'applicazione superi *MaxHandles*, la chiamata ha esito negativo con codice di errore RC2017.

- Ogni destinazione aperta correttamente ha il valore del relativo attributo **OpenOutputCount** incrementato di uno. Se due o più destinazioni nell'elenco di distribuzione si risolvono nella stessa coda fisica, il relativo attributo **OpenOutputCount** viene incrementato del numero di destinazioni nell'elenco di distribuzione che si risolvono in tale coda.
- Qualsiasi modifica alle definizioni di coda che avrebbe causato la non validità di un handle se le code fossero state aperte singolarmente (ad esempio, una modifica nel percorso di risoluzione), non fa sì che l'handle dell'elenco di distribuzione diventi non valido. Tuttavia, si verifica un errore per quella particolare coda quando l'handle dell'elenco di distribuzione viene utilizzato su una successiva chiamata MQPUT.
- È valido che un elenco di distribuzione contenga una sola destinazione.

9. Le seguenti note si applicano all'utilizzo delle code cluster.
   - Quando una coda del cluster viene aperta per la prima volta e il gestore code locale non è un gestore code del repository completo, il gestore code locale ottiene le informazioni sulla coda del cluster da un gestore code del repository completo. Quando la rete è occupata, potrebbero essere necessari alcuni secondi perché il gestore code locale riceva le informazioni necessarie dal gestore code del repository. Di conseguenza, l'applicazione che emette la chiamata MQOPEN potrebbe dover attendere fino a 10 secondi prima che il controllo ritorni dalla chiamata MQOPEN. Se il gestore code locale non riceve le informazioni necessarie sulla coda cluster entro questo periodo di tempo, la chiamata ha esito negativo con codice motivo RC2189.
   - Quando una coda del cluster viene aperta e ci sono più istanze della coda nel cluster, l'istanza effettivamente aperta dipende dalle opzioni specificate nella chiamata MQOPEN:
     – Se le opzioni specificate includono una delle seguenti:
       - OOBRW
       - OOINPQ
       - OOINPX
       - OOINPS
       - OSET

       l'istanza della coda cluster aperta deve essere l'istanza locale. Se non è presente alcuna istanza locale della coda, la chiamata MQOPEN non riesce.
     – Se le opzioni specificate non includono nessuna delle precedenti, ma includono una o entrambe le seguenti:
       - OOINQ
       - OOOUT

       l'istanza aperta è l'istanza locale, se presente, e un'istanza remota, altrimenti. Tuttavia, l'istanza scelta dal gestore code può essere modificata da un'uscita del carico di lavoro del cluster (se presente).

   Per ulteriori informazioni sulle code cluster, consultare Code cluster.
10. Alle applicazioni avviate da un controllo trigger viene passato il nome della coda associata all'applicazione quando questa viene avviata. Questo nome coda può essere specificato nel parametro **OBJDSC** per aprire la coda. Per ulteriori dettagli, consultare la descrizione della struttura MQTMC.
11. Quando si utilizza l'opzione OORLOQ, la coda locale viene già restituita quando viene aperta una coda locale, alias o modello, ma non è questo il caso quando, ad esempio, viene aperta una coda remota o una coda cluster non locale; il nome ResolvedQName e ResolvedQMgrvengono immessi con il nome RemoteQName e RemoteQMgrtrovato nella definizione della coda remota o in modo simile con la coda cluster remota scelta. Se OORLOQ viene specificato durante l'apertura, ad esempio, di una coda remota, ResolvedQName sarà ora la coda di trasmissione in cui verranno inseriti i messaggi. Il nome ResolvedQMgrverrà immesso con il nome del gestore code locale che ospita la coda di trasmissione. Se un utente è autorizzato per la ricerca, l'input o l'output su una coda, dispone dell'autorità richiesta

per specificare questo indicatore sulla chiamata MQOPEN. Non è necessaria alcuna autorizzazione speciale.

Parametri

La chiamata MQOPEN ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Handle di connessione.

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

OBJDSC (MQOD) - input/output

Descrittore oggetto.

Si tratta di una struttura che identifica l'oggetto da aprire; consultare “MQOD (Object descriptor) su IBM i” a pagina 1192 per i dettagli.

Se il campo *ODON* nel parametro **OBJDSC** è il nome di una coda modello, una coda locale dinamica viene creato con gli attributi della coda modello; ciò accade indipendentemente dalle opzioni di apertura specificate dal parametro **OPTS** . Le successive operazioni che utilizzano il *HOBJ* restituito dalla chiamata MQOPEN vengono eseguite sulla nuova coda dinamica e non sulla coda modello. Ciò è vero anche per le chiamate MQINQ e MQSET. Il nome della coda modello nel parametro **OBJDSC** viene sostituito con il nome della coda dinamica creata. Il tipo di coda dinamica è determinato dal valore dell'attributo **DefinitionType** della coda modello (consultare “Attributi per le code” a pagina 1406). Per informazioni sulle opzioni di chiusura applicabili alle code dinamiche, consultare la descrizione della chiamata MQCLOSE.

OPTS (numero intero con segno a 10 cifre) - immissione

Opzioni che controllano l'azione di MQOPEN.

È necessario specificare almeno una delle seguenti opzioni:

- OOBRW
- OOINP* (solo uno di questi)
- OOINQ
- OOOUT
- OSET
- OORLQ

È possibile specificare altre opzioni come richiesto. Se sono richieste più opzioni, è possibile aggiungere i valori (non aggiungere la stessa costante più di una volta). Le combinazioni non valide vengono annotate; tutte le altre combinazioni sono valide. Sono consentite solo le opzioni applicabili al tipo di oggetto specificato da *OBJDSC* (consultare Opzioni MQOPEN valide per ciascun tipo di coda).

Opzioni di accesso: le seguenti opzioni controllano il tipo di operazioni che è possibile eseguire sull'oggetto:

OOINPQ
Aprire la coda per richiamare i messaggi utilizzando il valore predefinito definito dalla coda.

La coda viene aperta per essere utilizzata con successive chiamate MQGET. Il tipo di accesso è condiviso o esclusivo, a seconda del valore dell'attributo della coda **DefInputOpenOption** ; consultare “Attributi per le code” a pagina 1406 per i dettagli.

Questa opzione è valida solo per le code locali, alias e modello; non è valida per le code remote, gli elenchi di distribuzione e gli oggetti che non sono code.

OOINPS
Aprire la coda per richiamare i messaggi con accesso condiviso.

La coda viene aperta per essere utilizzata con successive chiamate MQGET. La chiamata può avere esito positivo se la coda è attualmente aperta da questa o un'altra applicazione con OOINPS, ma ha esito negativo con codice motivo RC2042 se la coda è attualmente aperta con OOINPX.

Questa opzione è valida solo per le code locali, alias e modello; non è valida per le code remote, gli elenchi di distribuzione e gli oggetti che non sono code.

OOINPX
Aprire la coda per ottenere i messaggi con accesso esclusivo.

La coda viene aperta per essere utilizzata con successive chiamate MQGET. La chiamata non riesce con codice di errore RC2042 se la coda è attualmente aperta da questa o da un'altra applicazione per l'input di qualsiasi tipo (OOINPS o OOINPX).

Questa opzione è valida solo per le code locali, alias e modello; non è valida per le code remote, gli elenchi di distribuzione e gli oggetti che non sono code.

Le seguenti note si applicano a queste opzioni:

- È possibile specificare solo una di queste opzioni.
- Una chiamata MQOPEN con una di queste opzioni può riuscire anche se l'attributo della coda **InhibitGet** è impostato su QAGETI (anche se le successive chiamate MQGET avranno esito negativo mentre l'attributo è impostato su questo valore).
- Se la coda è definita come non condivisibile (ossia, l'attributo della coda **Shareability** ha il valore QANSHR), i tentativi di aprire la coda per l'accesso condiviso vengono considerati come tentativi di aprire la coda con accesso esclusivo.
- Se una coda alias viene aperta con una di queste opzioni, la verifica per l'uso esclusivo (o per il fatto che un'altra applicazione abbia un uso esclusivo) è rispetto alla coda di base in cui l'alias si risolve.
- Queste opzioni non sono valide se *ODMN* è il nome di un alias del gestore code; ciò è vero anche se il valore dell'attributo **RemoteQMgrName** nella definizione locale di una coda remota utilizzata per l'alias del gestore code è il nome del gestore code locale.

OOBRW
Aprire la coda per esaminare i messaggi.

La coda viene aperta per essere utilizzata con le chiamate MQGET successive con una delle seguenti opzioni:

- GMBRWF
- GMBRWN
- GMBRWC

Ciò è consentito anche se la coda è attualmente aperta per OOINPX. Una chiamata MQOPEN con l'opzione OOBRW stabilisce un cursore di esplorazione e lo posiziona logicamente prima del primo messaggio sulla coda; per ulteriori informazioni, consultare il campo *GMOPT* descritto in "MQGMO (opzioni Get - message) su IBM i" a pagina 1105 .

Questa opzione è valida solo per le code locali, alias e modello; non è valida per code remote, elenchi di distribuzione e oggetti che non sono code. Inoltre, non è valido se *ODMN* è il nome di un alias del gestore code; ciò è vero anche se il valore dell'attributo **RemoteQMgrName** nella definizione locale di una coda remota utilizzata per l'alias del gestore code è il nome del gestore code locale.

OOOUT
Aprire la coda per inserire i messaggi o un argomento o una stringa di argomenti per pubblicare i messaggi.

La coda viene aperta per essere utilizzata con chiamate MQPUT successive.

Una chiamata MQOPEN con questa opzione può avere esito positivo anche se l'attributo della coda **InhibitPut** è impostato su QAPUTI (anche se le chiamate MQPUT successive avranno esito negativo mentre l'attributo è impostato su questo valore).

Questa opzione è valida per tutti i tipi di coda, inclusi gli argomenti e gli elenchi di distribuzione.

OOINQ
Aprire l'oggetto per interrogare gli attributi.

La coda, l'elenco dei nomi, la definizione del processo o il gestore code viene aperto per essere utilizzato con le chiamate MQINQ successive.

Questa opzione è valida per tutti i tipi di oggetto diversi dagli elenchi di distribuzione. Non è valido se *ODMN* è il nome di un alias del gestore code; ciò è vero anche se il valore dell'attributo **RemoteQMgrName** nella definizione locale di una coda remota utilizzata per l'alias del gestore code è il nome del gestore code locale.

OSET
Aprire la coda per impostare gli attributi.

La coda viene aperta per essere utilizzata con le chiamate MQSET successive.

Questa opzione è valida per tutti i tipi di coda diversi dagli elenchi di distribuzione. Non è valido se *ODMN* è il nome di una definizione locale di una coda remota; ciò è vero anche se il valore dell'attributo **RemoteQMgrName** nella definizione locale di una coda remota utilizzata per l'alias del gestore code è il nome del gestore code locale.

Opzioni di bind: le seguenti opzioni si applicano quando l'oggetto che si sta aprendo è una coda del cluster; queste opzioni controllano il collegamento dell'handle della coda a una istanza della coda del cluster:

OOBNDO
Collega l'handle alla destinazione quando la coda è aperta.

Ciò fa sì che il gestore code locale esegua il bind dell'handle di coda a una istanza della coda di destinazione quando la coda viene aperta. Di conseguenza, tutti i messaggi inseriti utilizzando questo handle vengono inviati alla stessa istanza della coda di destinazione e allo stesso instradamento.

Questa opzione è valida solo per le code e interessa solo le code cluster. Se specificata per una coda che non è una coda cluster, l'opzione viene ignorata.

DN OBN
Non eseguire il collegamento a una specifica destinazione.

Questo arresta il gestore code locale che esegue il bind dell'handle della coda a un'istanza della coda di destinazione. Di conseguenza, le successive chiamate MQPUT che utilizzano questo handle possono comportare l'invio di messaggi a *diverse* istanze della coda di destinazione o l'invio alla stessa istanza ma tramite instradamenti differenti. Inoltre, consente all'istanza selezionata di essere modificata successivamente dal gestore code locale, da un gestore code remoto o da un agent MCA (message channel agent), in base alle condizioni di rete.

Nota: Le applicazioni client e server che devono scambiare una *serie* di messaggi per completare una transazione non devono utilizzare OOBNDN (o OOBNDQ quando *DefBind* ha il valore BNDNOT), poiché i messaggi successivi nella serie possono essere inviati a istanze differenti dell'applicazione server.

Se OOBRW o una delle opzioni OOINP* viene specificata per una coda cluster, il gestore code viene forzato a selezionare l'istanza locale della coda cluster. Di conseguenza, il collegamento dell'handle della coda è fisso, anche se è specificato OOBNDN.

Se OOINQ è specificato con OOBNDN, le chiamate MQINQ successive che utilizzano tale handle possono interrogare istanze differenti della coda cluster, anche se di solito tutte le istanze hanno gli stessi valori di attributo.

OOBNDN è valido solo per le code e interessa solo le code cluster. Se specificata per una coda che non è una coda cluster, l'opzione viene ignorata.

OOBNDQ
Utilizzare il binding predefinito per la coda.

Ciò fa sì che il gestore code locale esegua il bind dell'handle di coda nel modo definito dall'attributo della coda **DefBind** . Il valore di questo attributo è BNDOPN o BNDNOT.

OOBNDQ è il valore predefinito se OOBNDO e OOBNDN non sono specificati.

OOBNDQ è definito per aiutare la documentazione del programma. Non è previsto che questa opzione venga utilizzata con una delle altre due opzioni di collegamento, ma poiché il relativo valore è zero tale utilizzo non può essere rilevato.

Opzioni di contesto: le seguenti opzioni controllano l'elaborazione del contesto del messaggio:

OOSAVA
Salva contesto quando viene richiamato il messaggio.

Le informazioni di contesto sono associate a questo handle di coda. Queste informazioni vengono impostate dal contesto di qualsiasi messaggio richiamato utilizzando questo handle. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Queste informazioni di contesto possono essere inoltrate a un messaggio che viene successivamente inserito in una coda utilizzando le chiamate MQPUT o MQPUT1 . Consultare le opzioni PMPASI e PMPASA descritte in “MQPMO (Put - message options) su IBM i” a pagina 1206.

Finché un messaggio non viene richiamato correttamente, il contesto non può essere passato a un messaggio inserito in una coda.

Per un messaggio richiamato utilizzando una delle opzioni di ricerca GMBRW* non vengono salvate le relative informazioni di contesto (anche se i campi di contesto nel parametro **MSGDSC** sono impostati dopo una ricerca).

Questa opzione è valida solo per le code locali, alias e modello; non è valida per code remote, elenchi di distribuzione e oggetti che non sono code. È necessario specificare una delle opzioni OOINP*.

OOPASI
Consenti il passaggio del contesto di identità.

Ciò consente all'opzione PMPASI di essere specificata nel parametro **PMO** quando un messaggio viene inserito su una coda; ciò fornisce al messaggio le informazioni sul contesto di identità da una coda di input aperta con l'opzione OOSAVA. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

È necessario specificare l'opzione OOOUT.

Questa opzione è valida per tutti i tipi di coda, inclusi gli elenchi di distribuzione.

OOPASA
Consenti il passaggio di tutto il contesto.

Ciò consente di specificare l'opzione PMPASA nel parametro **PMO** quando un messaggio viene inserito su una coda; ciò fornisce al messaggio le informazioni sul contesto di origine e di identità da una coda di input aperta con l'opzione OOSAVA. Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Questa opzione implica OOPASI, che non deve quindi essere specificato. È necessario specificare l'opzione OOOUT.

Questa opzione è valida per tutti i tipi di coda, inclusi gli elenchi di distribuzione.

OOSETI
Consenti l'impostazione del contesto di identità.

Ciò consente di specificare l'opzione PMSETI nel parametro **PMO** quando un messaggio viene inserito in una coda; ciò fornisce al messaggio le informazioni di contesto dell'identit ... contenute nel parametro **MSGDSC** specificato nella chiamata MQPUT o MQPUT1 . Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Questa opzione implica OOPASI, che non deve quindi essere specificato. È necessario specificare l'opzione OOOUT.

Questa opzione è valida per tutti i tipi di coda, inclusi gli elenchi di distribuzione.

OOSETA
Consenti l'impostazione di tutto il contesto.

Ciò consente all'opzione PMSETA di essere specificata nel parametro **PMO** quando un messaggio viene inserito in una coda; ciò fornisce al messaggio le informazioni sul contesto di origine e di identità contenute nel parametro **MSGDSC** specificato nella chiamata MQPUT o MQPUT1 . Per ulteriori informazioni sul contesto del messaggio, consultare Contesto del messaggio e Informazioni sul contesto di controllo.

Questa opzione implica le opzioni seguenti, che non è necessario specificare:

- OOPASI
- OOPASA
- OOSETI

È necessario specificare l'opzione OOOUT.

Questa opzione è valida per tutti i tipi di coda, inclusi gli elenchi di distribuzione.

Altre opzioni: le seguenti opzioni controllano il controllo dell'autorizzazione e cosa accade quando il gestore code è in fase di sospensione:

OOALTU
Convalidare con l'identificativo utente specificato.

Ciò indica che il campo *ODAU* nel parametro **OBJDSC** contiene un identificativo utente che deve essere utilizzato per convalidare questa chiamata MQOPEN. La chiamata può avere esito positivo solo se questo *ODAU* è autorizzato ad aprire l'oggetto con le opzioni di accesso specificate, indipendentemente dal fatto che l'identificativo utente con cui l'applicazione è in esecuzione sia autorizzato a farlo. Ciò non si applica alle opzioni di contesto specificate, tuttavia, che sono sempre controllate rispetto all'identificativo utente con cui è in esecuzione l'applicazione.

Questa opzione è valida per tutti i tipi di oggetto.

OOFIQ
Errore se il gestore code è in fase di sospensione.

Questa opzione forza l'esito negativo della chiamata MQOPEN se il gestore code è in stato di sospensione.

Questa opzione è valida per tutti i tipi di oggetto.

OORLQ
Immettere il nome della coda locale aperta.

Questa opzione specifica che ResolvedQName nella struttura MQOD (se disponibile) deve essere immesso con il nome della coda locale che è stata aperta. Il nome ResolvedQMgrverrà immesso in modo simile con il nome del gestore code locale che ospita la coda locale.

Tabella 750. Opzioni MQOPEN valide per ciascun tipo di coda

Opzione	Alias ("1" a pagina 1369)	Locale e modello	Remoto	Cluster non locale	Elenco di distribuzione	Argomento
OOINPQ	✓	✓	-	-	-	-
OOINPS	✓	✓	-	-	-	-
OOINPX	✓	✓	-	-	-	-

Tabella 750. Opzioni MQOPEN valide per ciascun tipo di coda (Continua)						
Opzione	**Alias ("1" a pagina 1369)**	**Locale e modello**	**Remoto**	**Cluster non locale**	**Elenco di distribuzione**	**Argomento**
OOBRW	✓	✓	-	-	-	-
OOOUT	✓	✓	✓	✓	✓	✓
OOINQ	✓	✓	"2" a pagina 1369	✓	-	-
OSET	✓	✓	"2" a pagina 1369	-	-	-
OOBNDO ("3" a pagina 1369)	✓	✓	✓	✓	✓	-
OOBNDN ("3" a pagina 1369)	✓	✓	✓	✓	✓	-
OOBNDQ ("3" a pagina 1369)	✓	✓	✓	✓	✓	-
OOSAVA	✓	✓	-	-	-	-
OOPASI	✓	✓	✓	✓	✓	"5" a pagina 1369
OOPASA	✓	✓	✓	✓	✓	"5" a pagina 1369
OOSETI	✓	✓	✓	✓	✓	"5" a pagina 1369
OOSETA	✓	✓	✓	✓	✓	"5" a pagina 1369
OOALTU	✓	✓	✓	✓	✓	✓
OOFIQ	✓	✓	✓	✓	✓	✓
OORLQ	✓	✓	✓	✓	-	-

Note:

1. La validità delle opzioni per gli alias dipende dalla validità dell'opzione per la coda in cui l'alias si risolve.
2. Questa opzione è valida solo per la definizione locale di una coda remota.
3. Questa opzione può essere specificata per qualsiasi tipo di coda, ma viene ignorata se la coda non è una coda cluster.
4. Questo attributo viene ignorato per un argomento.
5. Questi attributi possono essere utilizzati con un argomento, ma influenzano solo il contesto impostato per il messaggio conservato, non i campi di contesto inviati a qualsiasi sottoscrittore.

HOBJ (numero intero con segno a 10 cifre) - emissione

Handle oggetto.

Questo handle rappresenta l'accesso stabilito all'oggetto. Deve essere specificato nelle successive chiamate di accodamento messaggi che operano sull'oggetto. Cessa di essere valida quando viene

emessa la chiamata MQCLOSE o quando termina l'unità di elaborazione che definisce l'ambito dell'handle.

L'ambito della maniglia è limitato alla più piccola unità di elaborazione parallela supportata dalla piattaforma su cui è in esecuzione l'applicazione; l'handle non è valido al di fuori dell'unità di elaborazione parallela da cui è stata emessa la chiamata MQOPEN:

- Su IBM i, l'ambito dell'handle è il lavoro che emette la chiamata.

CMPCOD (numero intero con segno a 10 cifre) - output

Codice di completamento.

Il valore è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

Dichiarazione RPG

```
     C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     C                     CALLP     MQOPEN(HCONN : OBJDSC : OPTS :
     C                                      HOBJ : CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQOPEN           PR                  EXTPROC('MQOPEN')
D* Connection handle
D HCONN                          10I 0 VALUE
D* Object descriptor
D OBJDSC                        468A
D* Options that control the action of MQOPEN
D OPTS                           10I 0 VALUE
D* Object handle
D HOBJ                           10I 0
D* Completion code
D CMPCOD                         10I 0
D* Reason code qualifying CMPCOD
D REASON                         10I 0
```

IBM i MQPUT (Inserisci messaggio) su IBM i

La chiamata MQPUT inserisce un messaggio in una coda, in un elenco di distribuzione o in un argomento. La coda, l'elenco di distribuzione o l'argomento devono essere già aperti.

- “Dichiarazione RPG” a pagina 1380

Sintassi

MQPUT *(HCONN, HOBJ, MSGDSC, PMO, BUFLEN, BUFFER, CMPCOD, REASON)*

Note d'utilizzo

Argomenti

Le seguenti note si applicano all'uso degli argomenti:

1. Quando si utilizza MQPUT per pubblicare i messaggi su un argomento, dove uno o più sottoscrittori a tale argomento non possono ricevere la pubblicazione a causa di un problema con la coda del sottoscrittore (ad esempio, è piena), il codice motivo restituito alla chiamata MQPUT e il comportamento di consegna dipende dall'impostazione degli attributi PMSGDLV o NPMSGDLV sull'TOPIC. Si noti che la consegna di una pubblicazione alla coda di messaggi non recapitabili quando viene specificato RODLQ o l'eliminazione del messaggio quando viene specificato RODISC, viene considerata una consegna corretta del messaggio. Se nessuna delle pubblicazioni viene consegnata, MQPUT restituirà RC2502. Ciò può verificarsi nei seguenti casi:
   - Un messaggio viene pubblicato in un TOPIC con PMSGDLV o NPMSGDLV (a seconda della persistenza del messaggio) impostato su ALL e qualsiasi sottoscrizione (durevole o meno) ha una coda che non può ricevere la pubblicazione.
   - Un messaggio viene pubblicato in un TOPIC con PMSGDLV o NPMSGDLV (in base alla persistenza del messaggio) impostato su ALLDUR e una sottoscrizione durevole ha una coda che non può ricevere la pubblicazione.

   MQPUT può restituire RCNONE anche se le pubblicazioni non possono essere consegnate ad alcuni sottoscrittori nei seguenti casi:

   - Un messaggio viene pubblicato in un TOPIC con PMSGDLV o NPMSGDLV (a seconda della persistenza del messaggio) impostato su ALLAVAIL e qualsiasi sottoscrizione, durevole o meno, ha una coda che non può ricevere la pubblicazione.
   - Un messaggio viene pubblicato in un TOPIC con PMSGDLV o NPMSGDLV (in base alla persistenza del messaggio) impostato su ALLDUR e una sottoscrizione non durevole ha una coda che non può ricevere la pubblicazione.
2. Se non sono presenti sottoscrittori per l'argomento utilizzato, il messaggio pubblicato non viene inviato ad alcuna coda e viene eliminato. Non fa alcuna differenza se questo messaggio è persistente o non persistente, o se ha una scadenza illimitata o una piccola scadenza, viene ancora scartato se non ci sono sottoscrittori. L'eccezione è se il messaggio deve essere conservato, nel qual caso, sebbene non venga inviato ad alcuna coda di sottoscrittori, viene memorizzato rispetto all'argomento per essere consegnato a qualsiasi nuova sottoscrizione o a qualsiasi sottoscrittore che richieda le pubblicazioni conservate utilizzando MQSUBRQ.

MQPUT e MQPUT1

Sia le chiamate MQPUT che MQPUT1 possono essere utilizzate per inserire i messaggi su una coda; la chiamata da utilizzare dipende dalle circostanze

- La chiamata MQPUT deve essere utilizzata quando più messaggi devono essere inseriti nella *stessa* coda.

  Viene emessa prima una chiamata MQOPEN che specifica l'opzione OOOUT, seguita da una o più richieste MQPUT per aggiungere messaggi alla coda; infine la coda viene chiusa con una chiamata MQCLOSE. Ciò fornisce prestazioni migliori rispetto all'utilizzo ripetuto della chiamata di MQPUT1 .
- La chiamata MQPUT1 deve essere utilizzata quando solo *un* messaggio deve essere inserito in una coda.

Questa chiamata incapsula le chiamate MQOPEN, MQPUT e MQCLOSE in una singola chiamata, riducendo il numero di chiamate che devono essere emesse.

Code di destinazione

Se un'applicazione inserisce una sequenza di messaggi nella stessa coda senza utilizzare i gruppi di messaggi, l'ordine di tali messaggi viene conservato se vengono soddisfatte le seguenti condizioni. Alcune condizioni si applicano alle code di destinazione locali e remote; altre condizioni si applicano solo alle code di destinazione remote.

Condizioni per le code di destinazione locali e remote

- Tutte le chiamate MQPUT si trovano all'interno della stessa unità di lavoro o nessuna di esse si trova all'interno di un'unità di lavoro.

  Quando i messaggi vengono inseriti in una particolare coda all'interno di una singola unità di lavoro, i messaggi provenienti da altre applicazioni potrebbero essere intervallati dalla sequenza di messaggi sulla coda.
- Tutte le chiamate MQPUT vengono effettuate utilizzando la stessa gestione oggetto *HOBJ*.

  In alcuni ambienti, la sequenza dei messaggi viene conservata anche quando vengono utilizzati diversi handle di oggetto, a condizione che le chiamate vengano effettuate dalla stessa applicazione. Il significato di "stessa applicazione" è determinato dall'ambiente:

  – Su IBM i, l'applicazione è il lavoro.
- Tutti i messaggi hanno la stessa priorità.

Ulteriori condizioni per le code di destinazione remota

- Esiste un solo percorso dal gestore code di invio al gestore code di destinazione.

  Se esiste la possibilità che alcuni messaggi nella sequenza possano andare su un percorso diverso (ad esempio, a causa della riconfigurazione, del bilanciamento del traffico o della selezione del percorso in base alla dimensione del messaggio), l'ordine dei messaggi nel gestore code di destinazione non può essere garantito.
- I messaggi non vengono inseriti temporaneamente nelle code di messaggi non recapitabili nei gestori code di invio, intermedi o di destinazione.

  Se uno o più messaggi vengono inseriti temporaneamente in una coda di messaggi non recapitabili (ad esempio, perché una coda di trasmissione o la coda di destinazione è temporaneamente piena), i messaggi possono arrivare sulla coda di destinazione fuori sequenza.
- I messaggi sono tutti persistenti o non persistenti.

  Se un canale sull'instradamento tra i gestori code di invio e di destinazione ha il proprio attributo **CDNPM** impostato su NPFAST, i messaggi non persistenti possono saltare in anticipo rispetto ai messaggi persistenti, determinando la mancata conservazione dell'ordine dei messaggi persistenti relativi ai messaggi non persistenti. Tuttavia, l'ordine dei messaggi persistenti relativi l'uno all'altro e dei messaggi non persistenti relativi l'uno all'altro, viene conservato.

Se queste condizioni non vengono soddisfatte, è possibile utilizzare i gruppi di messaggi per conservare l'ordine dei messaggi, ma si noti che ciò richiede che le applicazioni di invio e di ricezione utilizzino il supporto di raggruppamento dei messaggi. Per ulteriori informazioni sui gruppi di messaggi, consultare:

- Campo *MDMFL* in MQMD
- Opzione PMLOGO in MQPMO
- Opzione GMLOGO in MQGMO

Liste di distribuzione

Le seguenti note si applicano all'uso degli elenchi di distribuzione.

1. I messaggi possono essere inseriti in un elenco di distribuzione utilizzando version-1 o version-2 MQPMO. Se viene utilizzato un MQPMO version-1 (o un MQPMO version-2 con *PMREC* uguale a zero), l'applicazione non può fornire record di messaggi put o record di risposta. Ciò significa che non sarà possibile identificare le code che rilevano errori, se il messaggio viene inviato con esito positivo ad alcune code nell'elenco di distribuzione e non ad altre.

   Se l'applicazione fornisce record di messaggi di inserimento o record di risposta, il campo *PMVER* deve essere impostato su PMVER2.

   Un MQPMO version-2 può essere utilizzato anche per inviare messaggi a una coda singola che non si trova in un elenco di distribuzione, verificando che *PMREC* sia zero.
2. I parametri del codice di completamento e del codice di errore sono impostati come segue:
   - Se gli inserimenti nelle code nell'elenco di distribuzione hanno esito positivo o negativo nello stesso modo, i parametri del codice di completamento e del codice motivo vengono impostati per descrivere il risultato comune. I record di risposta MQRR (se forniti dall'applicazione) non sono impostati in questo caso.

     Ad esempio, se ogni inserimento ha esito positivo, il codice di completamento è impostato su CCOK e il codice di errore è RCNONE; se ogni inserimento ha esito negativo perché tutte le code sono bloccate per gli inserimenti, i parametri sono impostati su CCFAIL e RC2051.
   - Se gli inserimenti nelle code nell'elenco di distribuzione non hanno esito positivo o negativo nello stesso modo:
     - Il parametro del codice di completamento è impostato su CCWARN se almeno un inserimento ha avuto esito positivo e su CCFAIL se tutti hanno avuto esito negativo.
     - Il parametro codice di errore è impostato su RC2136.
     - I record di risposta (se forniti dall'applicazione) sono impostati sui codici di completamento individuali e sui codici motivo per le code nell'elenco di distribuzione.

     Se l'inserimento in una destinazione non riesce perché l'apertura per tale destinazione non è riuscita, i campi nel record di risposta sono impostati su CCFAIL e RC2137; tale destinazione è inclusa in *PMIDC*.
3. Se una destinazione nell'elenco di distribuzione si risolve in una coda locale, il messaggio viene posizionato su tale coda in formato normale (cioè, non come un messaggio dell'elenco di distribuzione). Se più di una destinazione si risolve nella stessa coda locale, viene inserito un messaggio nella coda per ciascuna di tali destinazioni.

   Se una destinazione nell'elenco di distribuzione si risolve in una coda remota, un messaggio viene inserito nella coda di trasmissione appropriata. Quando più destinazioni si risolvono nella stessa coda di trasmissione, un singolo messaggio dell'elenco di distribuzione contenente tali destinazioni può essere inserito nella coda di trasmissione, anche se tali destinazioni non erano adiacenti nell'elenco di destinazione fornito dall'applicazione. Tuttavia, questa operazione può essere eseguita solo se la coda di trasmissione supporta i messaggi dell'elenco di distribuzione (consultare l'attributo della coda **DistLists** descritto in "Attributi per le code" a pagina 1406).

   Se la coda di trasmissione non supporta gli elenchi di distribuzione, una copia del messaggio in formato normale viene inserita nella coda di trasmissione per ogni destinazione che utilizza tale coda di trasmissione.

   Se un elenco di distribuzione con i dati del messaggio dell'applicazione è troppo grande per una coda di trasmissione, il messaggio dell'elenco di distribuzione viene suddiviso in messaggi dell'elenco di distribuzione più piccoli, ciascuno contenente meno destinazioni. Se i dati del messaggio dell'applicazione si adattano solo alla coda, i messaggi dell'elenco di distribuzione non possono essere utilizzati e il gestore code genera una copia del messaggio in formato normale per ciascuna destinazione che utilizza tale coda di trasmissione.

   Se destinazioni differenti hanno una priorità o una persistenza del messaggio differenti (ciò può verificarsi quando l'applicazione specifica PRQDEF o PEQDEF), i messaggi non vengono conservati nello stesso messaggio dell'elenco di distribuzione. Invece, il gestore code genera tutti i messaggi dell'elenco di distribuzione necessari per soddisfare i diversi valori di priorità e persistenza.

4. Un inserimento in un elenco di distribuzione potrebbe risultare in:
   - Un singolo messaggio dell'elenco di distribuzione oppure
   - Un numero di messaggi di elenco di distribuzione più piccoli, oppure
   - Una combinazione di messaggi dell'elenco di distribuzione e messaggi normali oppure
   - Solo messaggi normali.

   Quale dei precedenti si verifica dipende dal fatto che:
   - Le destinazioni nell'elenco sono locali, remote o miste.
   - Le destinazioni hanno la stessa priorità e persistenza del messaggio.
   - Le code di trasmissione possono contenere i messaggi dell'elenco di distribuzione.
   - La lunghezza massima dei messaggi delle code di trasmissione è sufficiente per contenere il messaggio in formato di elenco di distribuzione.

   Tuttavia, indipendentemente da quale di queste ricorre, ogni messaggio *fisico* risultante (ovvero, ogni messaggio normale o messaggio dell'elenco di distribuzione risultante dall'inserimento) conta come solo *un* messaggio quando:
   - Verifica se l'applicazione ha superato il numero massimo consentito di messaggi in un'unità di lavoro (consultare l'attributo del gestore code **MaxUncommittedMsgs**).
   - Verifica se le condizioni di attivazione sono soddisfatte.
   - Incrementando le profondità della coda e verificando se la profondità massima della coda delle code viene superata.
5. Qualsiasi modifica alle definizioni di coda che avrebbe causato la non validità di un handle se le code fossero state aperte singolarmente (ad esempio, una modifica nel percorso di risoluzione), non fa sì che l'handle dell'elenco di distribuzione diventi non valido. Tuttavia, si verifica un errore per quella particolare coda quando l'handle dell'elenco di distribuzione viene utilizzato su una successiva chiamata MQPUT.

Intestazioni

Se un messaggio viene inserito con una o più strutture di intestazione IBM MQ all'inizio dei dati del messaggio dell'applicazione, il gestore code esegue determinati controlli sulle strutture di intestazione per verificarne la validità. Se il gestore code rileva un errore, la chiamata ha esito negativo con un codice motivo appropriato. I controlli effettuati variano in funzione delle strutture particolari presenti. Inoltre, i controlli vengono eseguiti solo se viene utilizzato un MQMD version-2 o successivo nella chiamata MQPUT o MQPUT1 ; i controlli non vengono eseguiti se viene utilizzato un MQMD version-1 , anche se un MQMDE è presente all'inizio dei dati del messaggio dell'applicazione.

Le seguenti strutture di intestazione IBM MQ vengono convalidate completamente dal gestore code MQDH, MQMDE.

Per altre strutture di intestazione IBM MQ , il gestore code esegue una convalida, ma non controlla tutti i campi. Le strutture non supportate dal gestore code locale e le strutture che seguono il primo MQDLH nel messaggio non vengono convalidate.

Oltre ai controlli generali sui campi nelle strutture IBM MQ , devono essere soddisfatte le seguenti condizioni:

- Una struttura IBM MQ non deve essere suddivisa su due o più segmenti; la struttura deve essere interamente contenuta in un segmento.
- La somma delle lunghezze delle strutture in un messaggio PCF deve essere uguale alla lunghezza specificata dal parametro **BUFLEN** nella chiamata MQPUT o MQPUT1 . Un messaggio PCF è un messaggio che ha uno dei seguenti nomi di formato:
  - MN FMADM
  - FMEVNT
  - FMPCF

- Le strutture IBM MQ non devono essere troncate, tranne nelle seguenti situazioni in cui sono consentite le strutture troncate:
  - Messaggi che sono messaggi di report.
  - Messaggi PCF.
  - Messaggi contenenti una struttura di MQDLH. (Le strutture *che seguono* il primo MQDLH possono essere troncate; le strutture che precedono MQDLH non possono.)

Memorizza nel buffer

Il parametro **BUFFER** mostrato nell'esempio di programmazione RPG viene dichiarato come stringa; ciò limita la lunghezza massima del parametro a 256 byte. Se è richiesto un buffer più grande, il parametro deve essere dichiarato come una struttura o come un campo in un file fisico. Ciò aumenterà la lunghezza massima possibile a circa 32 KB.

Parametri

La chiamata MQPUT ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Handle di connessione.

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

HOBJ (numero intero con segno a 10 cifre) - immissione

Handle oggetto.

Questo handle rappresenta la coda a cui viene aggiunto il messaggio o l'argomento in cui viene pubblicato il messaggio. Il valore *HOBJ* è stato restituito da una precedente chiamata MQOPEN che specificava l'opzione OOOUT.

MSGDSC (MQMD) - input/output

Descrittore del messaggio.

Questa struttura descrive gli attributi del messaggio che si sta inviando e riceve informazioni sul messaggio una volta completata la richiesta di inserimento. Vedi “MQMD (Message Descriptor) su IBM i” a pagina 1140 per i dettagli.

Se l'applicazione fornisce un MQMD version-1 , ai dati del messaggio può essere anteposto un prefisso con una struttura MQMDE per specificare i valori per i campi esistenti in MQMD version-2 ma non in version-1. Il campo *MDFMT* in MQMD deve essere impostato su FMMDE per indicare che è presente MQMDE. Per ulteriori dettagli, vedere “MQMDE (estensione descrittore messaggi) su IBM i” a pagina 1186.

PMO (MQPM) - input/output

Opzioni che controllano l'azione di MQPUT.

Vedi “MQPMO (Put - message options) su IBM i” a pagina 1206 per i dettagli.

BUFLEN (numero intero con segno a 10 cifre) - input

Lunghezza del messaggio in *BUFFER*.

Zero è valido e indica che il messaggio non contiene dati dell'applicazione. Il limite superiore per *BUFLEN* dipende da diversi fattori:

- Se la coda di destinazione è una coda condivisa, il limite superiore è 63 KB (64 512 byte).
- Se la destinazione è una coda locale o si risolve in una coda locale (ma non è una coda condivisa), il limite superiore dipende dal fatto che:
  - Il gestore code locale supporta la segmentazione.

– L'applicazione mittente specifica l'indicatore che consente al gestore code di segmentare il messaggio. Questo indicatore è MFSEGA e può essere specificato in un MQMD version-2 o in un MQMDE utilizzato con un MQMD version-1 .

Se vengono soddisfatte entrambe le condizioni, *BUFLEN* non può superare 999 999 999 meno il valore del campo *MDOFF* in MQMD. Il messaggio logico più lungo che può essere inserito è quindi 999 999 999 byte (quando *MDOFF* è zero). Tuttavia, i vincoli delle risorse imposti dal sistema operativo o dall'ambiente in cui l'applicazione è in esecuzione possono risultare in un limite inferiore.

Se una o entrambe le condizioni precedentemente descritte non vengono soddisfatte, *BUFLEN* non può superare il valore più piccolo dell'attributo **MaxMsgLength** della coda e dell'attributo **MaxMsgLength** del gestore code.

- Se la destinazione è una coda remota o si risolve in una coda remota, si applicano le condizioni per le code locali, *ma a ogni gestore code attraverso il quale il messaggio deve passare per raggiungere la coda di destinazione* ; in particolare:

  1. La coda di trasmissione locale utilizzata per memorizzare temporaneamente il messaggio nel gestore code locale
  2. Code di trasmissione intermedie (se presenti) utilizzate per memorizzare il messaggio nei gestori code sull'instradamento tra i gestori code locali e di destinazione
  3. La coda di destinazione sul gestore code di destinazione

  Il messaggio più lungo che può essere inserito è quindi regolato dal più restrittivo di queste code e gestori code.

  Quando un messaggio si trova su una coda di trasmissione, ulteriori informazioni si trovano con i dati del messaggio e ciò riduce la quantità di dati dell'applicazione che possono essere trasmessi. In questa situazione si consiglia di sottrarre i byte LNMHD dai valori *MaxMsgLength* delle code di trasmissione quando si determina il limite per *BUFLEN*.

  Nota: Solo la mancata conformità con la condizione 1 può essere diagnosticata in modo sincrono (con codice di errore RC2030 o RC2031) quando il messaggio viene inserito. Se le condizioni 2 o 3 non vengono soddisfatte, il messaggio viene reindirizzato a una coda di messaggi non recapitabili (messaggi non recapitabili), su un gestore code intermedio o sul gestore code di destinazione. In questo caso, viene generato un messaggio di report se richiesto dal mittente.

BUFFER (stringa a 1 byte x BUFLEN) - input

Dati del messaggio.

Si tratta di un buffer contenente i dati dell'applicazione da inviare. Il buffer deve essere allineato su un limite appropriato alla natura dei dati nel messaggio. L'allineamento a 4 - byte deve essere adatto per la maggior parte dei messaggi (inclusi i messaggi che contengono le strutture di intestazioni di MQ), ma alcuni messaggi potrebbero richiedere un allineamento più rigoroso. Ad esempio, un messaggio contenente un numero intero binario a 64 bit potrebbe richiedere un allineamento a 8 byte.

Se *BUFFER* contiene dati carattere, dati numerici o entrambi, i campi *MDCSI* e *MDENC* nel parametro **MSGDSC** devono essere impostati sui valori appropriati per i dati; ciò consentirà al destinatario del messaggio di convertire i dati (se necessario) nella serie di caratteri e nella codifica utilizzati dal destinatario.

Nota: Tutti gli altri parametri nella chiamata MQPUT devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e codificare il gestore code locale fornito da ENNAT.

CMPCOD (numero intero con segno a 10 cifre) - output

Codice di completamento.

Il valore è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Codice di errore *CMPCOD*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCWARN:

RC2104
(2104, X'838 ') Opzione prospetto nel descrittore del messaggio non riconosciuta.

RC2136
(2136, X'858 ') Sono stati restituiti più codici di errore.

Se *CMPCOD* è CCFAIL:

RC2004
(2004, X'7D4') Parametro del buffer non valido.

RC2005
(2005, X'7D5') Parametro di lunghezza del buffer non valido.

RC2009
(2009, X'7D9') Connessione al gestore code persa.

RC2013
(2013, X'7DD') Scadenza non valida.

RC2014
(2014, X'7DE') Codice feedback non valido.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2019
(2019, X'7E3') Handle oggetto non valido.

RC2024
(2024, X'7E8') Non è possibile gestire ulteriori messaggi all'interno dell'unità di lavoro corrente.

RC2026
(2026, X'7EA') Descrittore messaggio non valido.

RC2027
(2027, X'7EB') Coda reply - to mancante.

RC2029
(2029, X'7ED') Tipo di messaggio nel descrizione del messaggio non valido.

RC2030
(2030, X'7EE') La lunghezza del messaggio è maggiore del massimo consentito per la coda.

RC2031
(2031, X'7EF') La lunghezza del messaggio è maggiore del massimo consentito per il gestore code.

RC2039
(2039, X'7F7') Coda non aperta per l'emissione.

RC2041
(2041, X'7F9') Definizione oggetto modificata dall'apertura.

RC2046
(2046, X'7FE') Opzioni non valide o non congruenti.

RC2047
(2047, X'7FF') Persistenza non valida.

RC2048
(2048, X'800 ') La coda non supporta i messaggi persistenti.

RC2050
(2050, X'802 ') Priorità messaggio non valida.

RC2051
(2051, X'803 ') Chiamate Put inibite per la coda.

RC2052
(2052, X'804 ') La coda è stata eliminata.

RC2053
(2053, X'805 ') La coda contiene già il numero massimo di messaggi.

RC2056
(2056, X'808 ') Nessuno spazio disponibile sul disco per la coda.

RC2058
(2058, X'80A') Nome gestore code non valido o sconosciuto.

RC2059
(2059, X'80B') Gestore code non disponibile per la connessione.

RC2061
(2061, X'80D') Le opzioni del prospetto nel descrizione del messaggio non sono valide.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2072
(2072, X'818 ') Supporto punto di sincronizzazione non disponibile.

RC2093
(2093, X'82D') Coda non aperta per passare tutto il contesto.

RC2094
(2094, X'82E') Coda non aperta per il contesto di identità del passaggio.

RC2095
(2095, X'82F') Coda non aperta per impostare tutto il contesto.

RC2096
(2096, X'830 ') Coda non aperta per il contesto di identità impostato.

RC2097
(2097, X'831 ') L'handle di coda a cui si fa riferimento non salva il contesto.

RC2098
(2098, X'832 ') Contesto non disponibile per la gestione code a cui si fa riferimento.

RC2101
(2101, X'835 ') Oggetto danneggiato.

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2135
(2135, X'857 ') Struttura intestazione di distribuzione non valida.

RC2136
(2136, X'858 ') Sono stati restituiti più codici di errore.

RC2137
(2137, X'859 ') Oggetto non aperto correttamente.

RC2149
(2149, X'865 ') Strutture PCF non valide.

RC2154
(2154, X'86A') Numero di record presenti non valido.

RC2156
(2156, X'86C') Record di risposta non validi.

RC2158
(2158, X'86E') Indicatori del record del messaggio Put non validi.

RC2159
(2159, X'86F') Inserisci record messaggio non validi.

RC2161
(2161, X'871 ') Gestore code in fase di sospensione.

RC2162
(2162, X'872 ') Chiusura del gestore code.

RC2173
(2173, X'87D') Struttura delle opzioni Put - message non valida.

RC2185
(2185, X'889 ') Specifica di persistenza incongruente.

RC2188
(2188, X'88C') Chiamata rifiutata dall'uscita del carico di lavoro del cluster.

RC2189
(2189, X'88D') Risoluzione del nome cluster non riuscita.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

RC2219
(2219, X'8AB') Chiamata MQI reimmessa prima del completamento della chiamata precedente.

RC2241
(2241, X'8C1') Gruppo di messaggi non completo.

RC2242
(2242, X'8C2') Messaggio logico non completo.

RC2245
(2245, X'8C5') Specifica dell'unità di lavoro non congruente.

RC2248
(2248, X'8C8') Estensione descrittore messaggio non valida.

RC2249
(2249, X'8C9') Indicatori messaggio non validi.

RC2250
(2250, X'8CA') Numero di sequenza messaggio non valido.

RC2251
(2251, X'8CB') Scostamento segmento messaggio non valido.

RC2252
(2252, X'8CC') Lunghezza originale non valida.

RC2253
(2253, X'8CD') La lunghezza dei dati nel segmento del messaggio è zero.

RC2255
(2255, X'8CF') Unità di lavoro non disponibile per il gestore code.

RC2257
(2257, X'8D1') Versione non corretta di MQMD fornita.

RC2258
(2258, X'8D2') Identificativo gruppo non valido.

RC2266
(2266, X'8DA') Uscita carico di lavoro cluster non riuscita.

RC2269
(2269, X'8DD') Errore di risorsa cluster.

RC2270
(2270, X'8DE') Nessuna coda di destinazione disponibile.

RC2420
(2420) È stata emessa una chiamata MQPUT, ma i dati del messaggio contengono una struttura MQEPH non valida.

RC2479
(2479, X'9AF') Impossibile conservare la pubblicazione.

RC2480
(2480, X'9B0') Il tipo di destinazione è stato modificato: la coda alias faceva riferimento a una coda ma ora fa riferimento a una sezione.

RC2502
(2502, X'9C6') Pubblicazione non riuscita e la pubblicazione non è stata consegnata ad alcun sottoscrittore

RC2551
(2551, X'9F7') La stringa di selezione specificata non è disponibile.

RC2554
(2554, X'9FA') Non è stato possibile analizzare il contenuto del messaggio per stabilire se il messaggio deve essere consegnato a un sottoscrittore con un selettore di messaggi esteso.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                    CALLP     MQPUT(HCONN : HOBJ : MSGDSC : PMO :
C                                    BUFLEN : BUFFER : CMPCOD :
C                                    REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQPUT            PR                  EXTPROC('MQPUT')
D* Connection handle
D HCONN                          10I 0 VALUE
D* Object handle
D HOBJ                           10I 0 VALUE
D* Message descriptor
D MSGDSC                        364A
D* Options that control the action of MQPUT
D PMO                           200A
D* Length of the message in Buffer
D BUFLEN                         10I 0 VALUE
D* Message data
D BUFFER                           *   VALUE
D* Completion code
D CMPCOD                         10I 0
D* Reason code qualifying CMPCOD
D REASON                         10I 0
```

IBM i MQPUT1 (Inserisci un messaggio) su IBM i

La chiamata MQPUT1 inserisce un messaggio in una coda o in un elenco di distribuzione o in un argomento. Non è necessario che la coda, l'elenco di distribuzione o l'argomento siano aperti.

Sintassi

MQPUT1 *(HCONN, OBJDSC, MSGDSC, PMO, BUFLEN, BUFFER, CMPCOD, REASON)*

Note d'utilizzo

1. Entrambe le chiamate MQPUT e MQPUT1 possono essere utilizzate per inserire i messaggi su una coda; la chiamata da utilizzare dipende dalle circostanze:
   - La chiamata MQPUT deve essere utilizzata quando più messaggi devono essere inseriti nella *stessa* coda.

     Viene emessa prima una chiamata MQOPEN che specifica l'opzione OOOUT, seguita da una o più richieste MQPUT per aggiungere messaggi alla coda; infine la coda viene chiusa con una chiamata MQCLOSE. Ciò fornisce prestazioni migliori rispetto all'utilizzo ripetuto della chiamata di MQPUT1 .
   - La chiamata MQPUT1 deve essere utilizzata quando solo *un* messaggio deve essere inserito in una coda.

     Questa chiamata incapsula le chiamate MQOPEN, MQPUT e MQCLOSE in una singola chiamata, riducendo il numero di chiamate che devono essere emesse.
2. Se un'applicazione inserisce una sequenza di messaggi nella stessa coda senza utilizzare i gruppi di messaggi, l'ordine di tali messaggi viene conservato se sono soddisfatte determinate condizioni. Tuttavia, nella maggior parte degli ambienti la chiamata MQPUT1 non soddisfa tali condizioni e quindi non conserva l'ordine dei messaggi. La chiamata MQPUT deve essere utilizzata in questi ambienti. Per i dettagli, consultare le note di utilizzo nella descrizione della chiamata MQPUT.
3. La chiamata MQPUT1 può essere utilizzata per inserire i messaggi negli elenchi di distribuzione. Per informazioni generali su questo argomento, consultare le note di utilizzo per le chiamate MQOPEN e MQPUT.

   Le seguenti differenze si applicano quando si utilizza la chiamata MQPUT1 :

   a. Se i record di risposta MQRR sono forniti dall'applicazione, devono essere forniti utilizzando la struttura MQOD; non possono essere forniti utilizzando la struttura MQPMO.

   b. Il codice motivo RC2137 non viene mai restituito da MQPUT1 nei record di risposta; se una coda non riesce ad aprirsi, il record di risposta per tale coda contiene il codice motivo effettivo risultante dall'operazione di apertura.

      Se un'operazione di apertura per una coda riesce con un codice di completamento CCWARN, il codice di completamento e il codice motivo nel record di risposta per tale coda vengono sostituiti dai codici di completamento e motivo risultanti dall'operazione di inserimento.

      Come per le chiamate MQOPEN e MQPUT, il gestore code imposta i record di risposta (se forniti) solo quando il risultato della chiamata non è lo stesso per tutte le code nell'elenco di distribuzione; ciò è indicato dalla chiamata che viene completata con il codice motivo RC2136.
4. Se la chiamata MQPUT1 viene utilizzata per inserire un messaggio in una coda cluster, la chiamata si comporta come se OOBNDN fosse stato specificato nella chiamata MQOPEN.
5. Se un messaggio viene inserito con una o più strutture di intestazione IBM MQ all'inizio dei dati del messaggio dell'applicazione, il gestore code esegue determinati controlli sulle strutture di intestazione per verificarne la validità. Per ulteriori informazioni, consultare le note di utilizzo per la chiamata MQPUT.
6. Se si verifica più di una delle situazioni di avvertenza (vedere il parametro **CMPCOD**), il codice motivo restituito è il *primo* nel seguente elenco che si applica:

   a. RC2136

   b. RC2242

   c. RC2241

   d. RC2049 o RC2104

7. Il parametro **BUFFER** mostrato nell'esempio di programmazione RPG viene dichiarato come stringa; ciò limita la lunghezza massima del parametro a 256 byte. Se è richiesto un buffer più grande, il parametro deve essere dichiarato come una struttura o come un campo in un file fisico. Ciò aumenterà la lunghezza massima possibile a circa 32 KB.

Parametri

La chiamata MQPUT1 presenta i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Handle di connessione.

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

OBJDSC (MQOD) - input/output

Descrittore oggetto.

Questa è una struttura che identifica la coda a cui viene aggiunto il messaggio. Vedi “MQOD (Object descriptor) su IBM i” a pagina 1192 per i dettagli.

L'utente deve essere autorizzato ad aprire la coda per l'emissione. La coda **non** deve essere una coda modello.

MSGDSC (MQMD) - input/output

Descrittore del messaggio.

Questa struttura descrive gli attributi del messaggio che si sta inviando e riceve le informazioni di feedback una volta completata la richiesta di inserimento. Vedi “MQMD (Message Descriptor) su IBM i” a pagina 1140 per i dettagli.

Se l'applicazione fornisce un MQMD version-1 , ai dati del messaggio può essere anteposto un prefisso con una struttura MQMDE per specificare i valori per i campi esistenti in MQMD version-2 ma non in version-1. Il campo *MDFMT* in MQMD deve essere impostato su FMMDE per indicare che è presente MQMDE. Per ulteriori dettagli, vedere “MQMDE (estensione descrittore messaggi) su IBM i” a pagina 1186.

PMO (MQPM) - input/output

Opzioni che controllano l'azione di MQPUT1.

Vedi “MQPMO (Put - message options) su IBM i” a pagina 1206 per i dettagli.

BUFLEN (numero intero con segno a 10 cifre) - input

Lunghezza del messaggio in *BUFFER*.

Zero è valido e indica che il messaggio non contiene dati dell'applicazione. Il limite superiore dipende da vari fattori; per ulteriori informazioni, consultare la descrizione del parametro **BUFLEN** della chiamata MQPUT.

BUFFER (stringa a 1 byte x BUFLEN) - input

Dati del messaggio.

Si tratta di un buffer contenente i dati del messaggio dell'applicazione da inviare. Il buffer deve essere allineato su un limite appropriato alla natura dei dati nel messaggio. L'allineamento a 4 byte dovrebbe essere adatto per la maggior parte dei messaggi (inclusi i messaggi che contengono le strutture di intestazione IBM MQ), ma alcuni messaggi potrebbero richiedere un allineamento più rigoroso. Ad esempio, un messaggio contenente un numero intero binario a 64 bit potrebbe richiedere un allineamento a 8 byte.

Se *BUFFER* contiene dati carattere, dati numerici o entrambi, i campi *MDCSI* e *MDENC* nel parametro **MSGDSC** devono essere impostati sui valori appropriati per i dati; ciò consentirà al destinatario del

messaggio di convertire i dati (se necessario) nella serie di caratteri e nella codifica utilizzati dal destinatario.

Nota: Tutti gli altri parametri sulla chiamata MQPUT1 devono essere nella serie di caratteri fornita dall'attributo del gestore code **CodedCharSetId** e dalla codifica del gestore code locale fornita da ENNAT.

CMPCOD (numero intero con segno a 10 cifre) - output

Codice di completamento.

Il valore è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Codice di errore *CMPCOD*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCWARN:

RC2104
(2104, X'838 ') Opzione prospetto nel descrittore del messaggio non riconosciuta.

RC2136
(2136, X'858 ') Sono stati restituiti più codici di errore.

RC2049
(2049, X'801 ') La priorità del messaggio supera il valore massimo supportato.

RC2241
(2241, X'8C1') Gruppo di messaggi non completo.

RC2242
(2242, X'8C2') Messaggio logico non completo.

Se *CMPCOD* è CCFAIL:

RC2001
(2001, X'7D1') La coda di base dell'alias non è un tipo valido.

RC2004
(2004, X'7D4') Parametro del buffer non valido.

RC2005
(2005, X'7D5') Parametro di lunghezza del buffer non valido.

RC2009
(2009, X'7D9') Connessione al gestore code persa.

RC2013
(2013, X'7DD') Scadenza non valida.

RC2014
(2014, X'7DE') Codice feedback non valido.

RC2017
(2017, X'7E1') Nessun ulteriore handle disponibile.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2024
(2024, X'7E8') Non è possibile gestire ulteriori messaggi all'interno dell'unità di lavoro corrente.

RC2026
(2026, X'7EA') Descrittore messaggio non valido.

RC2027
(2027, X'7EB') Coda reply - to mancante.

RC2029
(2029, X'7ED') Tipo di messaggio nel descrizione del messaggio non valido.

RC2030
(2030, X'7EE') La lunghezza del messaggio è maggiore del massimo consentito per la coda.

RC2031
(2031, X'7EF') La lunghezza del messaggio è maggiore del massimo consentito per il gestore code.

RC2035
(2035, X'7F3') Non autorizzato per l'accesso.

RC2042
(2042, X'7FA') Oggetto già aperto con opzioni in conflitto.

RC2043
(2043, X'7FB') Tipo oggetto non valido.

RC2044
(2044, X'7FC') Struttura descrittore oggetto non valida.

RC2046
(2046, X'7FE') Opzioni non valide o non congruenti.

RC2047
(2047, X'7FF') Persistenza non valida.

RC2048
(2048, X'800 ') La coda non supporta i messaggi persistenti.

RC2050
(2050, X'802 ') Priorità messaggio non valida.

RC2051
(2051, X'803 ') Chiamate Put inibite per la coda.

RC2052
(2052, X'804 ') La coda è stata eliminata.

RC2053
(2053, X'805 ') La coda contiene già il numero massimo di messaggi.

RC2056
(2056, X'808 ') Nessuno spazio disponibile sul disco per la coda.

RC2057
(2057, X'809 ') Tipo coda non valido.

RC2058
(2058, X'80A') Nome gestore code non valido o sconosciuto.

RC2059
(2059, X'80B') Gestore code non disponibile per la connessione.

RC2061
(2061, X'80D') Le opzioni del prospetto nel descrizione del messaggio non sono valide.

RC2063
(2063, X'80F') Si è verificato un errore di sicurezza.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2072
(2072, X'818 ') Supporto punto di sincronizzazione non disponibile.

RC2082
(2082, X'822 ') Coda di base alias sconosciuta.

RC2085
(2085, X'825 ') Nome oggetto sconosciuto.

RC2086
(2086, X'826 ') Gestore code oggetti sconosciuto.

RC2087
(2087, X'827 ') Gestore code remoto sconosciuto.

RC2091
(2091, X'82B') Coda di trasmissione non locale.

RC2092
(2092, X'82C') Coda di trasmissione con utilizzo errato.

RC2097
(2097, X'831 ') L'handle di coda a cui si fa riferimento non salva il contesto.

RC2098
(2098, X'832 ') Contesto non disponibile per la gestione code a cui si fa riferimento.

RC2101
(2101, X'835 ') Oggetto danneggiato.

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2135
(2135, X'857 ') Struttura intestazione di distribuzione non valida.

RC2136
(2136, X'858 ') Sono stati restituiti più codici di errore.

RC2149
(2149, X'865 ') Strutture PCF non valide.

RC2154
(2154, X'86A') Numero di record presenti non valido.

RC2155
(2155, X'86B') Record oggetto non validi.

RC2156
(2156, X'86C') Record di risposta non validi.

RC2158
(2158, X'86E') Indicatori del record del messaggio Put non validi.

RC2159
(2159, X'86F') Inserisci record messaggio non validi.

RC2161
(2161, X'871 ') Gestore code in fase di sospensione.

RC2162
(2162, X'872 ') Chiusura del gestore code.

RC2173
(2173, X'87D') Struttura delle opzioni Put - message non valida.

RC2184
(2184, X'888 ') Nome coda remota non valido.

RC2188
(2188, X'88C') Chiamata rifiutata dall'uscita del carico di lavoro del cluster.

RC2189
(2189, X'88D') Risoluzione del nome cluster non riuscita.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

RC2196
(2196, X'894 ') Coda di trasmissione sconosciuta.

RC2197
(2197, X'895 ') Coda di trasmissione predefinita sconosciuta.

RC2198
(2198, X'896 ') La coda di trasmissione predefinita non è locale.

RC2199
(2199, X'897 ') Errore di utilizzo della coda di trasmissione predefinita.

RC2258
(2258, X'8D2') Identificativo gruppo non valido.

RC2248
(2248, X'8C8') Estensione descrittore messaggio non valida.

RC2219
(2219, X'8AB') Chiamata MQI reimmessa prima del completamento della chiamata precedente.

RC2249
(2249, X'8C9') Indicatori messaggio non validi.

RC2250
(2250, X'8CA') Numero di sequenza messaggio non valido.

RC2251
(2251, X'8CB') Scostamento segmento messaggio non valido.

RC2252
(2252, X'8CC') Lunghezza originale non valida.

RC2253
(2253, X'8CD') La lunghezza dei dati nel segmento del messaggio è zero.

RC2255
(2255, X'8CF') Unità di lavoro non disponibile per il gestore code.

RC2257
(2257, X'8D1') Versione non corretta di MQMD fornita.

RC2266
(2266, X'8DA') Uscita carico di lavoro cluster non riuscita.

RC2269
(2269, X'8DD') Errore di risorsa cluster.

RC2270
(2270, X'8DE') Nessuna coda di destinazione disponibile.

RC2420
(2420) È stata emessa una chiamata MQPUT1 , ma i dati del messaggio contengono una struttura MQEPH non valida.

RC2551
(2551, X'9F7') La stringa di selezione specificata non è disponibile.

RC2554
(2554, X'9FA') Non è stato possibile analizzare il contenuto del messaggio per stabilire se il messaggio deve essere consegnato a un sottoscrittore con un selettore di messaggi esteso.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQPUT1(HCONN : OBJDSC : MSGDSC :
C                                      PMO : BUFLEN : BUFFER :
C                                      CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQPUT1           PR                  EXTPROC('MQPUT1')
D* Connection handle
D HCONN                          10I 0 VALUE
D* Object descriptor
D OBJDSC                        468A
D* Message descriptor
D MSGDSC                        364A
D* Options that control the action of MQPUT1
D PMO                           200A
D* Length of the message in BUFFER
D BUFLEN                         10I 0 VALUE
D* Message data
D BUFFER                           *   VALUE
D* Completion code
D CMPCOD                         10I 0
D* Reason code qualifying CMPCOD
D REASON                         10I 0
```

IBM i MQSET (Set object attributes) su IBM i

La chiamata MQSET viene utilizzata per modificare gli attributi di un oggetto rappresentato da un handle. L'oggetto deve essere una coda.

Sintassi

MQSET *(HCONN, HOBJ, SELCNT, SELS, IACNT, INTATR, CALEN, CHRATR, CMPCOD, REASON)*

Note d'utilizzo

1. Utilizzando questa chiamata, l'applicazione può specificare un array di attributi interi, una raccolta di stringhe di attributi di caratteri o entrambi. Se non si verifica alcun errore, gli attributi specificati vengono tutti impostati simultaneamente. Se si verifica un errore (ad esempio, se un selettore non è valido o se si tenta di impostare un attributo su un valore non valido), la chiamata ha esito negativo e non viene impostato alcun attributo.
2. I valori degli attributi possono essere determinati utilizzando la chiamata MQINQ; consultare "MQINQ (Interroga sugli attributi dell'oggetto) su IBM i" a pagina 1342 per i dettagli.

   Nota: Non tutti gli attributi con valori che possono essere interrogati utilizzando la chiamata MQINQ possono avere i relativi valori modificati utilizzando la chiamata MQSET. Ad esempio, non è possibile impostare alcun oggetto del processo o attributo del gestore code con questa chiamata.
3. Le modifiche agli attributi vengono conservate durante i riavvii del gestore code (tranne le modifiche alle code dinamiche temporanee, che non sopravvivono ai riavvii del gestore code).
4. Non è possibile modificare gli attributi di una coda modello utilizzando la chiamata MQSET. Tuttavia, se si apre una coda modello utilizzando la chiamata MQOPEN con l'opzione MQOO_SET, è possibile utilizzare la chiamata MQSET per impostare gli attributi della coda locale dinamica creata dalla chiamata MQOPEN.
5. Se l'oggetto impostato è una coda cluster, è necessario che sia presente un'istanza locale della coda cluster affinché l'apertura abbia esito positivo.

Per ulteriori informazioni sugli attributi dell'oggetto, consultare:

- “Attributi per le definizioni di processo su IBM i .” a pagina 1437
- “Attributi per il gestore code su IBM i” a pagina 1439

Parametri

La chiamata MQSET ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Handle di connessione.

Questo handle rappresenta la connessione al gestore code. Il valore di HCONN è stato restituito da una chiamata MQCONN o MQCONNX precedente.

HOBJ (numero intero con segno a 10 cifre) - immissione

Handle oggetto.

Questo handle rappresenta l'oggetto coda con gli attributi da impostare. L'handle è stato restituito da una precedente chiamata MQOPEN che specificava l'opzione OOSET.

SELCNT (numero intero con segno a 10 cifre) - immissione

Conteggio dei selettori.

Questo è il conteggio dei selettori forniti nell'array SELS . È il numero di attributi che devono essere impostati. Zero è un valore valido. Il numero massimo consentito è 256.

(numero intero con segno a 10 cifre x SELCNT) - immissione

Array di selettori di attributo.

Si tratta di un array di selettori di attributi **SELCNT** ; ogni selettore identifica un attributo (numero intero o carattere) con un valore che deve essere impostato.

Ogni selettore deve essere valido per il tipo di coda rappresentato da HOBJ . Sono consentiti solo alcuni valori IA* e CA*; questi valori sono elencati più avanti in questa sezione.

I selettori possono essere specificati in qualsiasi ordine. I valori di attributo che corrispondono ai selettori di attributi interi (selettori IA*) devono essere specificati in INTATR nello stesso ordine in cui tali selettori si verificano in SELS. I valori di attributo che corrispondono ai selettori di attributi di caratteri (selettori CA*) devono essere specificati in CHRATR nello stesso ordine in cui si verificano tali selettori. I selettori IA* possono essere intercalati con i selettori CA*; è importante solo l'ordine relativo all'interno di ciascun tipo.

Non è un errore specificare lo stesso selettore più di una volta; in questo caso, l'ultimo valore specificato per un particolare selettore è quello che ha effetto.

Nota:

1. I selettori di attributi di caratteri e numeri interi vengono assegnati in due intervalli differenti; i selettori IA* si trovano nell'intervallo tra IAFRST e IALAST e i selettori CA* nell'intervallo tra CAFRST e CALAST.

   Per ogni intervallo, le costanti IALSTU e CALSTU definiscono il valore più alto che il gestore code accetterà.
2. Se tutti i selettori IA* si verificano per primi, è possibile utilizzare gli stessi numeri di elemento per indirizzare gli elementi corrispondenti negli array SELS e INTATR .

Gli attributi che è possibile impostare sono elencati nella seguente tabella. Non è possibile impostare altri attributi utilizzando questa chiamata. Per i selettori di attributi CA*, la costante che definisce la lunghezza in byte della stringa richiesta in CHRATR viene fornita tra parentesi.

Tabella 751. Selettori di attributi MQSET per code		
Selettore	**Descrizione**	**Nota**
CATRGG	Dati trigger (LNTRGD).	“2” a pagina 1389
IADIST	Supporto elenco di distribuzione.	“1” a pagina 1389
IAIGET	Indica se le operazioni get sono consentite.	
IAIPUT	Se le operazioni di inserimento sono consentite.	
IATRGC	Controllo trigger.	“2” a pagina 1389
IATRGG	La lunghezza del trigger.	“2” a pagina 1389
IATRGP	La priorità dei messaggi per i trigger.	“2” a pagina 1389
IATRGT	Il tipo di trigger.	“2” a pagina 1389

Note:

1. Supportato solo sulle piattaforme seguenti:
   - AIX
   - IBM i
   - Windows

   e per i client IBM MQ connessi a questi sistemi.
2. Non supportato su VSE/ESA.

IACNT (numero intero con segno a 10 cifre) - input

Conteggio degli attributi interi.

Questo è il numero di elementi nell'array INTATR e deve essere almeno il numero di selettori IA* nel parametro **SELS** . Zero è un valore valido se non ce ne sono.

INTATR (numero intero con segno a 10 cifre x rxIACNT) - immissione

Array di attributi integer.

Questo è un array di IACNT valori di attributo integer. Questi valori attributo devono essere nello stesso ordine dei selettori IA* nell'array SELS .

CALEN (numero intero con segno a 10 cifre) - immissione

Lunghezza del buffer degli attributi carattere.

Questa è la lunghezza in byte del parametro **CHRATR** e deve essere almeno la somma delle lunghezze degli attributi carattere specificati nell'array SELS . Zero è un valore valido se non sono presenti selettori CA* in SELS.

CHRATR (stringa di caratteri a 1 byte x CALEN) - immissione

Attributi carattere.

Questo è il buffer contenente i valori di attributo carattere, concatenati insieme. La lunghezza del buffer viene fornita dal parametro **CALEN** .

Gli attributi dei caratteri devono essere specificati nello stesso ordine dei selettori CA* nell'array SELS . La lunghezza di ciascun attributo carattere è fissa (consultare SELS). Se il valore da impostare per un attributo contiene un numero di caratteri non vuoti inferiore alla lunghezza definita dell'attributo, il valore in CHRATR deve essere riempito a destra con spazi vuoti per far sì che il valore dell'attributo corrisponda alla lunghezza definita dell'attributo.

CMPCOD (numero intero con segno a 10 cifre) - output

Codice di completamento.

Il valore è uno dei seguenti:

CCOK
Completamento con esito positivo.

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Codice di errore CMPCOD.

Se CMPCOD è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se CMPCOD è CCFAIL:

RC2219
(2219, X'8AB') Chiamata MQI reimmessa prima del completamento della chiamata precedente.

RC2006
(2006, X'7D6') Lunghezza degli attributi dei caratteri non valida.

RC2007
(2007, X'7D7') Stringa di attributi carattere non valida.

RC2009
(2009, X'7D9') Connessione al gestore code persa.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2019
(2019, X'7E3') Handle oggetto non valido.

RC2020
(2020, X'7E4') Valore per l'attributo della coda di inibizione - ricezione o inibizione - inserimento non valido.

RC2021
(2021, X'7E5') Conteggio di attributi interi non valido.

RC2023
(2023, X'7E7') Array di attributi interi non valido.

RC2040
(2040, X'7F8') Coda non aperta per il set.

RC2041
(2041, X'7F9') Definizione oggetto modificata dall'apertura.

RC2101
(2101, X'835 ') Oggetto danneggiato.

RC2052
(2052, X'804 ') La coda è stata eliminata.

RC2058
(2058, X'80A') Nome gestore code non valido o sconosciuto.

RC2059
(2059, X'80B') Gestore code non disponibile per la connessione.

RC2162
(2162, X'872 ') Chiusura del gestore code.

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2065
(2065, X'811 ') Conteggio dei selettori non valido.

RC2067
(2067, X'813 ') Selettore attributo non valido.

RC2066
(2066, X'812 ') Conteggio dei selettori troppo grande.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2075
(2075, X'81B') Valore per l'attributo trigger - control non valido.

RC2076
(2076, X'81C') Valore per l'attributo trigger - depth non valido.

RC2077
(2077, X'81D') Valore per l'attributo trigger - message - priority non valido.

RC2078
(2078, X'81E') Valore per l'attributo di tipo trigger non valido.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQSET(HCONN : HOBJ : SELCNT :
C                                     SELS(1) : IACNT : INTATR(1) :
C                                     CALEN : CHRATR : CMPCOD :
C                                     REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQSET            PR                  EXTPROC('MQSET')
D* Connection handle
D HCONN                           10I 0 VALUE
D* Object handle
D HOBJ                            10I 0 VALUE
D* Count of selectors
D SELCNT                          10I 0 VALUE
D* Array of attribute selectors
D SELS                            10I 0
D* Count of integer attributes
D IACNT                           10I 0 VALUE
D* Array of integer attributes
```

```
     D INTATR                        10I 0
     D* Length of character attributes buffer
     D CALEN                         10I 0 VALUE
     D* Character attributes
     D CHRATR                          *   VALUE
     D* Completion code
     D CMPCOD                        10I 0
     D* Reason code qualifying CMPCOD
     D REASON                        10I 0
```

IBM i MQSETMP (Impostazione della proprietà dell'handle del messaggio) su IBM i

La chiamata MQSETMP imposta o modifica una proprietà di un handle del messaggio.

Sintassi

(Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength, Value, CompCode, Reason) MQSETMP

Note d'utilizzo

- È possibile utilizzare questa chiamata solo quando il gestore code stesso coordina l'unità di lavoro. Questo può essere:
  - Un'unità di lavoro locale, in cui le modifiche influenzano solo le risorse IBM MQ .
  - Un'unità di lavoro globale, in cui le modifiche possono influire sulle risorse appartenenti ad altri gestori risorse, nonché sulle risorse IBM MQ .

  Per ulteriori dettagli sulle unità di lavoro locali e globali, consultare "MQBEGIN (Inizio unità di lavoro) su IBM i" a pagina 1290.
- In ambienti in cui il gestore code non coordina l'unità di lavoro, utilizzare la chiamata di backout appropriata invece di MQBACK. L'ambiente potrebbe anche supportare un backout implicito causato dalla chiusura anomala dell'applicazione.
  - Su z/OS, utilizzare le seguenti chiamate:
    - I programmi batch (inclusi i programmi IMS batch DL/I) possono utilizzare la chiamata MQBACK se l'unità di lavoro influenza solo le risorse IBM MQ . Tuttavia, se l'unità di lavoro influisce sia sulle risorse IBM MQ che sulle risorse appartenenti ad altri gestori risorse (ad esempio, Db2), utilizzare la chiamata SRRBACK fornita da RRS (Recoverable Resource Service) z/OS . La chiamata SRRBACK ripristina le modifiche alle risorse appartenenti ai gestori risorse che sono state abilitate per il coordinamento RRS.
    - Le applicazioni CICS devono utilizzare il comando `EXEC CICS SYNCPOINT ROLLBACK` per eseguire il backout dell'unità di lavoro. Non utilizzare la chiamata MQBACK per applicazioni CICS .
    - Le applicazioni IMS (diverse dai programmi DL/I batch) devono utilizzare chiamate IMS come `ROLB` per eseguire il backout dell'unità di lavoro. Non utilizzare la chiamata MQBACK per applicazioni IMS (diverse dai programmi DL/I batch).
  - Su IBM i, utilizzare questa chiamata per le unità di lavoro locali coordinate dal gestore code. Ciò significa che una definizione di commit non deve esistere a livello di lavoro, ovvero il comando STRCMTCTL con il parametro **CMTSCOPE(*JOB)** non deve essere stato immesso per il lavoro.
- Se un'applicazione termina con modifiche non sottoposte a commit in un'unità di lavoro, la disposizione di tali modifiche dipende dal fatto che l'applicazione termini normalmente o in modo anomalo.

Consultare le note sull'utilizzo in “MQDISC (Disconnetti gestore code) su IBM i” a pagina 1327 per ulteriori dettagli.

- Quando un'applicazione inserisce o richiama i messaggi in gruppi o segmenti di messaggi logici, il gestore code conserva le informazioni relative al gruppo di messaggi e al messaggio logico per le ultime chiamate MQPUT e MQGET riuscite. Queste informazioni sono associate all'handle della coda e includono:
  - I valore dei campi *GroupId*, *MsgSeqNumber*, *Offset*e *MsgFlags* in MQMD.
  - Se il messaggio fa parte di un'unità di lavoro.
  - Per la chiamata MQPUT: se il messaggio è persistente o non persistente.

  Il gestore code conserva tre serie di informazioni di gruppi e segmenti, una per ciascuno dei seguenti:
  - L'ultima chiamata MQPUT riuscita (può far parte di un'unità di lavoro).
  - L'ultima chiamata MQGET riuscita che ha rimosso un messaggio dalla coda (può far parte di un'unità di lavoro).
  - L'ultima chiamata MQGET riuscita che ha visualizzato un messaggio sulla coda (non può far parte di un'unità di lavoro).

  Se l'applicazione inserisce o richiama i messaggi come parte di un'unità di lavoro e l'applicazione decide quindi di eseguire il backout dell'unità di lavoro, le informazioni sul gruppo e sul segmento vengono ripristinate sul valore che aveva in precedenza:
  - Le informazioni associate alla chiamata MQPUT vengono ripristinate al valore che aveva prima della prima chiamata MQPUT riuscita per tale handle di coda nell'unità di lavoro corrente.
  - Le informazioni associate alla chiamata MQGET vengono ripristinate al valore che aveva prima della prima chiamata MQGET riuscita per tale handle di coda nell'unità di lavoro corrente.

  Le code che sono state aggiornate dall'applicazione dopo l'avvio dell'unità di lavoro, ma al di fuori dell'ambito dell'unità di lavoro, non hanno le relative informazioni sul gruppo e sul segmento ripristinate se viene eseguito il backout dell'unità di lavoro.

  Il ripristino delle informazioni sul gruppo e sul segmento al suo valore precedente quando viene eseguito il backout di un'unità di lavoro consente all'applicazione di distribuire un gruppo di messaggi di grandi dimensioni o un messaggio logico di grandi dimensioni costituito da molti segmenti in diverse unità di lavoro e di riavviare nel punto corretto nel gruppo di messaggi o nel messaggio logico in caso di errore di una delle unità di lavoro.

  L'utilizzo di diverse unità di lavoro potrebbe essere vantaggioso se il gestore code locale ha solo una memoria di coda limitata. Tuttavia, l'applicazione deve conservare informazioni sufficienti per essere in grado di riavviare l'inserimento o il richiamo dei messaggi nel punto corretto se si verifica un errore di sistema.

  Per i dettagli su come riavviare il sistema nel punto corretto dopo un malfunzionamento del sistema, consultare l'opzione PMLOGO descritta in PMOPT (10 cifre intere con segno)e l'opzione GMLOGO descritta in GMOPT (10 cifre intere con segno).

  Le note di uso rimanenti si applicano solo quando il gestore code coordina le unità di lavoro:
- Un'unità di lavoro ha lo stesso ambito di un handle di connessione. Tutte le chiamate IBM MQ che interessano una particolare unità di lavoro devono essere eseguite utilizzando lo stesso handle di connessione. Le chiamate emesse utilizzando un handle di collegamento differente (ad esempio, le chiamate emesse da un'altra applicazione) influenzano un'unità di lavoro diversa. Consultare HCONN (10 - digit signed integer) - output per informazioni sull'ambito degli handle di connessione.
- Solo i messaggi inseriti o richiamati come parte dell'unità di lavoro corrente vengono influenzati da questa chiamata.
- Un'applicazione di lunga durata che emette chiamate MQGET, MQPUT o MQPUT1 all'interno di un'unità di lavoro, ma che non emette mai una chiamata di commit o di backout, può riempire le code con messaggi che non sono disponibili per altre applicazioni. Per evitare questa possibilità, l'amministratore deve impostare l'attributo del gestore code **MaxUncommittedMsgs** su un valore sufficientemente

basso per evitare che le applicazioni runaway riempiano le code, ma abbastanza alto per consentire il corretto funzionamento delle applicazioni di messaggistica previste.

Parametri

La chiamata MQSETMP ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Questo handle rappresenta la connessione al gestore code.

Il valore deve corrispondere all'handle di connessione utilizzato per creare l'handle del messaggio specificato nel parametro **HMSG** .

Se l'handle del messaggio è stato creato utilizzando HCUNAS, è necessario stabilire una connessione valida sul thread impostando una proprietà dell'handle del messaggio, altrimenti la chiamata ha esito negativo con il codice di errore RC2009 .

HMSG (numero intero con segno a 20 cifre) - input

Questo è l'handle del messaggio da modificare. Il valore è stato restituito da una precedente chiamata MQCRTMH.

SETOPT (MQSMPO) - input

Controllare come sono impostate le proprietà del messaggio.

Questa struttura permette alle applicazioni di specificare le opzioni che controllano la modalità di impostazione delle proprietà del messaggio. La struttura è un parametro di input nella chiamata MQSETMP. Per ulteriori informazioni, consultare MQSMPO .

PRNAME (MQCHARV) - input

Questo è il nome della proprietà da impostare.

Consultare Nomi proprietà e Limitazioni nome proprietà per ulteriori informazioni sull'utilizzo dei nomi proprietà.

PRPDSC (MQPD) - input/output

Questa struttura viene utilizzata per definire gli attributi di una proprietà, inclusi:

- cosa succede se la proprietà non è supportata
- a quale contesto di messaggio appartiene la propriet ...
- in quali messaggi viene copiata la propriet ... man mano che fluisce

Per ulteriori informazioni su questa struttura, consultare MQPD .

TYPE (numero intero con segno a 10 cifre) - input

Il tipo di dati della proprietà impostata. Può essere uno dei seguenti valori:

TIPOnota di carico
Un valore booleano. *ValueLength* deve essere 4.

TYPBST
Una stringa di byte. *ValueLength* deve essere uguale o maggiore di zero.

TYPI8
Un numero intero con segno a 8 bit. *ValueLength* deve essere 1.

TYPI16
Un numero intero con segno a 16 bit. *ValueLength* deve essere 2.

TYPI32
Un numero intero con segno a 32 bit. *ValueLength* deve essere 4.

TYPI64
Un numero intero con segno a 64 bit. *ValueLength* deve essere 8.

TYPF32
Un numero a virgola mobile a 32 bit. *ValueLength* deve essere 4.

TYPF64
Un numero a virgola mobile a 64 bit. *ValueLength* deve essere 8.

TIPOSTR
Una stringa di caratteri. *ValueLength* deve essere uguale o maggiore di zero o il valore speciale VLNULL.

TYPNUL
La proprietà esiste ma ha un valore null. *ValueLength* deve essere zero.

VALLEN (numero intero con segno a 10 cifre) - input

La lunghezza in byte del valore della proprietà nel parametro *Valore* .

Zero è valido solo per valori null o per stringhe o stringhe di byte. Zero indica che la proprietà esiste ma che il valore non contiene caratteri o byte.

Il valore deve essere maggiore o uguale a zero o al seguente valore speciale se il parametro *Tipo* ha TYPSTR impostato:

VLNULL
Il valore è delimitato dal primo null rilevato nella stringa. Il valore null non è incluso come parte della stringa. Questo valore non è valido se non è impostato anche TYPSTR.

Nota: il carattere null utilizzato per terminare una stringa se VLNULL è impostato è un valore null dalla serie di caratteri del valore.

VALUE (stringa bit 1 byte x VALLEN) - input

Il valore della proprietà da impostare. Il buffer deve essere allineato su un limite appropriato alla natura dei dati nel valore.

Nel linguaggio di programmazione C, il parametro viene dichiarato come un puntatore a void; l'indirizzo di qualsiasi tipo di dati può essere specificato come parametro.

Se *ValueLength* è zero, non si fa riferimento a *Valore* . In questo caso, l'indirizzo del parametro inoltrato dai programmi scritti nell'assembler C o System/390 può essere null.

CMPCOD (numero intero con segno a 10 cifre) - output

Il codice di completamento; è uno dei seguenti:

CCOK
Completamento con esito positivo.

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Il codice motivo che qualifica *CMPCOD*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCWARN:

RC2421
(2421, X'0975 ') Impossibile analizzare una cartella MQRFH2 contenente le proprietà.

Se *CMPCOD* è CCFAIL:

RC2204
(2204, X'089C') Adattatore non disponibile.

RC2130
(2130, X'852 ') Impossibile caricare il modulo di servizio adattatore.

RC2157
(2157, X'86D') Gli ASID principale e home differiscono.

RC2004
(2004, X'07D4') Parametro valore non valido.

RC2005
(2005, X'07D5') Parametro di lunghezza valore non valido.

RC2219
(2219, X'08AB') Chiamata MQI immessa prima del completamento della chiamata precedente.

RC2460
(2460, X'099C') Il puntatore della gestione messaggi non è valido.

RC2499
(2499, X'09C3') handle del messaggio già in uso.

RC2046
(2046, X'07FE') Opzioni non valide o non congruenti.

RC2482
(2482, X'09B2') Struttura descrittore proprietà non valido.

RC2442
(2442, X'098A') Nome proprietà non valido.

RC2473
(2473, X'09A9') Tipo dati proprietà non valido.

RC2472
(2472, X'09A8') Errore di formato numero rilevato nei dati del valore.

RC2463
(2463, X'099F') Struttura delle opzioni della proprietà del messaggio impostata non valida.

RC2111
(2111, X'083F') Identificativo serie di caratteri codificato del nome proprietà non valido.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

Per ulteriori dettagli, vedere “Codici di ritorno per IBM i (ILE RPG)” a pagina 1467.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQSETMP(HCONN : HMSG : SETOPT :
                                        PRNAME : PRPDSC :
                                        TYPE : VALLEN : VALUE :
                                        CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
DMQSETMP          PR                  EXTPROC('MQSETMP')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Message handle
D HMSG                          10I 0 VALUE
D* Options that control the action of MQSETMP
D SETOPT                        20A
D* Property name
D PRNAME                        32A
D* Property descriptor
D PRPDSC                        24A
D* Property data type
D TYPE                          10I 0 VALUE
D* Length of the Value area
D VALLEN                        10I 0 VALUE
```

```
     D* Property value
     D VALUE                          *   VALUE
     D* Completion code
     D CMPCOD                       10I 0
     D* Reason code qualifying CompCode
     D REASON                       10I 0
```

IBM i MQSTAT (Richiamo delle informazioni sullo stato) su IBM i

Utilizzare la chiamata MQSTAT per richiamare le informazioni sullo stato. Il tipo di informazioni di stato restituito è determinato dal valore STYPE specificato nella chiamata.

Sintassi

MQSTAT *(HCONN, STYPE, STAT, CMPCOD, REASON)*

Note d'utilizzo

1. Una chiamata a MQSTAT che specifica un tipo di STATAPT restituisce informazioni sulle precedenti operazioni MQPUT asincrone e MQPUT1 . La struttura MQSTAT inoltrata sulla chiamata viene completata con le prime informazioni di errore o di avvertenza asincrone registrate per tale connessione. Se ulteriori errori o avvertenze seguono il primo, normalmente non modificano questi valori. Tuttavia, se si verifica un errore con un codice di completamento CCWARN, viene restituito un errore successivo con un codice di completamento CCFAIL.
2. Se non si sono verificati errori dal momento in cui è stata stabilita la connessione o dall'ultima chiamata a MQSTAT, vengono restituiti un CMPCOD di CCOK e un MOTIVO di RCNONE.
3. I conteggi del numero di chiamate asincrone che sono state elaborate sotto l'handle di connessione vengono restituiti utilizzando tre contatori: STSPSC, STSPWC e STSPFC. Questi contatori vengono incrementati dal gestore code ogni volta che un'operazione asincrona viene elaborata correttamente, ha un'avvertenza o ha esito negativo (notare che, per motivi di account, un inserimento in un elenco di distribuzione viene contato una volta per coda di destinazione anziché una volta per elenco di distribuzione).
4. Una chiamata eseguita correttamente a MQSTAT determina la reimpostazione di tutte le precedenti informazioni di errore o conteggi.

Parametri

La chiamata MQSTAT ha i seguenti parametri:

Hconn (MQHCONN) - input

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

STYPE (numero intero con segno a 10 cifre) - input

Tipo di informazioni di stato richieste. L'unico valore valido è:

STATAPT
Restituisce le informazioni sulle precedenti operazioni di inserimento asincrone.

STS (MQSTS) - input/output

Struttura delle informazioni sullo stato. Vedi "MQSTS (Status reporting structure) su IBM i" a pagina 1266 per i dettagli.

CMPCOD (numero intero con segno a 10 cifre) - output

Il codice di completamento; è uno dei seguenti:

CCOK
Completamento con esito positivo.

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Il codice di errore che qualifica *CMPCOD*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCFAIL:

RC2374
(2374, X' 946 ') Uscita API non riuscita

RC2183
(2183, X'887 ') Impossibile caricare l'uscita API.

RC2219
(2219, X'8AB') Chiamata MQI immessa prima del termine della precedente chiamata.

RC2009
(2009, X'7D9') Connessione al gestore code persa.

RC2203
(2203, X'89B') Chiusura della connessione.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2162
(2162, X'872 ') Arresto del gestore code

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2430
(2430, X'97E') Errore con tipo MQSTAT.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2424
(2424, X' 978 ') Errore con la struttura MQSTS

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

RC2298
(2298, X'8FA') La funzione richiesta non è disponibile nell'ambiente corrente.

Per informazioni dettagliate su questi codici, consultare:

- Messaggi e codici di errore

Dichiarazione RPG

```
C*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
C                     CALLP     MQSTAT(HCONN : ETYPE : ERR :
C                                      CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
D.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
DMQSTAT           PR                  EXTPROC('MQSTAT')
D* Connection handle
D HCONN                          10I 0 VALUE
D* Status information type
D STYPE                          10I 0 VALUE
D* Status information
D STATUS                        296A
D* Completion code
D CMPCOD                         10I 0
D* Reason code qualifying CompCode
D REASON                         10I 0
```

IBM i MQSUB (Register Subscription) su IBM i

La chiamata MQSUB registra la sottoscrizione delle applicazioni a un particolare argomento.

- "Sintassi" a pagina 1399
- "Note d'utilizzo" a pagina 1399
- "Parametri" a pagina 1401
- "Dichiarazione RPG" a pagina 1404

Sintassi

MQSUB *(HCONN, SUBDSC, HOBJ, HSUB, CMPCOD, REASON)*

Note d'utilizzo

- La sottoscrizione viene effettuata a un argomento, denominato utilizzando il nome breve di un oggetto argomento predefinito, il nome completo della stringa argomento o è formato dalla concatenazione di due parti, come descritto in Combinazione di stringhe argomento.
- Il gestore code esegue i controlli di sicurezza quando viene emessa una chiamata MQSUB, per verificare che l'identificativo utente con cui l'applicazione è in esecuzione disponga del livello di autorizzazione appropriato prima che sia consentito l'accesso. L'oggetto dell'argomento appropriato viene individuato da un nome breve fornito nella chiamata o dall'oggetto del nome breve più vicino nella gerarchia dell'argomento che viene trovato se viene fornito un nome lungo. Viene effettuato un controllo di autorizzazione su questo oggetto argomento per garantire che l'autorità per la sottoscrizione sia impostata e sulla coda di destinazione per garantire che l'autorizzazione per l'emissione sia impostata. Se viene utilizzata l'opzione SDMAN, ciò significa che viene eseguito un controllo dell'autorizzazione sul nome della coda gestita associato a questo oggetto argomento e se viene fornita una coda non gestita, ciò significa che viene eseguito un controllo dell'autorizzazione sulla coda rappresentata dal parametro **HOBJ** .
- *HOBJ* restituito sulla chiamata MQSUB quando viene utilizzata l'opzione SOMAN, può essere interrogato per individuare attributi quali la soglia di backout e il nome della riaccodamento di backout eccessivo. È anche possibile richiedere il nome della coda gestita, ma non si dovrebbe tentare di aprirla direttamente.
- Le sottoscrizioni possono essere raggruppate consentendo la distribuzione di una sola pubblicazione al gruppo di sottoscrizioni anche se più di un gruppo corrispondeva alla pubblicazione. Le sottoscrizioni sono raggruppate utilizzando l'opzione SOGRP e per raggruppare le sottoscrizioni devono:
  - utilizzare la stessa coda denominata (che non utilizza l'opzione SOMAN) sullo stesso gestore code, rappresentata dal parametro **HOBJ** nella chiamata MQSUB
  - condividere lo stesso *SDCID*
  - essere dello stesso *SDSL*

  Questi attributi definiscono la serie di sottoscrizioni considerate nel gruppo e sono anche gli attributi che non possono essere modificati se una sottoscrizione è raggruppata. La modifica di *SDSL* risulta

in RC2512e la modifica di uno degli altri (che può essere modificato se una sottoscrizione non è raggruppata) risulta in RC2515.

- I campi in MQSD vengono completati al ritorno da una chiamata MQSUB che utilizza l'opzione SORES. L'MQSD restituito può essere passato direttamente in una chiamata MQSUB che utilizza l'opzione SOALT con tutte le modifiche necessarie alla sottoscrizione applicata all'MQSD. Alcuni campi hanno considerazioni speciali come indicato nella tabella.

Tabella 752. Output MQSD da MQSUB

Nome campo in MQSD	**Considerazioni speciali**
Opzioni di accesso o di creazione	Nessuna di queste opzioni è impostata al ritorno dalla chiamata MQSUB. Se successivamente si riutilizza l'MQSD in una chiamata MQSUB, l'opzione richiesta deve essere impostata esplicitamente.
Opzioni di durata, Opzioni di destinazione, Opzioni di registrazione & Opzioni Wildcard	Queste opzioni verranno impostate in base alle esigenze
Opzioni di pubblicazione	Queste opzioni verranno impostate come appropriato, ad eccezione di SONEWP che è applicabile solo a SOCRE.
Altre opzioni	Queste opzioni non vengono modificate al ritorno da una chiamata MQSUB. Controllano come viene emessa la chiamata API e non vengono memorizzati con la sottoscrizione. Devono essere impostati come richiesto in qualsiasi successiva chiamata MQSUB che riutilizzi MQSD.
ObjectName	Questo campo di sola immissione non viene modificato al ritorno da una chiamata MQSUB.
ObjectString	Questo campo di sola immissione non viene modificato al ritorno da una chiamata MQSUB. Il nome completo dell'argomento utilizzato viene restituito nel campo *SDRO* , se viene fornito un buffer.
ID AlternateUsere ID AlternateSecurity	Questi campi di solo input non vengono modificati al ritorno da una chiamata MQSUB. Controllano come viene emessa la chiamata API e non vengono memorizzati con la sottoscrizione. Devono essere impostati come richiesto su qualsiasi chiamata MQSUB successiva che riutilizzi MQSD.
SubExpiry	Al ritorno da una chiamata MQSUB che utilizza l'opzione SORES, questo campo verrà impostato sulla scadenza originale della sottoscrizione e non sul tempo di scadenza rimanente. Se si riutilizza MQSD in una chiamata MQSUB utilizzando l'opzione SOALT, si reimposterà la scadenza della sottoscrizione per iniziare di nuovo il conto alla rovescia.
SubName	Questo campo è un campo di input su una chiamata MQSUB e non viene modificato sull'output.
SubUserData e SelectionString	Questi campi a lunghezza variabile verranno restituiti all'output di una chiamata MQSUB utilizzando l'opzione SORES, se viene fornito un buffer, e anche una lunghezza del buffer positiva in *VCHRP*. Se non viene fornito alcun buffer, verrà restituita solo la lunghezza nel campo *VCHRL* di MQCHARV.If il buffer fornito è inferiore allo spazio richiesto per restituire il campo, nel buffer fornito vengono restituiti solo *VCHRP* byte. Se successivamente si riutilizza MQSD in una chiamata MQSUB utilizzando l'opzione SOALT e non viene fornito un buffer, ma viene fornito un *VCHRL* diverso da zero, se tale lunghezza corrisponde alla lunghezza esistente del campo, non verrà apportata alcuna modifica al campo.
Token SubCorrelId e PubAccounting	Se non si utilizza SOSCID, il *SDCID* verrà generato dal gestore code. Se non si utilizza SOSETI, il *SDACC* verrà generato dal gestore code. Questi campi verranno restituiti in MQSD da una chiamata MQSUB utilizzando l'opzione SORES. Se sono generati dal gestore code, il valore generato verrà restituito su una chiamata MQSUB utilizzando l'opzione SOCRE o SOALT.
PubPriority, SubLevel & PubApplIdentityData	Questi campi verranno restituiti in MQSD.

Tabella 752. Output MQSD da MQSUB (Continua)	
Nome campo in MQSD	**Considerazioni speciali**
Stringa ResObject	Questo campo di output solo verrà restituito in MQSD se viene fornito un buffer.

Parametri

La chiamata MQSUB ha i seguenti parametri:

HCONN (numero intero con segno a 10 cifre) - input

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

SUBDSC (MQSD) - input/output

Si tratta di una struttura che identifica l'oggetto con l'utilizzo registrato dall'applicazione. Per ulteriori informazioni, fare riferimento a "MQSD (Subscription descriptor) su IBM i" a pagina 1248.

HOBJ (numero intero con segno a 10 cifre) - input/output

Questo handle rappresenta l'accesso stabilito per ottenere i messaggi inviati a questa sottoscrizione. Questi messaggi possono essere memorizzati su una coda specifica o al gestore code può essere richiesto di gestire la memoria senza la necessità di una coda specifica.

Handle oggetto.

Se deve essere utilizzata una coda specifica, questa deve essere associata alla sottoscrizione al momento della creazione. Ciò può essere fatto in due modi:

- Fornendo questa gestione quando si richiama MQSUB con l'opzione SDCRT. Se questo handle viene fornito come parametro di input sulla chiamata, deve essere un handle di oggetto valido restituito da una precedente chiamata MQOPEN di una coda utilizzando almeno uno tra OOINP*, OOOUT (se ad esempio una coda remota) o l'opzione OOBRW. In caso contrario, la chiamata ha esito negativo con RC2019. Non può essere un handle di oggetto per una coda alias che si risolve in un oggetto argomento. In tal caso, la chiamata non riesce con RC2019
- Utilizzando il comando DEFINE SUB MQSC e fornito tale comando con il nome di un oggetto coda.

Se il gestore code deve gestire la memoria dei messaggi inviati a questa sottoscrizione, è necessario indicare quando viene creata la sottoscrizione, utilizzando l'opzione SOMAN e impostando il valore del parametro su HONONE. Il gestore code restituisce l'handle come parametro di output sulla chiamata e l'handle restituito è noto come handle gestito. Se viene specificato HONONE e non viene specificato anche SOMAN, la chiamata ha esito negativo con RC2019.

Un handle gestito restituito dal gestore code può essere utilizzato su una chiamata MQGET o MQCB, con o senza opzioni di esplorazione, su una chiamata MQINQ o su MQCLOSE. Non può essere utilizzato su MQPUT, MQSET o su un MQSUB successivo; il tentativo non riesce con RC2039 per MQPUT, RC2040 per MQSET o RC2038 per MQSUB.

Se l'opzione SORES nel campo *OPTS* nella struttura MQSD viene utilizzata per riprendere questa sottoscrizione, l'handle può essere restituito all'applicazione in questo parametro se viene specificato HONONE. È possibile utilizzare questa opzione indipendentemente dal fatto che la sottoscrizione utilizzi o meno un handle gestito. Può essere utile per le sottoscrizioni create utilizzando DEFINE SUB se si desidera la gestione della coda di sottoscrizione definita nel comando DEFINE SUB. Nel caso in cui venga ripresa una sottoscrizione creata amministrativamente, la coda viene aperta con OOINPQ e OOBRW. Se sono necessarie altre opzioni, l'applicazione deve aprire esplicitamente la coda di sottoscrizione e fornire l'handle dell'oggetto sulla chiamata. Se si verifica un problema durante l'apertura della coda, la chiamata avrà esito negativo con RC2522. Se viene fornito *HOBJ* , deve essere equivalente a *HOBJ* nella chiamata MQSUB originale. Ciò significa che se viene fornito un handle dell'oggetto restituito da una chiamata MQOPEN, l'handle deve essere nella stessa coda utilizzata in precedenza oppure la chiamata ha esito negativo con RC2019.

Se questa sottoscrizione viene modificata, utilizzando l'opzione SOALT nel campo *OPTS* nella struttura MQSD, è possibile fornire un *HOBJ* differente. Tutte le pubblicazioni che sono state consegnate alla coda precedentemente identificata tramite questo parametro rimangono su quella coda ed è responsabilità dell'applicazione richiamare tali messaggi se il parametro **HOBJ** ora rappresenta una coda diversa.

L'uso di questo parametro con varie opzioni di sottoscrizione è riepilogato nella seguente tabella:

Tabella 753. Utilizzo di Hobj con varie opzioni di sottoscrizione

Opzioni	**HOBJ**	**Descrizione**
SOCRT + SOMAN	Ignorato all'immissione	Crea una sottoscrizione con la memoria gestita del gestore code dei messaggi.
SOCRT	Handle oggetto valido	Crea una sottoscrizione fornendo una coda specifica come destinazione per i messaggi.
SORES	ONONE	Ripristina una sottoscrizione precedentemente creata (gestita o meno) e fa in modo che il gestore code restituisca l'handle dell'oggetto per l'utilizzo da parte dell'applicazione.
SORES	Valido, corrispondente, handle oggetto	Riprende una sottoscrizione creata precedentemente che utilizza una coda specifica come destinazione per i messaggi e utilizza un handle di oggetto con opzioni di apertura specifiche.
SOALT + SOMAN	ONONE	Modifica una sottoscrizione esistente che in precedenza utilizzava una specifica coda, per essere ora gestita.
SOALT	Handle oggetto valido	Modifica una sottoscrizione esistente per utilizzare una coda specifica (gestita o da una coda specifica differente).

Se è stato fornito o restituito, *HOBJ* deve essere specificato nelle successive chiamate MQGET necessarie per ricevere le pubblicazioni.

L'handle *HOBJ* cessa di essere valido quando viene emessa la chiamata MQCLOSE o quando termina l'unità di elaborazione che definisce l'ambito dell'handle. L'ambito dell'handle dell'oggetto restituito è uguale a quello dell'handle di collegamento specificato nella chiamata. Consultare HCONN per informazioni sull'ambito dell'handle. Un MQCLOSE dell'handle *HOBJ* non ha alcun effetto sull'handle *HSUB* .

HSUB (numero intero con segno a 10 cifre) - output

Questo handle rappresenta la sottoscrizione effettuata. Può essere utilizzato per altre due operazioni:

- Può essere utilizzato su una chiamata MQSUBRQ successiva per richiedere l'invio di pubblicazioni quando l'opzione SOPUBR è stata utilizzata durante l'esecuzione della sottoscrizione.
- Può essere utilizzato su una chiamata MQCLOSE successiva per rimuovere la sottoscrizione effettuata. L'handle *HSUB* cessa di essere valido quando viene emessa la chiamata MQCLOSE o quando l'unità di elaborazione che definisce l'ambito dell'handle termina. L'ambito dell'handle dell'oggetto restituito è uguale a quello dell'handle di collegamento specificato nella chiamata. Un MQCLOSE dell'handle *HSUB* non ha alcun effetto sull'handle *HOBJ* .

Questo handle non può essere passato a una chiamata MQGET o MQCB. È necessario utilizzare il parametro **HOBJ** . Il passaggio di questo handle a qualsiasi altro risultato della chiamata IBM MQ in RC2019.

CMPCOD (numero intero con segno a 10 cifre) - output

Il codice di completamento; è uno dei seguenti:

CCOK
Completamento riuscito

AVVCCN
Avvertenza (completamento parziale)

CCNON RIUSCITO
Chiamata non riuscita

REASON (numero intero con segno a 10 cifre) - output

Il codice di errore che qualifica *CMPCOD*.

Se *CMPCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CMPCOD* è CCFAIL:

RC2019
(2019 X'07E3') Handle oggetto non valido

RC2046
(2046 X'07FE') Opzioni non valide o non congruenti

RC2085
(2085 X'0825 ') Impossibile trovare l'oggetto identificato

RC2161
(2161 X'0871 ') Gestore code in fase di sospensione

RC2298
(2298 X'08FA') Funzione non supportata.

RC2424
(2424 X'0978 ') Descrittore sottoscrizione (MQSD) non valido

RC2425
(2441 X' 979 ') Stringa argomento non valida

RC2428
(2428 X'097C') il nome sottoscrizione specificato non corrisponde alle sottoscrizioni esistenti

RC2429
(2429 X'097D') Il nome della sottoscrizione esiste ed è utilizzato da un'altra applicazione

RC2431
(2431 X'097F') SubUserCampo dati non valido

RC2432
(2432 X'0980 ') La sottoscrizione esiste

RC2434
(2434 X'0982 ') Il nome della sottoscrizione corrisponde alla sottoscrizione esistente

RC2440
(2440 X'0988 ') Campo SubName non valido

RC2441
(2441 X'0989 ') Campo Objectstring non valido

RC2435
(2435 X'0983 ') Impossibile modificare l'attributo utilizzando SDALT oppure la sottoscrizione è stata creata con SDIMM.

RC2436
(2436 X'0984 ') Opzione SODUR non valida

RC2459
(2459, X'99B') Errore di sintassi della stringa di selezione.

RC2503
(2503 X'09C7') Le chiamate MQSUB sono attualmente inibite per gli argomenti sottoscritti.

RC2519
(2519, X'9D7') La stringa di selezione non è specificata nella descrizione di come utilizzare una struttura MQCHARV.

RC2551
(2551, X'9F7') La stringa di selezione specificata non è disponibile.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQSUB(HCONN : SUBDSC : HOBJ :
C                                     HSUB : CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQSUB            PR                  EXTPROC('MQSUB')
D* Connection handle
D HCONN                          10I 0 VALUE
D* Subscription descriptor
D SUBDSC                        400A
D* Object handle for queue
D HOBJ                           10I 0
D* Subscription object handle
D HSUB                           10I 0
D* Completion code
D CMPCOD                         10I 0
D* Reason code qualifying CompCode
D REASON                         10I 0
```

IBM i MQSUBRQ (Richiesta di sottoscrizione) su IBM i

La chiamata MQSUBRQ effettua una richiesta su una sottoscrizione.

Sintassi

MQSUBRQ *(HCONN, HSUB, ACTION, SUBROPT, CMPCOD, REASON)*

Note d'utilizzo

Le seguenti note sull'utilizzo si applicano all'utilizzo di SRAPUB:

1. Se questo verbo viene completato correttamente, le pubblicazioni conservate corrispondenti alla sottoscrizione specificata sono state inviate alla sottoscrizione e possono essere ricevute utilizzando MQGET o MQCB utilizzando l'HOBJ restituito sul verbo MQSUB originale che ha creato la sottoscrizione.
2. Se l'argomento sottoscritto dal verbo MQSUB originale che ha creato la sottoscrizione conteneva un carattere jolly, è possibile che venga inviata più di una pubblicazione conservata. Il numero di pubblicazioni inviate come risultato di questa chiamata viene registrato nel campo *SRNMP* nella struttura SBROPT.
3. Se questo verbo viene completato con un codice motivo RC2437 , non ci sono pubblicazioni attualmente conservate per l'argomento specificato.
4. Se questo verbo viene completato con un codice di errore RC2525 o RC2526 , ci sono attualmente pubblicazioni conservate per l'argomento specificato, ma si è verificato un errore che indica che non è stato possibile consegnarle.

5. L'applicazione deve disporre di una sottoscrizione corrente all'argomento prima di poter effettuare questa chiamata. Se la sottoscrizione è stata effettuata in un'istanza precedente dell'applicazione e non è disponibile un handle valido per la sottoscrizione, l'applicazione deve prima richiamare MQSUB con l'opzione SORES per ottenere un handle da utilizzare in questa chiamata.
6. Le pubblicazioni vengono inviate alla destinazione registrata per l'utilizzo con la sottoscrizione corrente di questa applicazione. Se le pubblicazioni devono essere inviate altrove, la sottoscrizione deve essere prima modificata utilizzando la chiamata MQSUB con l'opzione SOALT.

Parametri

La chiamata MQSUBRQ presenta i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Questo handle rappresenta la connessione al gestore code. Il valore di *HCONN* è stato restituito da una chiamata MQCONN o MQCONNX precedente.

In applicazioni z/OS per CICS , la chiamata MQCONN può essere omessa e il seguente valore specificato per *HCONN*:

HCDEFH
Handle di connessione predefinito.

HSUB (numero intero con segno a 10 cifre) - input

Questo handle rappresenta la sottoscrizione per cui è necessario richiedere un aggiornamento. Il valore di *HSUB* è stato restituito da una chiamata MQSUB precedente.

ACTION (numero intero con segno a 10 cifre) - input

Questo parametro controlla la particolare azione richiesta sulla sottoscrizione. È necessario specificare uno (e solo uno) dei seguenti:

SRAPUB

Questa azione richiede l'invio di una pubblicazione di aggiornamento per l'argomento specificato. Viene normalmente utilizzato se il sottoscrittore ha specificato l'opzione SOPUBR nella chiamata MQSUB quando ha effettuato la sottoscrizione. Se il gestore code dispone di una pubblicazione conservata per l'argomento, questa viene inviata al sottoscrittore. In caso contrario, la chiamata non riesce. Se a un'applicazione viene inviata una pubblicazione che è stata conservata, ciò viene indicato dalla proprietà del messaggio MQIsRetained di tale pubblicazione.

Dal momento che l'argomento nella sottoscrizione esistente rappresentato dal parametro **HSUB** può contenere caratteri jolly, il sottoscrittore potrebbe ricevere più pubblicazioni conservate.

SBROPT (MQSRO) - input/output

Queste opzioni controllano l'azione di MQSUBRQ, consultare “MQSRO - Opzioni di richieste di sottoscrizione” a pagina 605 per dettagli.

CMPCOD (numero intero con segno a 10 cifre) - output

Il codice di completamento; è uno dei seguenti:

CCOK
Completamento riuscito

AVVCCN
Avvertenza (completamento parziale)

CCNON RIUSCITO
Chiamata non riuscita

Motivo (numero intero con segno a 10 cifre) - output

Il codice di errore che qualifica *CMPCOD*.

Se *CPMCOD* è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CPMCOD* è CCFAIL:

RC2298
2298 (X'08FA') La funzione richiesta non è disponibile nell'ambiente corrente.

RC2437
2437 (X'0985 ') Non ci sono pubblicazioni conservate attualmente memorizzate per questo argomento.

RC2046
2046 (X'07FE') Il campo o il parametro Opzioni contiene opzioni non valide o una combinazione di opzioni non valide.

RC2161
2161 (X'0871 ') Gestore code in fase di sospensione

RC2438
2438 (X'0986 ') Nella chiamata MQSUBRQ, le opzioni MQSRO della richiesta di sottoscrizione non è valido.

Dichiarazione RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                    CALLP     MQSUBRQ(HCONN : HSUB : ACTION :
C                                       SBROPT : CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQSUBRQ          PR                  EXTPROC('MQSUBRQ')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Subscription handle
D HSUB                          10I 0 VALUE
D* Action requested on the subscription
D ACTION                        10I 0 VALUE
D* Subscription Request Options
D SBROPT                        16A
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CompCode
D REASON                        10I 0
```

IBM i Attributi degli oggetti su IBM i

Questa raccolta di argomenti elenca solo gli oggetti IBM MQ che possono essere oggetto di una chiamata di funzione MQINQ e fornisce dettagli sugli attributi che è possibile interrogare e sui selettori da utilizzare.

Attributi per le code

Utilizzare queste informazioni per informazioni sui diversi tipi di definizioni di coda e gli attributi supportati da ognuno.

Tipi di coda: il gestore code supporta i seguenti tipi di definizione di coda:

Coda locale
Questa è una coda fisica che memorizza i messaggi. La coda esiste sul gestore code locale.

Le applicazioni connesse al gestore code locale possono inserire e rimuovere messaggi da code di questo tipo. Il valore dell'attributo della coda **QType** è QTLOC.

Coda condivisa
Questa è una coda fisica che memorizza i messaggi. La coda esiste in un repository condiviso accessibile a tutti i gestori code appartenenti al gruppo di condivisione code proprietario del repository condiviso.

Le applicazioni connesse a qualsiasi gestore code nel gruppo di condivisione code possono inserire e rimuovere messaggi da code di questo tipo. Tali code sono effettivamente le stesse delle code locali. Il valore dell'attributo della coda **QType** è QTLOC.

- Le code condivise sono supportate solo su z/OS.

Coda cluster
Questa è una coda fisica che memorizza i messaggi. La coda esiste sul gestore code locale o su uno o più gestori code che appartengono allo stesso cluster del gestore code locale.

Le applicazioni connesse al gestore code locale possono inserire i messaggi nelle code di questo tipo, indipendentemente dall'ubicazione della coda. Se un'istanza della coda esiste sul gestore code locale, la coda si comporta nello stesso modo di una coda locale e le applicazioni connesse al gestore code locale possono rimuovere i messaggi dalla coda. Il valore dell'attributo della coda **QType** è QTCLUS.

Coda alias
Questa non è una coda fisica - è un nome alternativo per una coda locale. Il nome della coda locale in cui viene risolto l'alias fa parte della definizione della coda alias.

Le applicazioni connesse al gestore code locale possono inserire e rimuovere messaggi dalle code alias - i messaggi vengono inseriti e rimossi dalla coda locale in cui l'alias si risolve. Il valore dell'attributo della coda **QType** è QTALS.

Coda remota
Questa non è una coda fisica - è la definizione locale di una coda che esiste su un gestore code remoto. La definizione locale della coda remota contiene informazioni che indicano al gestore code locale come instradare i messaggi al gestore code remoto.

Le applicazioni connesse al gestore code locale possono posizionare i messaggi sulle code remote - i messaggi vengono posizionati sulla coda di trasmissione locale utilizzata per instradare i messaggi al gestore code remoto. Le applicazioni non possono rimuovere i messaggi dalle code remote. Il valore dell'attributo della coda **QType** è QTREM.

Una definizione di coda remota può essere utilizzata anche per:

- Aliasing coda di risposta

  In questo caso, il nome della definizione è il nome di una coda di risposta. Per ulteriori informazioni, consultare Definizioni degli alias della coda di risposta.

- Alias del gestore code

  In tal caso, il nome della definizione è un alias per un gestore code e non il nome di una coda. Per ulteriori informazioni, fare riferimento alla sezione Definizioni alias del gestore code.

Coda modello
Questa non è una coda fisica, ma una serie di attributi della coda da cui è possibile creare una coda locale.

I messaggi non possono essere memorizzati su code di questo tipo.

Alcuni attributi della coda si applicano a tutti i tipi di coda; altri attributi della coda si applicano solo a determinati tipi di coda. I tipi di coda a cui si applica un attributo sono indicati da una "X" in Tabella 754 a pagina 1408 e tabelle successive.

Tabella 754 a pagina 1408 riepiloga gli attributi specifici delle code. Gli attributi sono descritti in ordine alfabetico.

I nomi degli attributi mostrati nella tabella sono i nomi utilizzati con le chiamate MQINQ e MQSET. Quando i comandi MQSC vengono utilizzati per definire, modificare o visualizzare gli attributi, vengono utilizzati nomi brevi alternativi; per i dettagli, consultare Comandi MQSC .

Nella seguente tabella, le colonne si applicano come segue:

- La colonna per le code locali si applica anche alle code condivise.
- La colonna delle code modello indica quali attributi vengono ereditati dalla coda locale creata dalla coda modello.
- La colonna per le code cluster indica gli attributi che possono essere interrogati quando la coda cluster è aperta per l'interrogazione da sola o per l'interrogazione e l'output. Se la coda del cluster è aperta per l'interrogazione più uno o più di input, ricerca o impostazione, viene invece applicata la colonna per le code locali.

Tabella 754. Attributi per le code

Attributo	Descrizione	Locale	Modello	Alias	Remoto	Cluster
AlterationDate	Data dell'ultima modifica della definizione	X		X	X	
AlterationTime	Ora dell'ultima modifica della definizione	X		X	X	
QNameBackoutRequeue	Nome coda di riaccodamento di backout eccessivo	X	X			
BackoutThreshold	Soglia di ripristino	X	X			
BaseQName	Nome coda in cui si risolve l'alias			X		
NomeClusterChannel	Nome canale mittente del cluster	✓	✓			
ClusterName	Nome del cluster a cui appartiene la coda	X		X	X	
ClusterNamelist	Nome dell'oggetto elenco nomi contenente i nomi dei cluster a cui appartiene la coda	X		X	X	
CreationDate	Data di creazione della coda	X				
CreationTime	Ora di creazione della coda	X				
CurrentQDepth	Profondità corrente coda	X				
DefBind	Binding predefinito	X		X	X	X
DefinitionType	Il tipo di definizione della coda	X	X			
DefInputOpenOption	Opzione predefinita di apertura in immissione	X	X			
DefPersistence	Persistenza predefinita messaggio	X	X	X	X	X

Tabella 754. Attributi per le code (Continua)

Attributo	Descrizione	Locale	Modello	Alias	Remoto	Cluster
DefPriority	Priorità predefinita messaggio	X	X	X	X	X
DistLists	Supporto elenco di distribuzione	X	X			
HardenGetBackout	Indica se mantenere un conteggio di backout accurato	X	X			
InhibitGet	Controlla se le operazioni di acquisizione per la coda sono consentite	X	X	X		
InhibitPut	Controlla se le operazioni di inserimento per la coda sono consentite	X	X	X	X	X
InitiationQName	Nome della coda di iniziazione	X	X			
MaxMsgLength	La lunghezza massima del messaggio in byte	X	X			
MaxQDepth	Massima profondità coda	X	X			
MediaLog	Identità dell'estensione del log meno recente (o del ricevitore di giornale meno recente su IBM i) necessari per il recupero dei supporti di una coda specificata	✓	✓			
MsgDeliverySequence	Sequenza di consegna messaggi	X	X			
OpenInputCount	Numero di aperture per input	X				
OpenOutputCount	Numero di aperture per l'emissione	X				
ProcessName	Nome processo	X	X			
QDepthHighEvent	Controlla se vengono generati eventi Grandezza coda elevata	X	X			
QDepthHighLimit	Limite massimo per la profondità della coda	X	X			

Tabella 754. Attributi per le code (Continua)

Attributo	Descrizione	Locale	Modello	Alias	Remoto	Cluster
QDepthLowEvent	Controlla se vengono generati eventi Profondità coda bassa	X	X			
QDepthLowLimit	Limite inferiore per la profondità della coda	X	X			
QDepthMaxEvent	Controlla se vengono generati eventi Coda piena	X	X			
QDesc	Descrizione coda	X	X	X	X	X
QName	Nome coda	X		X	X	X
QServiceInterval	Destinazione per intervallo di servizio coda	X	X			
QServiceIntervalEvento	Controlla se vengono generati eventi Intervallo di servizio elevato o Intervallo di servizio OK	X	X			
QTYPE	Tipo coda	X		X	X	X
RemoteQmgrName	Nome del gestore code remoto				X	
RemoteQName	Nome della coda remota				X	
RetentionInterval	Intervallo di conservazione	X	X			
Ambito	Controlla se una voce per la coda esiste anche in una directory della cella	X		X	X	
Condivisione	Condividibilità coda	X	X			
TriggerControl	Controllo trigger	X	X			
TriggerData	Dati trigger	X	X			
TriggerDepth	Capacità di Trigger	X	X			
TriggerMsgPriority	Priorità messaggi di soglia per i trigger	X	X			
TriggerType	Tipo trigger	X	X			
Utilizzo	Utilizzo coda	X	X			
XmitQName	Nome coda di trasmissione				X	

IBM i *AlterationDate (stringa di caratteri a 12 byte) su IBM i*

Data dell'ultima modifica della definizione.

Tabella 755. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X		X	X	

Questa è la data dell'ultima modifica della definizione. Il formato della data è YYYY-MM-DD, riempito con due spazi vuoti finali per rendere la lunghezza di 12 byte (ad esempio, 1992-09-23¬¬, dove ¬¬ rappresenta due caratteri vuoti).

I valori di alcuni attributi (ad esempio, *CurrentQDepth*) cambiano man mano che il gestore code opera. Le modifiche a questi attributi non hanno effetto su *AlterationDate*.

Per determinare il valore di questo attributo, utilizzare il selettore CAALTD con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNDATE.

IBM i *AlterationTime (stringa di caratteri a 8 byte) su IBM i*

L'ora dell'ultima modifica della definizione.

Tabella 756. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X		X	X	

Questa è l'ora dell'ultima modifica della definizione. Il formato dell'ora è HH.MM.SS utilizzando l'orologio di 24 ore, con uno zero iniziale se l'ora è inferiore a 10 (ad esempio 09.10.20). Viene utilizzata l'ora locale.

I valori di alcuni attributi (ad esempio, *CurrentQDepth*) cambiano man mano che il gestore code opera. Le modifiche a questi attributi non hanno effetto su *AlterationTime*.

Per determinare il valore di questo attributo, utilizzare il selettore CAALTT con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNTIME.

IBM i *BackoutRequeueQName (stringa di caratteri a 48 byte) su IBM i*

Nome coda di riaccodamento di backout eccessivo.

Tabella 757. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Le applicazioni in esecuzione all'interno di WebSphere Application Server e quelle che utilizzano IBM MQ Application Server Facilities utilizzano questo attributo per determinare dove devono andare i messaggi di cui è stato eseguito il backout. Per tutte le altre applicazioni, oltre a consentire la query del valore, il gestore code non esegue alcuna azione in base al valore dell'attributo.

Per determinare il valore di questo attributo, utilizzare il selettore CABRQN con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQN.

IBM i *BackoutThreshold (numero intero con segno a 10 cifre) attivo IBM i*

La soglia di ripristino.

Tabella 758. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Le applicazioni in esecuzione all'interno di WebSphere Application Server e quelle che utilizzano IBM MQ Application Server Facilities utilizzano questo attributo per determinare se è necessario eseguire il backout di un messaggio. Per tutte le altre applicazioni, oltre a consentire la query del valore, il gestore code non esegue alcuna azione in base al valore dell'attributo.

Per determinare il valore di questo attributo, utilizzare il selettore IABTHR con la chiamata MQINQ.

IBM i *BaseQName (stringa di caratteri a 48 byte) su IBM i*

Il nome della coda in cui viene risolto l'alias.

Tabella 759. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
		X		

Questo è il nome di una coda definita sul gestore code locale. Per ulteriori informazioni sui nomi delle code, consultare la descrizione del campo *ODON* in MQOD. La coda è uno dei seguenti tipi:

TLOC
Coda locale.

TREM
Definizione locale di una coda remota.

QTCLUS
Coda cluster.

Per determinare il valore di questo attributo, utilizzare il selettore CABASQ con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQN.

IBM i *BaseType (struttura parametro intero) su IBM i*

Il tipo di oggetto in cui l'alias viene risolto.

Tabella 760. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
		X		

Questo attributo può avere uno dei seguenti valori:

OTQ
Il tipo di oggetto di base è una coda

OTTOP
Il tipo di oggetto di base è un argomento

IBM i *CFStrucName (stringa di caratteri a 12 byte) su IBM i*

Il nome della struttura con funzione di accoppiamento.

Tabella 761. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Questo è il nome della struttura CFS (coupling facility structure) in cui sono memorizzati i messaggi sulla coda. Il primo carattere del nome è compreso tra A e Z, mentre i restanti caratteri sono compresi tra A e Z, tra 0 e 9 o sono vuoti.

Il nome completo della struttura nella CF (Coupling Facility) si ottiene aggiungendo il suffisso del valore dell'attributo del gestore code **QSGName** al valore dell'attributo della coda **CFStrucName** .

Questo attributo si applica solo alle code condivise; viene ignorato se *QSGDisp* non ha il valore QSGDSH.

Per determinare il valore di questo attributo, utilizzare il selettore CACFSN con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNCFSN.

z/OS questo attributo è supportato solo su z/OS.

Nome ClusterChannel(stringa di caratteri a 20 byte)

`ClusterChannelNome` è il nome generico dei canali mittenti del cluster che utilizzano questa coda come coda di trasmissione. L'attributo specifica quali canali mittenti del cluster inviano messaggi a un canale ricevente del cluster da questa coda di trasmissione cluster.

Tabella 762. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

La configurazione predefinita del gestore code per tutti i canali mittenti del cluster prevede di inviare i messaggi da una singola coda di trasmissione, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. La configurazione predefinita può essere modificata modificando l'attributo del gestore code, **DefClusterXmitQueueType**. Il valore predefinito dell'attributo è `SCTQ`. È possibile modificare il valore in `CHANNEL`. Se si imposta l'attributo **DefClusterXmitQueueType** su `CHANNEL`, ogni canale mittente del cluster utilizza per impostazione predefinita una specifica coda di trasmissione del cluster. `SYSTEM.CLUSTER.TRANSMIT.`*`ChannelName`*.

È anche possibile impostare l'attributo della coda di trasmissione `ClusterChannelName` su un canale mittente del cluster manualmente. I messaggi destinati al gestore code connesso dal canale mittente del cluster vengono memorizzati nella coda di trasmissione che identifica il canale mittente del cluster. Non vengono memorizzati nella coda di trasmissione del cluster predefinita. Se si imposta l'attributo `ClusterChannelName` su un valore vuoto, il canale passa alla coda di trasmissione del cluster predefinita quando il canale viene riavviato. La coda predefinita è `SYSTEM.CLUSTER.TRANSMIT.`*`ChannelName`* o `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, a seconda del valore dell'attributo del gestore code `DefClusterXmitQueueType`.

Specificando gli asterischi, "`*`", in **ClusterChannelName**, è possibile associare una coda di trasmissione a una serie di canali mittenti del cluster. Gli asterischi possono essere all'inizio, alla fine o in qualsiasi numero di posizioni intermedie della stringa di nome canale. **ClusterChannelName** è limitato a una lunghezza di 20 caratteri: `MQ_CHANNEL_NAME_LENGTH`.

IBM i *ClusterName (stringa di caratteri a 48 byte) su IBM i*

Nome del cluster a cui appartiene la coda.

Tabella 763. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X		X	X	

Questo è il nome del cluster a cui appartiene la coda. Se la coda appartiene a più di un cluster, *ClusterNamelist* specifica il nome di un oggetto elenco nomi che identifica i cluster e *ClusterName* è vuoto. Almeno uno tra *ClusterName* e *ClusterNamelist* deve essere vuoto.

Per determinare il valore di questo attributo, utilizzare il selettore CACLN con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNCLUN.

IBM i *ClusterNamelist (stringa di carattere a 48 byte) su IBM i*

Nome dell'oggetto elenco nomi contenente i nomi dei cluster a cui appartiene la coda.

Tabella 764. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X		X	X	

Questo è il nome di un oggetto elenco nomi che contiene i nomi dei cluster a cui appartiene questa coda. Se la coda appartiene ad un solo cluster, l'oggetto elenco nomi contiene un solo nome. In alternativa, *ClusterName* può essere utilizzato per indicare il nome del cluster, nel qual caso *ClusterNamelist* è vuoto. Almeno uno tra *ClusterName* e *ClusterNamelist* deve essere vuoto.

Per determinare il valore di questo attributo, utilizzare il selettore CACLNL con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNNLN.

IBM i *CreationDate (stringa di caratteri a 12 byte) su IBM i*

Data in cui è stata creata la coda.

Tabella 765. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X				

Questa è la data in cui è stata creata la coda. Il formato della data è YYYY-MM-DD, riempito con due spazi vuoti finali per rendere la lunghezza di 12 byte (ad esempio, 1992-09-23¬¬, dove ¬¬ rappresenta due caratteri vuoti).

- Su IBM i, la data di creazione di una coda potrebbe essere diversa da quella dell'entità del sistema operativo sottostante (file o spazio utente) che rappresenta la coda.

Per determinare il valore di questo attributo, utilizzare il selettore CACRTD con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da LNCRTD.

IBM i *CreationTime (stringa di caratteri a 8 byte) su IBM i*

L'ora in cui la coda è stata creata.

Tabella 766. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X				

Questa è l'ora in cui è stata creata la coda. Il formato dell'ora è HH.MM.SS utilizzando l'orologio di 24 ore, con uno zero iniziale se l'ora è inferiore a 10 (ad esempio 09.10.20). Viene utilizzata l'ora locale.

- Su IBM i, l'ora di creazione di una coda potrebbe essere diversa da quella dell'entità del sistema operativo sottostante (spazio utente o file) che rappresenta la coda.

Per determinare il valore di questo attributo, utilizzare il selettore CACRTT con la chiamata MQINQ. La lunghezza di questo attributo viene fornita da LNCRTT.

IBM i *CurrentQDepth (numero intero con segno a 10 cifre) su IBM i*

Profondità della coda corrente.

Tabella 767. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X				

Questo è il numero di messaggi presenti nella coda. Viene incrementato durante una chiamata MQPUT e durante il backout di una chiamata MQGET. Viene ridotto durante una chiamata MQGET non browse e durante il backout di una chiamata MQPUT. Il risultato è che il conteggio include i messaggi che sono stati inseriti nella coda all'interno di un'unità di lavoro, ma di cui non è stato ancora eseguito il commit, anche se non sono idonei per essere richiamati dalla chiamata MQGET. Allo stesso modo, esclude i messaggi che sono stati richiamati all'interno di un'unità di lavoro utilizzando la chiamata MQGET, ma che devono ancora essere sottoposti a commit.

Il conteggio include anche i messaggi che hanno superato il tempo di scadenza ma non sono stati ancora eliminati, sebbene questi messaggi non siano idonei per essere richiamati. Consultare il campo *MDEXP* descritto in "MQMD (Message Descriptor) su IBM i" a pagina 1140.

L'elaborazione dell'unità di lavoro e la segmentazione dei messaggi possono causare il superamento di *MaxQDepth*da parte di *CurrentQDepth* . Tuttavia, ciò non influisce sulla richiamabilità dei messaggi - *tutti* i messaggi sulla coda possono essere richiamati utilizzando la chiamata MQGET nel modo normale.

Il valore di questo attributo varia quando il gestore code opera.

Per determinare il valore di questo attributo, utilizzare il programma di selezione IACDEP con la chiamata MQINQ.

IBM i *DefBind (numero intero con segno a 10 cifre) su IBM i*

Collegamento predefinito.

Tabella 768. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X		X	X	X

Questo attributo rappresenta il collegamento predefinito utilizzato quando OOBNDQ viene specificato nella chiamata MQOPEN e la coda è una coda cluster. DefBind può avere uno dei seguenti valori:

BNDOPN
Collegamento corretto dalla chiamata MQOPEN.

NON BND
Collegamento non corretto.

BNDGRP
Il bind non è corretto dalla chiamata MQOPEN, ma è corretto su MQPUT per tutti i messaggi in un gruppo logico.

Per determinare il valore di questo attributo, utilizzare il selettore IADBND con la chiamata MQINQ.

IBM i *DefinitionType (numero intero con segno a 10 cifre) su IBM i*

Il tipo di definizione della coda.

Tabella 769. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Indica come è stata definita la coda. Il valore è uno dei seguenti:

QDPRE
Coda permanente predefinita.

La coda è una coda permanente creata dall'amministratore di sistema; solo l'amministratore di sistema può eliminarla.

Le code predefinite vengono create utilizzando il comando MQSC DEFINE e possono essere eliminate solo utilizzando il comando MQSC DELETE . Le code predefinite non possono essere create dalle code modello.

I comandi possono essere emessi da un operatore o da un utente autorizzato che invia un messaggio di comando alla coda di input del comando (consultare l'attributo **CommandInputQName** descritto in “Attributi per il gestore code su IBM i” a pagina 1439).

QDPERM
Coda permanente definita dinamicamente.

La coda è una coda permanente creata da un'applicazione che emette una chiamata MQOPEN con il nome di una coda modello specificata nel descrittore oggetto MQOD. La definizione della coda modello aveva il valore QDPERM per l'attributo **DefinitionType** .

Questo tipo di coda può essere eliminato utilizzando la chiamata MQCLOSE. Per ulteriori dettagli, vedere “MQCLOSE (Chiudi oggetto) su IBM i” a pagina 1304.

Il valore dell'attributo **QSGDisp** per una coda dinamica permanente è QSGDQM.

TDATEMP
Coda temporanea definita dinamicamente.

La coda è una coda temporanea creata da un'applicazione che emette una chiamata MQOPEN con il nome di una coda modello specificata nel descrittore oggetto MQOD. La definizione della coda modello aveva il valore QDTEMP per l'attributo **DefinitionType** .

Questo tipo di coda viene eliminato automaticamente dalla chiamata MQCLOSE quando viene chiusa dall'applicazione che l'ha creato.

Il valore dell'attributo **QSGDisp** per una coda dinamica temporanea è QSGDQM.

QDSHAR
Coda condivisa definita dinamicamente.

La coda è una coda permanente condivisa creata da un'applicazione che emette una chiamata MQOPEN con il nome di una coda modello specificata nel descrittore oggetto MQOD. La definizione della coda modello aveva il valore QDSHAR per l'attributo **DefinitionType** .

Questo tipo di coda può essere eliminato utilizzando la chiamata MQCLOSE. Per ulteriori dettagli, vedere “MQCLOSE (Chiudi oggetto) su IBM i” a pagina 1304.

Il valore dell'attributo **QSGDisp** per una coda dinamica condivisa è QSGDSH.

Questo attributo in una definizione di coda modello non indica come è stata definita la coda modello, poiché le code modello sono sempre predefinite. Invece, il valore di questo attributo nella coda modello

viene utilizzato per determinare il *DefinitionType* di ciascuna delle code dinamiche create dalla definizione della coda modello utilizzando la chiamata MQOPEN.

Per determinare il valore di questo attributo, utilizzare il selettore IADEFT con il richiamo MQINQ.

IBM i *DefInputOpenOption (numero intero con segno a 10 cifre) su IBM i*

Opzione predefinita di apertura in immissione.

Tabella 770. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Questo è il modo predefinito in cui la coda deve essere aperta per l'input. Si applica se l'opzione OOINPQ viene specificata nella chiamata MQOPEN quando la coda viene aperta. Può avere uno dei seguenti valori:

OOINPX
Aprire la coda per ottenere i messaggi con accesso esclusivo.

La coda viene aperta per essere utilizzata con successive chiamate MQGET. La chiamata non riesce con codice di errore RC2042 se la coda è attualmente aperta da questa o da un'altra applicazione per l'input di qualsiasi tipo (OOINPS o OOINPX).

OOINPS
Aprire la coda per richiamare i messaggi con accesso condiviso.

La coda viene aperta per essere utilizzata con successive chiamate MQGET. La chiamata può avere esito positivo se la coda è attualmente aperta da questa o un'altra applicazione con OOINPS, ma ha esito negativo con codice motivo RC2042 se la coda è attualmente aperta con OOINPX.

Per determinare il valore di questo attributo, utilizzare il programma di selezione IADINP con la chiamata MQINQ.

IBM i *DefPersistence (numero intero con segno a 10 cifre) su IBM i*

Persistenza del messaggio predefinita.

Tabella 771. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X	X	X	X

Questa è la persistenza predefinita dei messaggi sulla coda. Si applica se PEQDEF è specificato nel descrittore del messaggio quando il messaggio viene inserito.

Se è presente più di una definizione nel percorso di risoluzione del nome coda, la persistenza predefinita viene presa dal valore di questo attributo nella *prima* definizione nel percorso al momento della chiamata MQPUT o MQPUT1 . È possibile che si tratti di:

- Una coda alias
- Una coda locale
- Una definizione locale di una coda remota
- Un alias del gestore code
- Una coda di trasmissione (ad esempio, la coda *DefXmitQName*)

Può avere uno dei seguenti valori:

PEPER
Il messaggio è persistente.

Ciò significa che il messaggio sopravvive agli errori di sistema e ai riavvii del gestore code. I messaggi persistenti non possono essere posizionati su:

- Code dinamiche temporanee
- Code condivise

I messaggi persistenti possono essere posizionati su code dinamiche permanenti e code predefinite.

PENPER
Il messaggio non è persistente.

Ciò significa che il messaggio di solito non sopravvive agli errori di sistema o ai riavvii del gestore code. Ciò si applica anche se viene trovata una copia intatta del messaggio nella memoria ausiliaria durante il riavvio del gestore code.

Nel caso speciale di code condivise, i messaggi non persistenti *sopravvivono* ai riavvii dei gestori code nel gruppo di condivisione code, ma non sopravvivono agli errori della CF utilizzata per memorizzare i messaggi nelle code condivise.

Sia i messaggi persistenti che quelli non persistenti possono esistere nella stessa coda.

Per determinare il valore di questo attributo, utilizzare il selettore IADPER con la chiamata MQINQ.

IBM i *DefPriority (numero intero con segno a 10 cifre) su IBM i*

La priorità messaggi predefinita.

Tabella 772. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X	X	X	X

Questa è la priorità predefinita per i messaggi sulla coda. Ciò si applica se PRQDEF viene specificato nel descrittore del messaggio quando il messaggio viene inserito nella coda.

Se esiste più di una definizione nel percorso di risoluzione del nome coda, la priorità predefinita per il messaggio viene presa dal valore di questo attributo nella *prima* definizione nel percorso al momento dell'operazione di inserimento. È possibile che si tratti di:

- Una coda alias
- Una coda locale
- Una definizione locale di una coda remota
- Un alias del gestore code
- Una coda di trasmissione (ad esempio, la coda *DefXmitQName*)

Il modo in cui un messaggio viene inserito in una coda dipende dal valore dell'attributo **MsgDeliverySequence** della coda:

- Se l'attributo **MsgDeliverySequence** è MSPRIO, la posizione logica in cui un messaggio viene posizionato sulla coda dipende dal valore del campo *MDPRI* nel descrittore del messaggio.
- Se l'attributo **MsgDeliverySequence** è MSFIFO, i messaggi vengono collocati nella coda come se avessero una priorità uguale a *DefPriority* della coda risolta, indipendentemente dal valore del campo *MDPRI* nel descrittore del messaggio. Tuttavia, il campo *MDPRI* conserva il valore specificato dall'applicazione che ha inserito il messaggio. Per ulteriori informazioni, consultare l'attributo **MsgDeliverySequence** descritto in “Attributi per le code” a pagina 1406 .

Le priorità sono comprese nell'intervallo tra zero (minimo) e *MaxPriority* (massimo); consultare l'attributo **MaxPriority** descritto in “Attributi per il gestore code su IBM i” a pagina 1439.

Per determinare il valore di questo attributo, utilizzare il selettore IAD PRI con la chiamata MQINQ.

IBM i *DefReadAhead (numero intero con segno a 10 cifre) su IBM i*

Specifica il comportamento di lettura anticipata predefinito per i messaggi non permanenti consegnati al client.

Tabella 773. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X	X		

DefReadPuò essere impostato su uno dei seguenti valori:

NUMERO
I messaggi non persistenti non vengono inviati in anticipo al client prima che un'applicazione li richieda. Un massimo di un messaggio non persistente può andare perduto se il client termina in maniera irregolare.

RAHSÌ
I messaggi non persistenti vengono inviati in anticipo al client prima che un'applicazione li richieda. I messaggi non persistenti possono essere persi se il client termina in modo anomalo o se il client non utilizza tutti i messaggi inviati.

RAHDIS
La lettura anticipata dei messaggi non persistenti non è abilitata per questa coda. I messaggi non vengono inviati in anticipo al client indipendentemente dal fatto che la lettura anticipata sia richiesta dall'applicazione client.

Per determinare il valore di questo attributo, utilizzare il programma di selezione IADRAH con la chiamata MQINQ.

IBM i *DefPResp (numero intero con segno a 10 cifre) su IBM i*

L'attributo tipo di risposta put predefinito (DEFPRESP) definisce il valore utilizzato dalle applicazioni quando il tipo PutResponsein MQPMO è stato impostato su PMRASQ. Questo attributo è valido per tutti i tipi di coda.

Tabella 774. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X	X	X	X

Può avere uno dei seguenti valori:

SINC
L'operazione di inserimento viene emessa in modo sincrono restituendo una risposta.

ASINC
L'operazione di inserimento viene eseguita in modo asincrono, restituendo una sottoserie di campi MQMD.

Per determinare il valore di questo attributo, utilizzare il selettore IADPRT con la chiamata MQINQ.

IBM i *DistLists (numero intero con segno a 10 cifre) su IBM i*

Supporto elenco di distribuzione.

Tabella 775. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Indica se i messaggi dell'elenco di distribuzione possono essere inseriti nella coda. L'attributo viene impostato da un agente MCA (Message Channel Agent) per informare il gestore code locale se il gestore code all'altra estremità del canale supporta gli elenchi di distribuzione. Quest' ultimo gestore code (denominato "gestore code partner") è quello che riceve successivamente il messaggio, dopo che è stato rimosso dalla coda di trasmissione locale da un MCA mittente.

L'attributo viene impostato dall'MCA mittente ogni volta che si stabilisce una connessione all'MCA ricevente sul gestore code partner. In questo modo, l'MCA mittente può far sì che il gestore code locale posizioni nella coda di trasmissione solo i messaggi che il gestore code partner può elaborare correttamente.

Questo attributo viene utilizzato principalmente con le code di trasmissione, ma l'elaborazione descritta viene eseguita indipendentemente dall'utilizzo definito per la coda (consultare l'attributo **Usage**).

Può avere uno dei seguenti valori:

DLSUPP
Elenchi di distribuzione supportati.

Ciò indica che i messaggi dell'elenco di distribuzione possono essere memorizzati nella coda e trasmessi al gestore code partner in tale modulo. Ciò riduce la quantità di elaborazione richiesta per inviare il messaggio a più destinazioni.

DLNSUP
Elenchi di distribuzione non supportati.

Ciò indica che i messaggi dell'elenco di distribuzione non possono essere archiviati nella coda, perché il gestore code partner non supporta gli elenchi di distribuzione. Se un'applicazione inserisce un messaggio dell'elenco di distribuzione e tale messaggio deve essere inserito in questa coda, il gestore code suddivide il messaggio dell'elenco di distribuzione e inserisce i singoli messaggi nella coda. Ciò aumenta la quantità di elaborazione richiesta per inviare il messaggio a più destinazioni, ma garantisce che i messaggi vengano elaborati correttamente dal gestore code partner.

Per determinare il valore di questo attributo, utilizzare il selettore IADIST con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET.

IBM i *HardenGetBackout (numero intero con segno a 10 cifre) su IBM i*

Indica se mantenere un conteggio di backout accurato.

Tabella 776. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Per ogni messaggio, viene mantenuto un conteggio del numero di volte in cui il messaggio viene richiamato da una chiamata MQGET all'interno di un'unità di lavoro e tale unità di lavoro viene successivamente ripristinata. Questo conteggio è disponibile nel campo *MDBOC* nel descrittore del messaggio una volta completata la chiamata MQGET.

Il conteggio di backout dei messaggi sopravvive quando il gestore code viene riavviato. Tuttavia, per garantire che il conteggio sia accurato, le informazioni devono essere "rinforzate" (registrate sul disco o su un altro dispositivo di memorizzazione permanente) ogni volta che un messaggio viene richiamato da una chiamata MQGET all'interno di un'unità di lavoro per questa coda. Se questa operazione non viene eseguita e si verifica un malfunzionamento del gestore code insieme al backout della chiamata MQGET, il conteggio potrebbe non essere incrementato.

Tuttavia, il potenziamento delle informazioni per ogni chiamata MQGET all'interno di un'unità di lavoro impone un costo delle prestazioni e l'attributo **HardenGetBackout** deve essere impostato su QABH solo se il conteggio deve essere accurato.

- Su IBM i, il conteggio di backout del messaggio è sempre forzato, indipendentemente dall'impostazione di questo attributo.

Sono possibili i seguenti valori:

QABH
Conteggio di backout ricordato.

Il potenziamento viene utilizzato per garantire che il numero di backout per i messaggi su questa coda sia accurato.

QABNH
Il conteggio di backout potrebbe non essere ricordato.

Il potenziamento non viene utilizzato per garantire che il numero di backout per i messaggi su questa coda sia accurato. Il conteggio potrebbe quindi essere inferiore a quello che dovrebbe essere.

Per determinare il valore di questo attributo, utilizzare il selettore IAHGB con la chiamata MQINQ.

IBM i *InhibitGet (numero intero con segno a 10 cifre) il IBM i*

Controlla se le operazioni di acquisizione per questa coda sono consentite.

Tabella 777. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X	X		

Se la coda è una coda alias, le operazioni di acquisizione devono essere consentite sia per l'alias che per la coda di base al momento dell'operazione di acquisizione, affinché la chiamata MQGET abbia esito positivo. Il valore è uno dei seguenti:

QAGETI
Le operazioni get sono inibite.

Le chiamate MQGET hanno esito negativo con codice motivo RC2016. Questo include le chiamate MQGET che specificano GMBRWF o GMBRWN.

Nota: Se una chiamata MQGET che opera all'interno di un'unità di lavoro viene completata correttamente, la modifica del valore dell'attributo **InhibitGet** in QAGETI non impedisce il commit dell'unità di lavoro.

QAGETA
Le operazioni get sono consentite.

Per determinare il valore di questo attributo, utilizzare il selettore IAIGET con il richiamo MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET.

IBM i *InhibitPut (numero intero con segno a 10 cifre) su IBM i*

Controlla se le operazioni di inserimento per questa coda sono consentite.

Tabella 778. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X	X	X	X

Se nel percorso di risoluzione del nome della coda è presente più di una definizione, le operazioni di inserimento devono essere consentite per *ogni* definizione del percorso (incluse le definizioni di alias del gestore code) al momento dell'operazione di inserimento, affinché la chiamata MQPUT o MQPUT1 abbia esito positivo. Può avere uno dei seguenti valori:

DAPUTI
Le operazioni di inserimento sono inibite.

Le chiamate MQPUT e MQPUT1 hanno esito negativo con codice motivo RC2051.

Nota: Se una chiamata MQPUT che opera all'interno di un'unità di lavoro viene completata correttamente, la modifica del valore dell'attributo **InhibitPut** successivamente in QAPUTI non impedisce il commit dell'unità di lavoro.

QAPUTA
Le operazioni di inserimento sono consentite.

Per determinare il valore di questo attributo, utilizzare il selettore IAIPUT con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET.

IBM i *InitiationQName (stringa di caratteri a 48 byte) su IBM i*

Nome della coda di iniziazione.

Tabella 779. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Questo è il nome di una coda definita sul gestore code locale; la coda deve essere di tipo QTLOC. Il gestore code invia un messaggio trigger alla coda di avvio quando l'avvio dell'applicazione è richiesto come risultato dell'arrivo di un messaggio sulla coda a cui appartiene questo attributo. La coda di avvio deve essere monitorata da un'applicazione di controllo trigger che avvierà l'applicazione appropriata dopo aver ricevuto il messaggio di trigger.

Per determinare il valore di questo attributo, utilizzare il selettore CAINIQ con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQN.

IBM i *MaxMsgLunghezza (numero intero con segno a 10 cifre) su IBM i*

Lunghezza massima del messaggio in byte.

Tabella 780. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Questo è un limite superiore per la lunghezza del messaggio *fisico* più lungo che può essere inserito nella coda. Tuttavia, poiché l'attributo della coda **MaxMsgLength** può essere impostato indipendentemente dall'attributo del gestore code **MaxMsgLength** , il limite superiore effettivo per la lunghezza del messaggio fisico più lungo che può essere inserito nella coda è il minore di questi due valori.

Se il gestore code supporta la segmentazione, è possibile per un'applicazione inserire un messaggio *logico* più lungo del minore dei due attributi **MaxMsgLength** , ma solo se l'applicazione specifica l'indicatore MFSEGA in MQMD. Se viene specificato tale indicatore, il limite superiore per la lunghezza di un messaggio logico è 999 999 999 999 byte, ma di solito, i vincoli di risorsa imposti dal sistema operativo o dall'ambiente in cui l'applicazione è in esecuzione, risultano in un limite inferiore.

Un tentativo di inserire nella coda un messaggio troppo lungo ha esito negativo con il codice di errore:

- RC2030 se il messaggio è troppo grande per la coda
- RC2031 se il messaggio è troppo grande per il gestore code, ma non troppo grande per la coda

Il limite inferiore per l'attributo **MaxMsgLength** è zero. Il limite superiore è determinato dall'ambiente:

- Su IBM i, la lunghezza massima del messaggio è 100 MB (104 857 600 byte).

Per ulteriori informazioni, consultare il parametro **BUFLEN** descritto in "MQPUT (Inserisci messaggio) su IBM i" a pagina 1370.

Per determinare il valore di questo attributo, utilizzare il selettore IAMLEN con la chiamata MQINQ.

IBM i *MaxQDepth (numero intero con segno a 10 cifre) su IBM i*

Profondità massima della coda.

Tabella 781. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Si tratta del limite superiore definito per il numero di messaggi fisici che possono esistere sulla coda contemporaneamente. Un tentativo di inserire un messaggio su una coda che già contiene *MaxQDepth* messaggi, ha esito negativo con codice di errore RC2053.

L'elaborazione dell'unità di lavoro e la segmentazione dei messaggi possono causare il superamento di *MaxQDepth*da parte del numero effettivo di messaggi fisici sulla coda. Tuttavia, ciò non influisce sulla richiamabilità dei messaggi - *tutti* i messaggi sulla coda possono essere richiamati utilizzando la chiamata MQGET nel modo normale.

Il valore di questo attributo è zero o maggiore. Il limite superiore è determinato dall'ambiente.

Nota: È possibile che lo spazio di memoria disponibile per la coda sia esaurito anche se nella coda sono presenti meno di *MaxQDepth* messaggi.

Per determinare il valore di questo attributo, utilizzare il selettore IAMDEP con la chiamata MQINQ.

IBM i *MediaLog (numero intero con segno a 10 cifre) su IBM i*

Identità dell'estensione della registrazione (o ricevitore di giornale su IBM i) necessario per il recupero dei supporti di una particolare coda.

Tabella 782. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Sui gestori code in cui è in uso la registrazione circolare, il valore viene restituito come una stringa nulla.

IBM i *MsgDeliverySequenza (numero intero con segno a 10 cifre) su IBM i*

La sequenza di distribuzione dei messaggi.

Tabella 783. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Ciò determina l'ordine in cui i messaggi vengono restituiti all'applicazione dalla chiamata MQGET:

MFIFO
I messaggi vengono restituiti in ordine FIFO (first in, first out).

Ciò significa che una chiamata MQGET restituirà il *primo* messaggio che soddisfa i criteri di selezione specificati sulla chiamata, indipendentemente dalla priorità del messaggio.

MSPRIO
I messaggi vengono restituiti in ordine di priorità.

Ciò significa che una chiamata MQGET restituirà il messaggio *priorità più alta* che soddisfa i criteri di selezione specificati nella chiamata. All'interno di ciascun livello di priorità, i messaggi vengono restituiti in ordine FIFO (first in, first out).

Se gli attributi rilevanti vengono modificati mentre sono presenti messaggi nella coda, la sequenza di consegna è la seguente:

- L'ordine in cui i messaggi vengono restituiti dalla chiamata MQGET è determinato dai valori degli attributi **MsgDeliverySequence** e **DefPriority** in vigore per la coda nel momento in cui il messaggio arriva sulla coda:
  - Se *MsgDeliverySequence* è MSFIFO quando arriva il messaggio, il messaggio viene inserito nella coda come se la sua priorità fosse *DefPriority*. Ciò non influisce sul valore del campo *MDPRI* nel descrittore del messaggio; quel campo conserva il valore che aveva quando il messaggio è stato inserito per la prima volta.
  - Se *MsgDeliverySequence* è MSPRIO quando arriva il messaggio, il messaggio viene posizionato nella coda nella posizione appropriata alla priorità data dal campo *MDPRI* nel descrittore del messaggio.

  Se il valore dell'attributo **MsgDeliverySequence** viene modificato mentre sono presenti messaggi nella coda, l'ordine dei messaggi nella coda non viene modificato.

  Se il valore dell'attributo **DefPriority** viene modificato mentre ci sono messaggi nella coda, i messaggi non verranno necessariamente consegnati in ordine FIFO, anche se l'attributo **MsgDeliverySequence** è impostato su MSFIFO; quelli collocati nella coda con priorità più alta vengono consegnati per primi.

Per determinare il valore di questo attributo, utilizzare il selettore IAMDS con la chiamata MQINQ.

IBM i *OpenInputConteggio (numero intero con segno a 10 cifre) il IBM i*

Numero di aperture per l'input.

Tabella 784. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X				

Questo è il numero di handle attualmente validi per la rimozione dei messaggi dalla coda con la chiamata MQGET. È il numero totale di tali handle noti al gestore code *locale* . Se la coda è una coda condivisa, il conteggio non include le aperture per l'input eseguite per la coda su altri gestori code nel gruppo di condivisione code a cui appartiene il gestore code locale.

Il conteggio include gli handle in cui una coda alias che si risolve in questa coda è stata aperta per l'input. Il conteggio non include gli handle in cui la coda è stata aperta per le azioni che non includevano l'input (ad esempio, una coda aperta solo per la ricerca).

Il valore di questo attributo varia quando il gestore code opera.

Per determinare il valore di questo attributo, utilizzare il selettore IAOIC con la chiamata MQINQ.

IBM i *OpenOutputConteggio (numero intero con segno a 10 cifre) il IBM i*

Numero di aperture per l'output.

Tabella 785. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X				

Questo è il numero di handle attualmente validi per l'aggiunta di messaggi alla coda con la chiamata MQPUT. È il numero totale di tali handle noti al gestore code *locale* ; non include le aperture per l'output eseguite per questa coda sui gestori code remoti. Se la coda è una coda condivisa, il conteggio non include le aperture per l'output eseguite per la coda su altri gestori code nel gruppo di condivisione code a cui appartiene il gestore code locale.

Il conteggio include gli handle in cui una coda alias che si risolve in questa coda è stata aperta per l'output. Il conteggio non include gli handle in cui la coda è stata aperta per le azioni che non includevano l'output (ad esempio, una coda aperta solo per l'interrogazione).

Il valore di questo attributo varia quando il gestore code opera.

Per definire il valore di questo attributo, utilizzare il selettore IAOOC con la chiamata MQINQ.

IBM i *ProcessName (stringa di caratteri a 48 byte) su IBM i*

Il nome del processo.

Tabella 786. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Questo è il nome di un oggetto processo definito nel gestore code locale. L'oggetto processo identifica un programma che può servire la coda.

Per determinare il valore di questo attributo, utilizzare il selettore CAPRON con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNPRON.

IBM i *QDepthHighEvento (numero intero con segno a 10 cifre) su IBM i*

Controlla se vengono generati eventi Grandezza coda elevata.

Tabella 787. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Un evento Profondità massima coda indica che un'applicazione ha inserito un messaggio in una coda, il che ha fatto sì che il numero di messaggi nella coda diventasse maggiore o uguale alla soglia massima di profondità della coda (consultare l'attributo **QDepthHighLimit**).

Nota: Il valore di questo attributo può essere modificato dinamicamente.

L'evento QDepthHighpuò avere uno dei seguenti valori:

EVRDIS
Report eventi disabilitato.

EVRENA
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore IAQDHE con la chiamata MQINQ.

IBM i *QDepthHighLimite (numero intero con segno a 10 cifre) su IBM i*

Limite massimo per la profondità della coda.

Tabella 788. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Si tratta della soglia rispetto alla quale la profondità della coda viene confrontata per generare un evento Grandezza coda elevata. Questo evento indica che un'applicazione ha inserito un messaggio in una coda e che il numero di messaggi nella coda è diventato maggiore o uguale alla soglia massima di profondità della coda. Consultare l'attributo **QDepthHighEvent** .

Il valore è espresso come percentuale della profondità massima della coda (attributo**MaxQDepth**) ed è compreso tra zero e 100. Il valore predefinito è 80.

Per determinare il valore di questo attributo, utilizzare il selettore IAQDHL con la chiamata MQINQ.

IBM i *QDepthLowEvento (numero intero con segno a 10 cifre) su IBM i*

Controlla se vengono generati eventi Profondità minima coda.

Tabella 789. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Un evento Profondità minima coda indica che un'applicazione ha richiamato un messaggio da una coda, il che ha fatto sì che il numero di messaggi nella coda diventasse inferiore o uguale alla soglia di profondità minima della coda (consultare l'attributo **QDepthLowLimit**).

Nota: Il valore di questo attributo può essere modificato dinamicamente.

QDepthLowL'evento può avere uno dei seguenti valori:

EVRDIS
Report eventi disabilitato.

EVRENA
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore IAQDLE con la chiamata MQINQ.

IBM i *QDepthLowLimite (numero intero con segno a 10 cifre) su IBM i*

Limite basso per la profondità della coda.

Tabella 790. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Questa è la soglia rispetto alla quale la profondità della coda viene confrontata per generare un evento Grandezza coda bassa. Questo evento indica che un'applicazione ha richiamato un messaggio da una coda e ciò ha fatto sì che il numero di messaggi sulla coda diventasse inferiore o uguale alla soglia inferiore della profondità della coda. Consultare l'attributo **QDepthLowEvent** .

Il valore è espresso come percentuale della profondità massima della coda (attributo**MaxQDepth**) ed è compreso tra zero e 100. Il valore predefinito è 20.

Per determinare il valore di questo attributo, utilizzare il selettore IAQDLL con la chiamata MQINQ.

IBM i *QDepthMaxEvento (numero intero con segno a 10 cifre) su IBM i*

Controlla se vengono generati eventi Coda piena.

Tabella 791. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Un evento Coda piena indica che un inserimento in una coda è stato rifiutato perché la coda è piena, ovvero la profondità della coda ha già raggiunto il valore massimo.

Nota: Il valore di questo attributo può essere modificato dinamicamente.

Può avere uno dei seguenti valori:

EVRDIS
Report eventi disabilitato.

EVRENA
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore IAQDME con la chiamata MQINQ.

IBM i *QDesc (stringa di caratteri a 64 byte) su IBM i*

Descrizione della coda.

Tabella 792. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X	X	X	X

Questo è un campo che può essere utilizzato per commenti descrittivi. Il contenuto del campo non ha alcun significato per il gestore code, ma il gestore code potrebbe richiedere che il campo contenga solo caratteri che possono essere visualizzati. Non può contenere caratteri null; se necessario, viene riempito a destra con spazi. In un'installazione DBCS, il campo può contenere caratteri DBCS (con una lunghezza massima di 64 byte).

Nota: Se questo campo contiene caratteri non presenti nella serie di caratteri del gestore code (come definito dall'attributo del gestore code **CodedCharSetId**), tali caratteri potrebbero essere tradotti in modo non corretto se questo campo viene inviato a un altro gestore code.

Per determinare il valore di questo attributo, utilizzare il selettore CAQD con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQD.

IBM i *QName (stringa di caratteri a 48 byte) su IBM i*

Il nome della coda.

Tabella 793. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X		X	X	X

Questo è il nome di una coda definita sul gestore code locale. Per ulteriori informazioni sui nomi delle code, consultare Regole per la denominazione degli oggetti IBM MQ. Tutte le code definite su un gestore code condividono lo stesso spazio dei nomi della coda. Pertanto, una coda QTLOC e una coda QTALS non possono avere lo stesso nome.

Per determinare il valore di questo attributo, utilizzare CAQN selector con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQN.

IBM i *QServiceInterval (numero intero con segno a 10 cifre) su IBM i*

Destinazione dell'intervallo di servizio della coda.

Tabella 794. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Questo è l'intervallo di servizio utilizzato per il confronto per generare eventi Intervallo di servizio elevato e Intervallo di servizio OK. Consultare l'attributo **QServiceIntervalEvent** .

Il valore è espresso in unità di millisecondi ed è compreso tra zero e 999 999 999.

Per determinare il valore di questo attributo, utilizzare il selettore IAQSI con la chiamata MQINQ.

IBM i *QServiceIntervalEvento (numero intero con segno a 10 cifre) su IBM i*

Controlla se vengono generati eventi Intervallo di servizio elevato o Intervallo di servizio OK.

Tabella 795. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

- Un evento Intervallo servizio elevato viene generato quando un controllo indica che non sono stati richiamati messaggi dalla coda per almeno il tempo indicato dall'attributo **QServiceInterval** .
- Un evento Intervallo di servizio OK viene generato quando un controllo indica che i messaggi sono stati richiamati dalla coda entro il tempo indicato dall'attributo **QServiceInterval** .

Nota: Il valore di questo attributo può essere modificato dinamicamente.

Questo attributo può avere uno dei seguenti valori:

QSIEHI
Eventi di intervallo del servizio coda alto abilitati.

- Gli eventi Intervallo servizio coda elevato sono **abilitati** e

- Gli eventi OK dell'intervallo di servizio della coda sono **disabilitati**.

QSIEOK
Eventi di intervallo del servizio coda OK abilitati.

- Gli eventi Intervallo servizio coda elevato sono **disabilitati** e
- Gli eventi OK dell'intervallo di servizio della coda sono **abilitati**.

QSIENO
Nessun evento di intervallo del servizio coda abilitato.

- Gli eventi Intervallo servizio coda elevato sono **disabilitati** e
- Gli eventi OK dell'intervallo di servizio della coda sono **disabilitati**.

Per le code condivise, il valore di questo attributo viene ignorato; viene assunto il valore QSIENO.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore IAQSIE con la chiamata MQINQ.

IBM i *QSGDisp (numero intero con segno a 10 cifre) su IBM i*

Disposizione del gruppo di condivisione code.

Tabella 796. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X		X	X	

Specifica la disposizione della coda. Il valore è uno dei seguenti:

QSGDQM
Disposizione gestore code.

L'oggetto ha la disposizione del gestore code. Ciò significa che la definizione dell'oggetto è nota solo al gestore code locale; la definizione non è nota ad altri gestori code nel gruppo di condivisione code.

È possibile per ogni gestore code nel gruppo di condivisione code avere un oggetto con lo stesso nome e tipo dell'oggetto corrente, ma si tratta di oggetti separati e non esiste alcuna correlazione tra di essi. I loro attributi non sono vincolati ad essere gli stessi.

CSGDCP
Disposizione oggetto copiato.

L'oggetto è una copia locale di una definizione di oggetto principale che esiste nel repository condiviso. Ogni gestore code nel gruppo di condivisione code può avere la propria copia dell'oggetto. Inizialmente, tutte le copie hanno gli stessi attributi, ma utilizzando i comandi MQSC ciascuna copia può essere modificata in modo che i propri attributi differiscano da quelli delle altre copie. Gli attributi delle copie vengono risincronizzati quando la definizione principale nel repository condiviso viene modificata.

DSDSGQ
Disposizione condivisa.

L'oggetto presenta una disposizione condivisa. Ciò significa che esiste nel repository condiviso una singola istanza dell'oggetto nota a tutti i gestori code nel gruppo di condivisione code. Quando un gestore code nel gruppo accede all'oggetto, accede alla sola istanza condivisa dell'oggetto.

Per stabilire il valore di questo attributo, utilizzare il selettore IAQSG con la chiamata MQINQ.

z/OS questo attributo è supportato solo su z/OS.

IBM i *QType (numero intero con segno a 10 cifre) su IBM i*

Il tipo di coda.

Tabella 797. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X		X	X	X

Questo attributo può presentare uno dei seguenti valori:

TALS
Definizione coda alias.

QTCLUS
Coda cluster.

TLOC
Coda locale.

TREM
Definizione locale di una coda remota.

Per determinare il valore di questo attributo, utilizzare il selettore IAQTYP con la chiamata MQINQ.

IBM i *RemoteQMgrNome (stringa di caratteri a 48 byte) su IBM i*

Il nome del gestore code remoto.

Tabella 798. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
			X	

Questo è il nome del gestore code remoto su cui è definita la coda *RemoteQName* . Se la coda *RemoteQName* ha un valore *QSGDisp* QSGDCP o QSGDSH, *RemoteQMgrName* può essere il nome del gruppo di condivisione code che possiede *RemoteQName*.

Se un'applicazione apre la definizione locale di una coda remota, *RemoteQMgrName* non deve essere vuoto e non deve essere il nome del gestore code locale. Se *XmitQName* è vuoto, la coda locale con lo stesso nome di *RemoteQMgrName* viene utilizzata come coda di trasmissione. Se non è presente alcuna coda con il nome *RemoteQMgrName*, viene utilizzata la coda identificata dall'attributo del gestore code **DefXmitQName** .

Se questa definizione viene utilizzata per un alias del gestore code, *RemoteQMgrName* è il nome del gestore code di cui si sta eseguendo l'alias. Può essere il nome del gestore code locale. In caso contrario, se *XmitQName* è vuoto quando si verifica l'apertura, deve essere presente una coda locale con lo stesso nome di *RemoteQMgrName* ; questa coda viene utilizzata come coda di trasmissione.

Se questa definizione viene utilizzata per un alias di risposta, questo nome è il nome del gestore code che deve essere *MDRM*.

Nota: Non viene eseguita alcuna convalida sul valore specificato per questo attributo quando la definizione della coda viene creata o modificata.

Per determinare il valore di questo attributo, utilizzare il selettore CARQMN con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQMN.

IBM i *RemoteQName (stringa di caratteri a 48 byte) su IBM i*

Nome della coda remota.

Tabella 799. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
			X	

Questo è il nome della coda come è noto sul gestore code remoto *RemoteQMgrName*.

Se un'applicazione apre la definizione locale di una coda remota, quando si verifica l'apertura *RemoteQName* non deve essere vuoto.

Se questa definizione viene utilizzata per una definizione di alias del gestore code, quando si verifica l'apertura *RemoteQName* deve essere vuoto.

Se la definizione viene utilizzata per un alias di risposta, questo nome è il nome della coda che deve essere *MDRQ*.

Nota: Non viene eseguita alcuna convalida sul valore specificato per questo attributo quando la definizione della coda viene creata o modificata.

Per determinare il valore di questo attributo, utilizzare CARQN selector con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQN.

IBM i *RetentionInterval (numero intero con segno a 10 cifre) su IBM i*

L'intervallo di mantenimento.

Tabella 800. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Questa è l'orario in cui la coda deve essere conservata. Una volta trascorso questo tempo, la coda è idonea per l'eliminazione.

L'ora viene misurata in ore, contando la data e ora in cui è stata creata la coda. La data di creazione della coda viene registrata in *CreationDate* e l'ora di creazione della coda viene registrata in **CreationTime** .

Queste informazioni vengono fornite per consentire a un'applicazione di manutenzione o all'operatore di identificare ed eliminare le code che non sono più richieste.

Nota: Il gestore code non tenta mai di eliminare le code in base a questo attributo o di impedire l'eliminazione delle code con un intervallo di conservazione non scaduto; è responsabilità dell'utente far sì che venga eseguita qualsiasi azione richiesta.

Un intervallo di conservazione realistico deve essere utilizzato per impedire l'accumulo delle code dinamiche permanenti (consultare *DefinitionType*). Tuttavia, questo attributo può essere utilizzato anche con code predefinite.

Per determinare il valore di questo attributo, utilizzare il selettore IARINT con la chiamata MQINQ.

IBM i *Ambito (numero intero con segno a 10 cifre) su IBM i*

Controlla se una voce per questa coda esiste anche in una directory della cella.

Tabella 801. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X		X	X	

Una directory di celle viene fornita da un Servizio nomi installabile. Può avere uno dei seguenti valori:

SCOQM
Ambito del gestore code.

La definizione della coda ha un ambito gestore code. Ciò significa che la definizione della coda non si estende oltre il gestore code proprietario. Per aprire la coda per l'output da un altro gestore code, è necessario specificare il nome del gestore code proprietario oppure l'altro gestore code deve avere una definizione locale della coda.

SCOCEL
Ambito cella.

La definizione della coda ha un ambito cella. Ciò significa che la definizione di coda viene inserita anche in una directory di celle disponibile per tutti i gestori code nella cella. La coda può essere aperta per l'output da qualsiasi gestore code nella cella semplicemente specificando il nome della coda; non è necessario specificare il nome del gestore code che possiede la coda. Tuttavia, la definizione della coda non è disponibile per alcun gestore code nella cella che ha anche una definizione locale di una coda con quel nome, poiché la definizione locale ha la precedenza.

Una directory di celle viene fornita da un servizio di nomi installabile come LDAP (Lightweight Directory Access Protocol. Si noti che IBM MQ non supporta più il servizio nomi DCE (Distributed Computing Environment) precedentemente utilizzato per l'inserimento di definizioni di coda in una directory DCE (non più supportato).

Le code modello e dinamiche non possono avere l'ambito della cella.

Questo valore è valido solo se è stato configurato un servizio nomi che supporta una directory della cella.

Per determinare il valore di questo attributo, utilizzare il selettore IASCOP con la chiamata MQINQ.

Il supporto per questo attributo è soggetto alle seguenti limitazioni:

- Su IBM i, l'attributo è supportato, ma è valido solo SCOQM.

IBM i *Condividibilità (numero intero con segno a 10 cifre) su IBM i*

Indica se la coda può essere condivisa per l'input.

Tabella 802. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Indica se la coda può essere aperta per l'input più volte contemporaneamente. Può avere uno dei seguenti valori:

QASR
La coda è condivisibile.

Sono consentite più aperture con l'opzione OOINPS.

QANSHR
La coda non è condivisibile.

Una chiamata MQOPEN con l'opzione OOINPS viene trattata come OOINPX.

Per determinare il valore di questo attributo, utilizzare il selettore IASHAR con la chiamata MQINQ.

IBM i *TriggerControl (numero intero con segno a 10 cifre) su IBM i*

Controllo trigger.

Tabella 803. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Ciò controlla se i messaggi di trigger vengono scritti in una coda di iniziazione, in modo da far sì che un'applicazione venga avviata per servire la coda. Il valore è uno dei seguenti:

COFF
Messaggi di trigger non richiesti.

Nessun messaggio trigger deve essere scritto per questa coda. Il valore di *TriggerType* è irrilevante in questo caso.

TCON
Messaggi trigger richiesti.

I messaggi trigger devono essere scritti per questa coda, quando si verificano gli eventi trigger appropriati.

Per determinare il valore di questo attributo, utilizzare il selettore IATRGC con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET.

IBM i *TriggerData (stringa di caratteri a 64 byte) su IBM i*

I dati del trigger.

Tabella 804. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Si tratta di dati in formato libero che il gestore code inserisce nel messaggio trigger quando un messaggio in arrivo su questa coda causa la scrittura di un messaggio trigger nella coda di avvio.

Il contenuto di questi dati non è significativo per il gestore code. È significativo per l'applicazione di controllo dei trigger che elabora la coda di iniziazione o per l'applicazione avviata dal controllo dei trigger.

La stringa di caratteri non può contenere valori null. Se necessario, viene riempito a destra con spazi vuoti.

Per determinare il valore di questo attributo, utilizzare il selettore CATRGD con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET. La lunghezza di questo attributo è fornita da LNTRGD.

IBM i *TriggerDepth (numero intero con segno a 10 cifre) su IBM i*

La lunghezza del trigger.

Tabella 805. Tipi di coda a cui si applica questo attributo				
Locale	**Modello**	**Alias**	**Remoto**	**Cluster**
X	X			

Questo è il numero di messaggi con priorità *TriggerMsgPriority* o superiore che devono essere sulla coda prima che venga scritto un messaggio trigger. Questo si applica quando *TriggerType* è impostato su TTDPTH. Il valore di *TriggerDepth* è uno o superiore. Questo attributo non viene utilizzato altrimenti.

Per determinare il valore di questo attributo, utilizzare il selettore IATRGD con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET.

IBM i *TriggerMsgPriorità (numero intero con segno a 10 cifre) su IBM i*

Priorità del messaggio di soglia per i trigger su IBM MQ for IBM i.

Tabella 806. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Questa è la priorità del messaggio al di sotto della quale i messaggi non contribuiscono alla generazione di messaggi trigger (ossia, il gestore code ignora questi messaggi quando determina se deve essere generato un messaggio trigger). *TriggerMsgPriority* può essere compreso nell'intervallo tra zero (il più basso) e *MaxPriority* (il più alto; consultare "Attributi per il gestore code su IBM i" a pagina 1439); un valore zero fa sì che tutti i messaggi contribuiscano alla generazione di messaggi trigger.

Per determinare il valore di questo attributo, utilizzare il selettore IATRGP con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET.

IBM i *TriggerType (numero intero con segno a 10 cifre) su IBM i*

Il tipo di trigger.

Tabella 807. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Ciò controlla le condizioni in cui i messaggi trigger vengono scritti come risultato dei messaggi in arrivo su questa coda. Il valore è uno dei seguenti:

TTNONE
Nessun messaggio trigger.

Non viene scritto alcun messaggio trigger come risultato dei messaggi su questa coda. Questo ha lo stesso effetto dell'impostazione di *TriggerControl* su TCOFF.

TTFRST
Attiva messaggio quando la profondità della coda è compresa tra 0 e 1.

Un messaggio trigger viene scritto ogni volta che il numero di messaggi con priorità *TriggerMsgPriority* o maggiore sulla coda cambia da 0 a 1.

TTEVRY
Messaggio trigger per ogni messaggio.

Un messaggio trigger viene scritto ogni volta che un messaggio di priorità *TriggerMsgPriority* o superiore arriva sulla coda.

TTDPTH
Messaggio trigger quando viene superata la soglia di profondità.

Un messaggio di trigger viene scritto ogni volta che il numero di messaggi con priorità *TriggerMsgPriority* o superiore sulla coda è uguale o superiore a *TriggerDepth*. Dopo che

il messaggio di trigger è stato scritto, *TriggerControl* viene impostato su TCOFF per impedire ulteriori trigger fino a quando non viene esplicitamente attivato di nuovo.

Per determinare il valore di questo attributo, utilizzare il selettore IATRGT con la chiamata MQINQ. Per modificare il valore di questo attributo, utilizzare la chiamata MQSET.

IBM i *Utilizzo (numero intero con segno a 10 cifre) su IBM i*

Utilizzo della coda.

Tabella 808. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
X	X			

Indica per cosa viene utilizzata la coda. Il valore è uno dei seguenti:

USNORM
Utilizzo normale.

Si tratta di una coda che le normali applicazioni utilizzano durante l'inserimento e il richiamo di messaggi; la coda non è una coda di trasmissione.

USTRAN
Coda di trasmissione.

Questa è una coda utilizzata per contenere i messaggi destinati ai gestori code remoti. Quando un'applicazione normale invia un messaggio a una coda remota, il gestore code locale memorizza temporaneamente il messaggio nella coda di trasmissione appropriata in un formato speciale. Un agente del canale dei messaggi legge quindi il messaggio dalla coda di trasmissione e lo trasporta al gestore code remoto. Per ulteriori informazioni sulle code di trasmissione, consultare Code di trasmissione.

Solo le applicazioni con privilegi possono aprire una coda di trasmissione per OOOUT per inserire direttamente i messaggi. Solo le applicazioni del programma di utilità normalmente dovrebbero eseguire questa operazione. È necessario fare attenzione che il formato dei dati del messaggio sia corretto (consultare “MQXQH (Transmission - queue header) su IBM i” a pagina 1281), altrimenti potrebbero verificarsi errori durante il processo di trasmissione. Il contesto non viene passato o impostato a meno che non venga specificata una delle opzioni di contesto PM*.

Per determinare il valore di questo attributo, utilizzare il selettore IAUSAG con la chiamata MQINQ.

IBM i *XmitQName (stringa di caratteri a 48 byte) su IBM i*

Il nome della coda di trasmissione.

Tabella 809. Tipi di coda a cui si applica questo attributo

Locale	Modello	Alias	Remoto	Cluster
			X	

Se questo attributo non è vuoto quando si verifica un'apertura, per una coda remota o per una definizione dell'alias del gestore code, specifica il nome della coda di trasmissione locale da utilizzare per l'inoltro del messaggio.

Se *XmitQName* è vuoto, la coda locale con lo stesso nome di *RemoteQMgrName* viene utilizzata come coda di trasmissione. Se non è presente alcuna coda con il nome *RemoteQMgrName*, viene utilizzata la coda identificata dall'attributo del gestore code **DefXmitQName** .

Questo attributo viene ignorato se la definizione viene utilizzata come alias del gestore code e *RemoteQMgrName* è il nome del gestore code locale. Viene ignorato anche se la definizione è utilizzata come una definizione di alias di coda risposta.

Per determinare il valore di questo attributo, utilizzare il selettore CAXQN con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQN.

Attributi per gli elenchi nomi

Questo argomento riepiloga gli attributi specifici degli elenchi nomi. Gli attributi sono descritti in ordine alfabetico.

Nota: I nomi degli attributi visualizzati sono i nomi utilizzati con le chiamate MQINQ e MQSET.

Descrizioni attributo

Un oggetto elenco nomi ha i seguenti attributi:

AlterationDate (stringa di caratteri a 12 byte)

Data dell'ultima modifica della definizione.

Questa è la data dell'ultima modifica della definizione. Il formato della data è YYYY-MM-DD, riempito con due spazi finali per rendere la lunghezza di 12 byte.

Per determinare il valore di questo attributo, utilizzare il selettore CAALTD con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNDATE.

AlterationTime (stringa di caratteri a 8 byte)

L'ora dell'ultima modifica della definizione.

Questa è l'ora dell'ultima modifica della definizione. Il formato dell'ora è HH.MM.SS.

Per determinare il valore di questo attributo, utilizzare il selettore CAALTT con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNTIME.

NameCount (numero intero con segno a 10 cifre)

Numero di nomi nell'elenco nomi.

È maggiore o uguale a zero. Viene definito il seguente valore:

NCMXNL
Numero massimo di nomi in un elenco nomi.

Per stabilire il valore di questo attributo, utilizzare il selettore IANAMC con la chiamata MQINQ.

NamelistDesc (stringa di caratteri a 64 byte)

Descrizione elenco nomi.

Questo è un campo che potrebbe essere utilizzato per i commenti descrittivi; il suo valore è stabilito dal processo di definizione. Il contenuto del campo non ha alcun significato per il gestore code, ma il gestore code potrebbe richiedere che il campo contenga solo caratteri che possono essere visualizzati. Non può contenere caratteri null; se necessario, viene riempito a destra con spazi. In un'installazione DBCS, questo campo può contenere caratteri DBCS (con una lunghezza massima di 64 byte).

Nota: Se questo campo contiene caratteri non presenti nella serie di caratteri del gestore code (come definito dall'attributo del gestore code **CodedCharSetId**), tali caratteri potrebbero essere tradotti in modo non corretto se questo campo viene inviato a un altro gestore code.

Per determinare il valore di questo attributo, utilizzare il selettore CALSTD con la chiamata MQINQ.

La lunghezza di questo attributo è fornita da LNNLD.

NamelistName (stringa di caratteri a 48 byte)

Il nome dell'elenco dei nomi.

Questo è il nome di un elenco nomi definito nel gestore code locale.

Ogni elenco nomi ha un nome diverso dai nomi di altri elenchi nomi appartenenti al gestore code, ma potrebbe duplicare i nomi di altri oggetti gestore code di tipi differenti (ad esempio, code).

Per determinare il valore di questo attributo, utilizzare il selettore CALSTN con la chiamata MQINQ.

La lunghezza di questo attributo è fornita da LNNLN.

Nomi (stringa di caratteri a 48 byte x NameCount)

Un elenco di nomi *NameCount* .

Ogni nome è il nome di un oggetto definito sul gestore code locale. Per ulteriori informazioni sui nomi oggetto, consultare Denominazione degli oggetti IBM MQ.

Per determinare il valore di questo attributo, utilizzare il programma di selezione CANAMS con la chiamata MQINQ.

La lunghezza di ogni nome nell'elenco è fornita da LNOBJN.

IBM i Attributi per le definizioni di processo su IBM i .

Questo argomento riepiloga gli attributi specifici per le definizioni di processi. Gli attributi sono descritti in ordine alfabetico.

Nota: I nomi degli attributi visualizzati sono i nomi utilizzati con le chiamate MQINQ e MQSET. Quando i comandi MQSC vengono utilizzati per definire, modificare o visualizzare gli attributi, vengono utilizzati nomi brevi alternativi; per i dettagli, consultare Comandi MQSC .

Descrizioni attributo

Un oggetto definizione processo ha i seguenti attributi:

AlterationDate (stringa di caratteri a 12 byte)

Data dell'ultima modifica della definizione.

Questa è la data dell'ultima modifica della definizione. Il formato della data è YYYY-MM-DD, riempito con due spazi finali per rendere la lunghezza di 12 byte.

Per determinare il valore di questo attributo, utilizzare il selettore CAALTD con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNDATE.

AlterationTime (stringa di caratteri a 8 byte)

L'ora dell'ultima modifica della definizione.

Questa è l'ora dell'ultima modifica della definizione. Il formato dell'ora è HH.MM.SS.

Per determinare il valore di questo attributo, utilizzare il selettore CAALTT con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNTIME.

ApplId (stringa di caratteri a 256 byte)

Identificativo applicazione.

Si tratta di una stringa di caratteri che identifica l'applicazione da avviare. Queste informazioni vengono utilizzate da un'applicazione di controllo trigger che elabora i messaggi sulla coda di iniziazione; le informazioni vengono inviate alla coda di iniziazione come parte del messaggio trigger.

Il significato di *ApplId* è determinato dall'applicazione trigger - monitor. Il controllo dei trigger fornito da IBM MQ richiede *ApplId* come nome di un programma eseguibile.

La stringa di caratteri non può contenere valori null. Se necessario, viene riempito a destra con spazi vuoti.

Per determinare il valore di questo attributo, utilizzare il selettore CAAPPI con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNPROA.

ApplType (numero intero con segno a 10 cifre)

Il tipo di applicazione.

Identifica la natura del programma da avviare in risposta alla ricezione di un messaggio trigger. Queste informazioni vengono utilizzate da un'applicazione di controllo trigger che elabora i messaggi sulla coda di iniziazione; le informazioni vengono inviate alla coda di iniziazione come parte del messaggio trigger.

ApplType può avere qualsiasi valore. È possibile utilizzare i seguenti valori per i tipi standard; i tipi di applicazione definiti dall'utente sono limitati ai valori compresi nell'intervallo tra ATUFST e ATULST:

ACICS
Transazione CICS .

AT400
Applicazione IBM i .

ATUFST
Il valore più basso per il tipo di applicazione definito dall'utente.

ATULST
Il valore più alto per il tipo di applicazione definito dall'utente.

Per determinare il valore di questo attributo, utilizzare il selettore IAAPPT con la chiamata MQINQ.

EnvData (stringa di caratteri a 128 byte)

Dati di ambiente.

Si tratta di una stringa di caratteri che contiene informazioni relative all'ambiente relative all'applicazione da avviare. Queste informazioni vengono utilizzate da un'applicazione di controllo trigger che elabora i messaggi sulla coda di iniziazione; le informazioni vengono inviate alla coda di iniziazione come parte del messaggio trigger.

Il significato di *EnvData* è determinato dall'applicazione trigger - monitor. Il controllo trigger fornito da IBM MQ accoda *EnvData* all'elenco di parametri passato all'applicazione avviata. L'elenco dei parametri è composto dalla struttura MQTMC2 , seguita da uno spazio vuoto, seguita da *EnvData* con spazi vuoti finali rimossi.

La stringa di caratteri non può contenere valori null. Se necessario, viene riempito a destra con spazi vuoti.

Per determinare il valore di questo attributo, utilizzare il selettore CAENVD con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNPROE.

ProcessDesc (stringa di caratteri a 64 byte)

Descrizione processo.

Questo è un campo che può essere utilizzato per commenti descrittivi. Il contenuto del campo non ha alcun significato per il gestore code, ma il gestore code potrebbe richiedere che il campo contenga solo caratteri che possono essere visualizzati. Non può contenere caratteri null; se necessario, viene riempito a destra con spazi. In un'installazione DBCS, il campo può contenere caratteri DBCS (con una lunghezza massima di 64 byte).

Nota: Se questo campo contiene caratteri non presenti nella serie di caratteri del gestore code (come definito dall'attributo del gestore code **CodedCharSetId**), tali caratteri potrebbero essere tradotti in modo non corretto se questo campo viene inviato a un altro gestore code.

Per determinare il valore di questo attributo, utilizzare il selettore CAPROD con la chiamata MQINQ.

La lunghezza di questo attributo è fornita da LNPROD.

ProcessName (stringa di caratteri a 48 byte)

Il nome del processo.

Questo è il nome di una definizione di processo definita nel gestore code locale.

Ciascuna definizione di processo ha un nome diverso da quello di altre definizioni di processi appartenenti al gestore code. Ma il nome della definizione del processo può essere uguale ai nomi di altri oggetti gestore code di tipi differenti (ad esempio, code).

Per determinare il valore di questo attributo, utilizzare il selettore CAPRON con la chiamata MQINQ.

La lunghezza di questo attributo è fornita da LNPRON.

UserData (stringa di caratteri a 128 byte)

Dati utente.

Questa è una stringa di caratteri che contiene le informazioni utente relative all'applicazione da avviare. Queste informazioni vengono utilizzate da un'applicazione di controllo trigger che elabora i messaggi sulla coda di iniziazione o dall'applicazione avviata dal controllo trigger. Le informazioni vengono inviate alla coda di iniziazione come parte del messaggio trigger.

Il significato di *UserData* è determinato dall'applicazione trigger - monitor. Il controllo trigger fornito da IBM MQ passa *UserData* all'applicazione avviata come parte dell'elenco di parametri. L'elenco dei parametri è costituito dalla struttura MQTMC2 (contenente *UserData*), seguita da uno spazio vuoto, seguito da *EnvData* con spazi vuoti finali rimossi.

La stringa di caratteri non può contenere valori null. Se necessario, viene riempito a destra con spazi vuoti.

Per determinare il valore di questo attributo, utilizzare il selettore CAUSRD con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNPROU.

IBM i Attributi per il gestore code su IBM i

Un riepilogo degli attributi del gestore code.

Alcuni attributi del gestore code sono corretti per particolari implementazioni, mentre altri possono essere modificati utilizzando il comando MQSC `ALTER QMGR`. Gli attributi possono essere visualizzati anche utilizzando il comando `DISPLAY QMGR`. La maggior parte degli attributi del gestore code può essere interrogata aprendo un oggetto OTQM speciale e utilizzando la chiamata MQINQ con l'handle restituito.

La seguente tabella riepiloga gli attributi specifici del gestore code. Gli attributi sono descritti in ordine alfabetico.

Nota: I nomi degli attributi visualizzati in questa sezione sono i nomi utilizzati con le chiamate MQINQ e MQSET. Quando i comandi MQSC vengono utilizzati per definire, modificare o visualizzare gli attributi, vengono utilizzati nomi brevi alternativi; per ulteriori informazioni, consultare Comandi MQSC .

Tabella 810. Attributi per il gestore code

Attributo	Descrizione
AlterationDate	Data dell'ultima modifica della definizione
AlterationTime	Ora dell'ultima modifica della definizione
AuthorityEvent	Controlla se vengono generati eventi di autorizzazione (non autorizzati)
BridgeEvent	Controlla se vengono generati eventi bridge IMS
ChannelAutoDef	Controlla se la definizione di canale automatica è consentita
ChannelAutoDefEvent	Controlla se vengono generati eventi di definizione automatica del canale
ChannelAutoDefExit	Nome dell'uscita utente per la definizione di canale automatica

Tabella 810. Attributi per il gestore code (Continua)	
Attributo	**Descrizione**
ChannelEvent	Controlla se vengono generati eventi del canale
ClusterCacheClusterCache	Controlla se la dimensione della cache del cluster è fissa o dinamicamente ridimensionata
ClusterWorkloadData	Dati utente per uscita carico di lavoro cluster
ClusterWorkloadExit	Nome dell'uscita utente per la gestione del carico di lavoro del cluster
ClusterWorkloadLength	Lunghezza massima dei dati del messaggio passati all'exit del carico di lavoro cluster
CodedCharSetId	CCSID (Coded character set identifier)
CommandEvent	Controlla se i messaggi di evento comando sono accodati
CommandInputQName	Nome coda di input comandi
CommandLevel	Livello di comandi
ConfigurationEvent	Evento di configurazione
DeadLetterQName	Nome della coda di messaggi non recapitabili
DefClusterXmitQueuecoda trasmissione	Tipo coda di trasmissione cluster predefinito
DefXmitQName	Nome coda di trasmissione predefinita
DistLists	Supporto elenco di distribuzione
InhibitEvent	Controlla se vengono generati eventi di inibizione (inibisci ricezione e inibisci immissione)
LocalEvent	Controlla se vengono generati eventi di errori locali
LoggerEvent	Controlla se vengono generati eventi di log di recupero
MaxHandles	Il numero massimo di puntatori
MaxMsgLength	La lunghezza massima del messaggio in byte
MaxPriority	Priorità massima
MaxUncommittedMsgs	Numero massimo di messaggi di cui non è stato eseguito il commit in un'unità di lavoro
PerformanceEvent	Controlla se vengono generati eventi relativi alle prestazioni
Piattaforma	La piattaforma su cui è in esecuzione il gestore code
PubSubMode	Se il motore di pubblicazione / sottoscrizione e l'interfaccia di pubblicazione / sottoscrizione accodata sono in esecuzione
QMgrDesc	La descrizione del gestore code
QMgrIdentifier	Identificativo univoco generato internamente del gestore code
QMgrName	Nome del gestore code
RemoteEvent	Controlla se vengono generati eventi di errore remoti
RepositoryName	Nome del cluster per il quale questo gestore code fornisce i servizi del repository

Tabella 810. Attributi per il gestore code (Continua)	
Attributo	**Descrizione**
RepositoryNamelist	Nome dell'oggetto elenco nomi contenente i nomi dei cluster per i quali questo gestore code fornisce i servizi del repository
SSLCRLNamelist	Il nome dell'oggetto elenco nomi contenente i nomi degli oggetti delle informazioni di autenticazione (vedere la nota 1)
SSLEvent	Controlla se vengono generati eventi TLS
SSLKeyRepository	Ubicazione del repository delle chiavi TLS (vedere la nota 1)
SSLKeyResetConteggio	Determina il numero di byte non codificati inviati e ricevuti all'interno di una conversazione TLS prima che la chiave di codifica venga rinegoziata
StartStopEvent	Controlla se vengono generati eventi di avvio e arresto
SyncPoint	Disponibilità Syncpoint
TraceRouteRecording	Controlla la registrazione delle informazioni di instradamento della traccia per i messaggi
TreeLifeTime	La durata, in secondi, degli argomenti non amministrativi
TriggerInterval	Intervallo messaggio trigger
Note: 1. Questo attributo non può essere interrogato utilizzando la chiamata MQINQ e non è descritto in questa sezione. Per ulteriori informazioni su questo attributo, consultare Modifica gestore code.	

IBM i *AlterationDate (stringa di caratteri a 12 byte) su IBM i*

Data dell'ultima modifica della definizione.

Questa è la data dell'ultima modifica della definizione. Il formato della data è YYYY-MM-DD, riempito con due spazi finali per rendere la lunghezza di 12 byte.

Per determinare il valore di questo attributo, utilizzare il selettore CAALTD con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNDATE.

IBM i *AlterationTime (stringa di caratteri a 8 byte) su IBM i*

L'ora dell'ultima modifica della definizione.

Questa è l'ora dell'ultima modifica della definizione. Il formato dell'ora è HH.MM.SS.

Per determinare il valore di questo attributo, utilizzare il selettore CAALTT con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNTIME.

IBM i *AuthorityEvent (numero intero con segno a 10 cifre) su IBM i*

Controlla se vengono generati eventi di autorizzazione (non autorizzati).

L'attributo AuthorityEvent deve essere impostato su uno dei seguenti valori:

EVRDIS
Report eventi disabilitato.

EVRENA
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore IAAUTE con la chiamata MQINQ.

IBM i *BridgeEvent (stringa di caratteri) su IBM i*

Questo attributo determina se i messaggi di evento bridge IMS vengono inseriti nel sistema SYSTEM.ADMIN.CHANNEL.EVENT . È supportato solo su z/OS.

IBM i *ChannelAutoDef (numero intero con segno a 10 cifre) su IBM i*

Controlla se la definizione di canale automatica è consentita.

Questo attributo controlla la definizione automatica dei canali di tipo CTRCVR e CTSVCN. Si noti che la definizione automatica dei canali CTCLSD è sempre abilitata. Può avere uno dei seguenti valori:

CHADDI
Definizione automatica del canale disabilitata.

CHADEN
Definizione automatica canale abilitata.

Per determinare il valore di questo attributo, utilizzare il selettore IACAD con la chiamata MQINQ.

IBM i *ChannelAutoDefEvent (numero intero con segno a 10 cifre) su IBM i*

Controlla se vengono generati eventi di definizione automatica del canale.

Questo si applica ai canali di tipo CTRCVR, CTSVCN e CTCLSD. Può avere uno dei seguenti valori:

EVRDIS
Report eventi disabilitato.

EVRENA
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Monitoraggio e prestazioni.

Per determinare il valore di questo attributo, utilizzare il selettore IAC ADE con la chiamata MQINQ.

IBM i *ChannelAutoDefExit (stringa di caratteri a 20 byte) su IBM i*

Nome dell'uscita utente per la definizione di canale automatica.

Se questo nome non è vuoto e *ChannelAutoDef* ha il valore CHADEN, l'exit viene richiamata ogni volta che il gestore code sta per creare una definizione di canale. Questo si applica ai canali di tipo CTRCVR, CTSVCN e CTCLSD. L'uscita può quindi effettuare una delle seguenti operazioni:

- Consentire alla creazione della definizione di canale di procedere senza modifiche.
- Modificare gli attributi della definizione di canale creata.
- Sopprimere completamente la creazione del canale.

Per stabilire il valore di questo attributo, utilizzare il selettore CACADX con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNEXN.

IBM i *ChannelEvent (stringa di caratteri) su IBM i*

Determina se vengono generati i messaggi di evento del canale.

Questo attributo determina se i messaggi di evento del canale vengono inseriti nel SISTEMA SYSTEM.ADMIN.CHANNEL.EVENT , e in tal caso, quale tipo di messaggi sono accodati (ad esempio 'canale avviato', 'canale arrestato', 'canale non attivato'). Prima dell'implementazione di questo attributo, l'unico modo per impedire l'accodamento dei messaggi di evento del canale era eliminare la coda di destinazione.

Questo attributo consente inoltre di raccogliere solo gli eventi bridge IMS (poiché ora è possibile disattivare gli eventi canale, non vengono inseriti nella stessa coda). Lo stesso vale per gli eventi TLS che possono essere raccolti anche senza dover raccogliere gli eventi del canale.

Questo attributo consente inoltre di raccogliere solo eventi significativi (ad esempio, quando i canali presentano errori, non quando vengono avviati e arrestati normalmente).

Il valore dell'attributo ChannelEvent può essere uno dei seguenti:

- EVREXP (vengono generati solo i seguenti eventi canale: RC2279, RC2283, RC2284, RC2295, RC2296).
- EVRENA (vengono generati tutti gli eventi del canale; ovvero, oltre agli eventi generati da EVREXP, vengono generati anche gli eventi RC2282e gli eventi RC2283).
- EVRDIS (non viene generato alcun evento del canale; questo è il valore predefinito iniziale del gestore code).

Per determinare il valore di questo attributo, utilizzare il selettore IACHNE con la chiamata MQINQ.

IBM i *ClusterCacheTipo (stringa di caratteri a 32 byte) su IBM i*

Controlla se la cache del cluster è a dimensione fissa o è ridimensionata dinamicamente.

Si tratta di una stringa di caratteri a 32 byte definita dall'utente che viene passata all'uscita del workload del cluster quando viene richiamata. Se non ci sono dati da passare all'uscita, la stringa è vuota.

Per determinare il valore di questo attributo, utilizzare il selettore CACLWD con la chiamata MQINQ.

IBM i *ClusterWorkloadDati (stringa di caratteri a 32 byte) su IBM i*

Dati utente per uscita carico di lavoro cluster.

Si tratta di una stringa di caratteri a 32 byte definita dall'utente che viene passata all'uscita del workload del cluster quando viene richiamata. Se non ci sono dati da passare all'uscita, la stringa è vuota.

Per determinare il valore di questo attributo, utilizzare il selettore CACLWD con la chiamata MQINQ.

IBM i *ClusterWorkloadExit (stringa di caratteri a 20 byte) su IBM i*

Nome dell'uscita utente per la gestione del workload del cluster.

Se questo nome non è vuoto, l'uscita viene richiamata ogni volta che un messaggio viene inserito in una coda cluster o spostato da una coda mittente cluster a un'altra. L'uscita può quindi accettare l'istanza della coda selezionata dal gestore code come destinazione del messaggio oppure selezionare un'altra istanza della coda.

Per determinare il valore di questo attributo, utilizzare il selettore CACLWX con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNEXN.

IBM i *ClusterWorkloadLunghezza (numero intero con segno a 10 cifre) su IBM i*

La lunghezza massima dei dati del messaggio passati all'uscita del carico di lavoro del cluster.

Questa è la lunghezza massima dei dati del messaggio passati all'uscita del workload del cluster. La lunghezza effettiva dei dati passati all'uscita è il minimo dei seguenti:

- La lunghezza del messaggio.
- L'attributo **MaxMsgLength** del gestore code.
- L'attributo **ClusterWorkloadLength** .

Per determinare il valore di questo attributo, utilizzare il selettore IACLWL con la chiamata MQINQ.

IBM i *CodedCharSetId (numero intero con segno a 10 cifre) su IBM i*

Coded character set identifier (CCSID)

Definisce la serie di caratteri utilizzata dal gestore code per tutti i campi stringa di caratteri definiti nell'MQI, ad esempio i nomi degli oggetti e la data e ora di creazione della coda. La serie di caratteri deve essere una serie di caratteri a byte singolo per i caratteri validi nei nomi oggetto. Non si applica ai dati dell'applicazione trasmessi nel messaggio. Il valore dipende dall'ambiente:

- Su IBM i, il valore è quello impostato nell'ambiente quando il gestore code viene creato per la prima volta.

Per determinare il valore di questo attributo, utilizzare il selettore IACCSI con la chiamata MQINQ.

IBM i *CommandEvent (intero) su IBM i*

Controlla se i messaggi vengono inseriti in una coda locale quando vengono emessi i comandi.

Controlla se i messaggi vengono scritti in una nuova coda eventi, SYSTEM.ADMIN.COMMAND.EVENT, ogni volta che vengono emessi comandi. Questa funzione è utile per la notifica della traccia dei comandi e per la diagnosi dei problemi. Per informazioni sull'attributo del gestore code CommandEvent , utilizzare il nuovo selettore di attributi iacev con uno dei seguenti valori:

- EVRENA - i messaggi di evento del comando vengono generati e inseriti nella coda per tutti i comandi riusciti.
- I messaggi di evento del comando EVND vengono generati e inseriti nella coda per tutti i comandi riusciti diversi dal comando DISPLAY (MQSC) e dal comando Inquire (PCF).
- EVRDIS - i messaggi di evento del comando non vengono generati o inseriti nella coda (questo è il valore predefinito iniziale del gestore code).

Per determinare il valore di questo attributo, utilizzare il selettore CMDEV con la chiamata MQINQ.

IBM i *CommandInputQName (stringa di caratteri a 48 byte) su IBM i*

Il nome della coda di immissione del comando.

CommandInputQName è il nome della coda di immissione comandi definita sul gestore code locale. Si tratta di una coda a cui gli utenti possono inviare comandi, se autorizzati. Il nome della coda dipende dall'ambiente:

- Su IBM i, il nome della coda è SYSTEM.ADMIN.COMMAND.QUEUEe solo i comandi PCF possono essere inviati. Tuttavia, un comando MQSC può essere inviato a questa coda se il comando MQSC è racchiuso all'interno di un comando PCF di tipo CMESC. Per ulteriori informazioni sul comando Escape, consultare Escape.

Per determinare il valore di questo attributo, utilizzare il selettore CACMDQ con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQN.

IBM i *CommandLevel (numero intero con segno a 10 cifre) su IBM i*

Il livello dei comandi. Indica il livello dei comandi di controllo del sistema supportati dal gestore code.

Il livello è uno dei seguenti valori:

CML800
Livello 800 dei comandi di controllo del sistema.

Questo valore viene restituito dalle applicazioni seguenti:

- IBM MQ for IBM i
  - Versione 8.0

CML900
Livello 900 dei comandi di controllo del sistema.

Questo valore viene restituito dalle applicazioni seguenti:

- IBM MQ for IBM i
  - Versione 9.0

CML910
Livello 910 dei comandi di controllo del sistema.

Questo valore viene restituito dalle applicazioni seguenti:

- IBM MQ for IBM i
  - Versione 9.1

CML920
Livello 920 dei comandi di controllo del sistema.

Questo valore viene restituito dalle applicazioni seguenti:

- IBM MQ for IBM i
  - Versione 9.2

CML930
Livello 930 dei comandi di controllo del sistema.

Questo valore viene restituito dalle applicazioni seguenti:

- IBM MQ for IBM i
  - Versione 9.3

La serie di comandi di controllo del sistema che corrisponde ad un determinato valore dell'attributo **CommandLevel** varia in base al valore dell'attributo **Platform** ; entrambi devono essere utilizzati per decidere quali comandi di controllo del sistema sono supportati.

Per stabilire il valore di questo attributo, utilizzare il selettore IACMDL con la chiamata MQINQ.

IBM i *ConfigurationEvent su IBM i*

Controlla se gli eventi di configurazione vengono generati e inviati al SISTEMA SYSTEM.ADMIN.CONFIG.EVENT .

L'attributo ConfigurationEvent può essere uno dei seguenti valori:

- EVRENA
- EVRDIS

Se l'attributo ConfigurationEvent è impostato su EVRENA e alcuni comandi vengono correttamente emessi da runmqsc o PCF, gli eventi di configurazione vengono generati e inviati a SYSTEM.ADMIN.CONFIG.EVENT . Vengono emessi eventi per i seguenti comandi, anche se un comando di modifica non modifica l'oggetto coinvolto. I comandi per cui vengono generati e inviati gli eventi di configurazione sono:

- DEFINE/ALTER AUTINFO
- DEFINIZIONE/MODIFICA CANALE
- ELENCO NOMI DEFINIZIONE / ALTER
- DEFINE/ALTER PROCESSO
- DEFINE/ALTER QLOCAL (a meno che non sia una coda dinamica temporanea)
- DEFINE/ALTER QMODEL/QALIAS/QREMOTE
- DELETE AUTINFO
- Elimina canale
- Eliminazione elenco nomi
- Eliminazione processo
- DELETE QLOCAL (a meno che non si tratti di una coda dinamica temporanea)
- ELIMINARE QMODEL/QALIAS/QREMOTE
- ALTER QMGR (a meno che l'attributo CONFIGEV non sia disabilitato e non venga modificato in abilitato)
- AGGIORNA QMGR
- Una chiamata MQSET, diversa da quella per una coda dinamica temporanea.

Gli eventi non vengono generati (se abilitati) nelle seguenti circostanze:

- Il comando o la chiamata MQSET non riesce.
- Il gestore code non può inserire il messaggio evento nella coda eventi. Il comando deve ancora essere completato correttamente.
- Code dinamiche temporanee.

- Modifiche dell'attributo interno effettuate direttamente o implicitamente (non da MQSET o dal comando); ciò influisce su TRIGGER, CURDEPTH, IPPROCS, OPPROCS, QDPHIEV, QDPLOEV, QDPMAXEV, QSVCIEV.
- Quando la coda di eventi di configurazione viene modificata, anche se verrà generato un messaggio di evento per tale modifica quando viene richiesto un aggiornamento.
- Modifiche al cluster mediante i comandi REFRESH/RESET CLUSTER e RESUME/SUSPEND QMGR.
- Creazione o eliminazione di un gestore code.

IBM i *DeadLetterQName (stringa di caratteri a 48 byte) su IBM i*

Il nome della coda di messaggi non recapitabili (non recapitabili).

Questo è il nome di una coda definita sul gestore code locale. I messaggi vengono inviati a questa coda se non possono essere instradati alla destinazione corretta.

Ad esempio, i messaggi vengono inseriti in questa coda quando:

- Un messaggio arriva a un gestore code, destinato a una coda non ancora definita su tale gestore code
- Un messaggio arriva a un gestore code, ma la coda a cui è destinato non può riceverlo perché:
  - La coda è piena
  - Le richieste di inserimento sono inibite
  - Il nodo di invio non dispone dell'autorità per inserire i messaggi nella coda

Le applicazioni possono anche inserire messaggi nella coda di messaggi non recapitabili.

I messaggi di report vengono trattati allo stesso modo dei messaggi ordinari; se il messaggio di report non può essere consegnato alla relativa coda di destinazione (in genere la coda specificata dal campo *MDRQ* nel descrittore del messaggio originale), il messaggio di report viene posizionato nella coda di messaggi non recapitabili (messaggio non recapitato).

Nota: I messaggi che hanno superato la loro scadenza (consultare il campo *MDEXP* descritto in "MQMD (Message Descriptor) su IBM i" a pagina 1140) non vengono trasferiti a questa coda quando vengono eliminati. Tuttavia, un messaggio di report di scadenza (ROEXP) viene ancora generato e inviato alla coda *MDRQ* , se richiesto dall'applicazione mittente.

I messaggi non vengono inseriti nella coda dei messaggi non recapitati (messaggio non recapitato) quando l'applicazione che ha emesso la richiesta di inserimento ha ricevuto una notifica sincrona del problema con il codice motivo restituito dalla chiamata MQPUT o MQPUT1 (ad esempio, un messaggio inserito in una coda locale per cui le richieste di inserimento sono inibite).

I messaggi sulla coda dei messaggi non recapitabili (messaggi non recapitati) a volte hanno come prefisso i dati del messaggio dell'applicazione con una struttura MQDLH. Questa struttura contiene informazioni aggiuntive che indicano il motivo per cui il messaggio è stato inserito nella coda dei messaggi non recapitabili (non recapitati). Per ulteriori dettagli su questa struttura, consultare "MQDLH (Intestazione non instradabile) su IBM i" a pagina 1093 .

Questa coda deve essere una coda locale, con un attributo **Usage** di USNORM.

Se una coda di messaggi non recapitabili non è supportata da un gestore code o non è stata definita, il nome è vuoto. Tutti i gestori code IBM MQ supportano una coda di messaggi non recapitabili (messaggi non recapitabili), ma per impostazione predefinita non è definita.

Se la coda dei messaggi non recapitabili (messaggio non recapito) non è definita, è piena o inutilizzabile per qualche altro motivo, un messaggio che sarebbe stato trasferito ad essa da un agente del canale dei messaggi viene conservato invece nella coda di trasmissione.

Per determinare il valore di questo attributo, utilizzare il selettore CADLQ con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQN.

DefClusterXmitQueueTipo (numero intero con segno a 10 cifre)

L'attributo DefClusterXmitQueue controlla la coda di trasmissione selezionata per impostazione predefinita dai canali mittenti del cluster da cui richiamare i messaggi, per inviare i messaggi ai canali riceventi del cluster.

I valori di **DefClusterXmitQueueType** sono MQCLXQ_SCTQ o MQCLXQ_CHANNEL.

MQCLXQ_SCTQ

Tutti i canali mittenti del cluster inviano messaggi da SYSTEM.CLUSTER.TRANSMIT.QUEUE. Il correlID dei messaggi inseriti nella coda di trasmissione identifica a quale canale mittente del cluster è destinato il messaggio.

SCTQ viene impostato quando viene definito un gestore code.

MQCLXQ_CHANNEL

Ogni canale mittente del cluster invia messaggi da una coda di trasmissione differente. Ciascuna coda di trasmissione viene creata come una coda dinamica permanente dalla coda modello SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE.

Se l'attributo del gestore code, DefClusterXmitQueueTipo, è impostato su CHANNEL, la configurazione predefinita viene modificata nei canali mittenti del cluster che vengono associati alle singole code di trasmissione cluster. Le code di trasmissione sono code dinamiche permanenti create a partire dalla coda modello SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE. Ciascuna coda di trasmissione è associata a un canale mittente del cluster. Poiché un canale mittente del cluster serve una coda di trasmissione, la coda di trasmissione contiene messaggi per un solo gestore code in un cluster. È possibile configurare i cluster in modo che ogni gestore code in un cluster contenga una sola coda cluster. In questo caso, il traffico di messaggi da un gestore code a ogni coda del cluster viene trasferito separatamente dai messaggi alle altre code.

Per interrogare il valore, richiamare MQINQo inviare un comando PCF MQCMD_INQUIRE_Q_MGR(Inquire Queue Manager), impostando il selettore MQIA_DEF_CLUSTER_XMIT_Q_TYPE . Per modificare il valore, inviare un comando PCF del gestore code di modifica (MQCMD_CHANGE_Q_MGR), impostando il selettore MQIA_DEF_CLUSTER_XMIT_Q_TYPE .

Riferimenti correlati

Modifica gestore code

Interrogazione gestore code

“MQINQ (Interroga sugli attributi dell'oggetto) su IBM i” a pagina 1342

La chiamata MQINQ restituisce un array di numeri interi e una serie di stringhe di caratteri contenenti gli attributi di un oggetto.

IBM i *DefXmitQName (stringa di caratteri a 48 byte) su IBM i*

Il nome della coda di trasmissione predefinita.

Questo è il nome della coda di trasmissione utilizzata per la trasmissione dei messaggi ai gestori code remoti, se non vi è alcuna altra indicazione di quale coda di trasmissione utilizzare.

Se non esiste una coda di trasmissione predefinita, il nome è completamente vuoto. Il valore iniziale di questo attributo è vuoto.

Per determinare il valore di questo attributo, utilizzare il selettore CADXQN con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQN.

IBM i *DistLists (numero intero con segno a 10 cifre) su IBM i*

Supporto elenco di distribuzione.

Ciò indica se il gestore code locale supporta gli elenchi di distribuzione sulle chiamate MQPUT e MQPUT1 . Può avere uno dei seguenti valori:

DLSUPP

Elenchi di distribuzione supportati.

DLNSUP
Elenchi di distribuzione non supportati.

Per determinare il valore di questo attributo, utilizzare il selettore IADIST con la chiamata MQINQ.

IBM i *InhibitEvent (numero intero con segno a 10 cifre) su IBM i*

Controlla se vengono generati eventi inibisci (inibisci ricezione e inibisci immissione).

Può avere uno dei seguenti valori:

EVRDIS
Report eventi disabilitato.

EVRENA
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Monitoraggio e prestazioni.

Per determinare il valore di questo attributo, utilizzare il selettore IAINHE con la chiamata MQINQ.

IBM i *LocalEvent (numero intero con segno a 10 cifre) su IBM i*

Controlla se vengono generati eventi di errore locali.

Il valore è uno dei seguenti:

EVRDIS
Report eventi disabilitato.

EVRENA
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring

Per determinare il valore di questo attributo, utilizzare il selettore IALCLE con la chiamata MQINQ.

IBM i *LoggerEvent (numero intero con segno a 10 cifre) su IBM i*

Controlla se vengono generati eventi del logger di ripristino.

Può avere uno dei seguenti valori:

Abilitato
Vengono generati eventi del programma di registrazione.

DISABILITATO
Gli eventi del programma di registrazione non vengono generati. Questo è il valore predefinito iniziale dei gestori code.

Per ulteriori informazioni sugli eventi, consultare Monitoraggio e prestazioni.

IBM i *MaxHandles (numero intero con segno a 10 cifre) su IBM i*

Numero massimo di puntatori.

Questo è il numero massimo di handle aperti che qualsiasi attività può utilizzare contemporaneamente. Ogni chiamata MQOPEN riuscita per una singola coda (o per un oggetto che non è una coda) utilizza un handle. Tale handle diventa disponibile per il riutilizzo quando l'oggetto viene chiuso. Tuttavia, quando viene aperto un elenco di distribuzione, a ogni coda nell'elenco di distribuzione viene assegnato un handle separato e in modo che la chiamata MQOPEN utilizzi un numero di handle pari al numero di code presenti nell'elenco di distribuzione. Questo deve essere preso in considerazione quando si decide un valore adatto per *MaxHandles*.

La chiamata MQPUT1 esegue una chiamata MQOPEN come parte della sua elaborazione; di conseguenza, MQPUT1 utilizza un numero di handle pari a quello di MQOPEN, ma gli handle vengono utilizzati solo per la durata della chiamata MQPUT1 .

Il valore è compreso tra 1 e 999 999 999. Su IBM i, il valore predefinito è 256.

Per determinare il valore di questo attributo, utilizzare il selettore IAMHND con la chiamata MQINQ.

IBM i *MaxMsgLunghezza (numero intero con segno a 10 cifre) su IBM i*

Lunghezza massima del messaggio in byte.

È la lunghezza del messaggio *fisico* più lungo che può essere gestito dal gestore code. Tuttavia, poiché l'attributo gestore code **MaxMsgLength** può essere impostato indipendentemente dall'attributo coda **MaxMsgLength** , il messaggio fisico più lungo che può essere inserito in una coda è il minore di questi due valori.

Se il gestore code supporta la segmentazione, è possibile per un'applicazione inserire un messaggio *logico* più lungo del minore dei due attributi **MaxMsgLength** , ma solo se l'applicazione specifica l'indicatore MFSEGA in MQMD. Se viene specificato tale indicatore, il limite superiore per la lunghezza di un messaggio logico è 999 999 999 999 byte, ma generalmente, i vincoli di risorsa imposti dal sistema operativo o dall'ambiente in cui è in esecuzione l'applicazione, risulteranno in un limite inferiore.

Il limite inferiore per l'attributo **MaxMsgLength** è 32 KB (32 768 byte). Su IBM i, la lunghezza massima del messaggio è 100 MB (104 857 600 byte).

Per determinare il valore di questo attributo, utilizzare il selettore IAMLEN con la chiamata MQINQ.

IBM i *MaxPriority (numero intero con segno a 10 cifre) su IBM i*

Priorità massima.

Questa è la priorità massima dei messaggi supportata dal gestore code. Le priorità vanno da zero (più basso) a *MaxPriority* (più alto).

Per determinare il valore di questo attributo, utilizzare il selettore IAM PRI con la chiamata MQINQ.

IBM i *MaxUncommittedMessaggi (numero intero con segno a 10 cifre) su IBM i*

Numero massimo di messaggi non sottoposti a commit all'interno di un'unità di lavoro.

Questo è il numero massimo di messaggi di cui non è stato eseguito il commit che possono esistere all'interno di un'unità di lavoro. Il numero di messaggi di cui non è stato eseguito il commit è la somma dei seguenti dall'inizio dell'unità di lavoro corrente:

- Messaggi immessi dall'applicazione con l'opzione PMSYP
- Messaggi richiamati dall'applicazione con l'opzione GMSYP
- Messaggi trigger e messaggi di report COA generati dal gestore code per i messaggi inseriti con l'opzione PMSYP
- Messaggi di report COD generati dal gestore code per i messaggi richiamati con l'opzione GMSYP

I seguenti messaggi non vengono conteggiati come non sottoposti a commit:

- Messaggi immessi o richiamati dall'applicazione all'esterno di un'unità di lavoro
- Attivare i messaggi o i messaggi di report COA/COD generati dal gestore code come risultato di messaggi immessi o richiamati all'esterno di un'unità di lavoro
- Messaggi di report di scadenza generati dal gestore code (anche se la chiamata che causa il messaggio di report di scadenza ha specificato GMSYP)
- Messaggi di evento generati dal gestore code (anche se la chiamata che causa il messaggio di evento ha specificato PMSYP o GMSYP)

Nota:

1. I messaggi di report di eccezione vengono generati dall'agent MCA (Message Channel Agent) o dall'applicazione e vengono quindi trattati allo stesso modo dei messaggi ordinari immessi o richiamati dall'applicazione.
2. Quando un messaggio o un segmento viene inserito con l'opzione PMSYP, il numero di messaggi di cui non è stato eseguito il commit viene incrementato di uno indipendentemente dal numero di messaggi

fisici effettivamente risultanti dall'inserimento. (È possibile che si verifichi più di un messaggio fisico se il gestore code deve suddividere il messaggio o il segmento.)

3. Quando un elenco di distribuzione viene inserito con l'opzione PMSYP, il numero di messaggi di cui non è stato eseguito il commit viene incrementato di uno *per ciascun messaggio fisico generato*. Può essere piccolo come uno o grande come il numero di destinazioni nell'elenco di distribuzione.

Il limite inferiore per questo attributo è 1; il limite superiore è 999 999 999.

Per stabilire il valore di questo attributo, utilizzare il selettore IAMUNC con la chiamata MQINQ.

IBM i *PerformanceEvent (numero intero con segno a 10 cifre) il IBM i*

Controlla se vengono generati eventi relativi alle prestazioni.

PerformanceEvent può avere uno dei seguenti valori:

EVRDIS
Report eventi disabilitato.

EVRENA
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il programma di selezione IAPFME con la chiamata MQINQ.

IBM i *Piattaforma (numero intero con segno a 10 cifre) su IBM i*

Piattaforma su cui è in esecuzione il gestore code.

Indica il sistema operativo su cui è in esecuzione il gestore code. Il valore è:

PL400
IBM i.

IBM i *Modalità PubSub(numero intero con segno a 10 cifre) su IBM i*

Se il motore di pubblicazione / sottoscrizione e l'interfaccia di pubblicazione / sottoscrizione accodata sono in esecuzione, consentendo quindi alle applicazioni di pubblicare / sottoscrivere utilizzando l'interfaccia di programmazione dell'applicazione e le code monitorate dall'interfaccia di pubblicazione / sottoscrizione accodata.

Può avere uno dei seguenti valori:

PSMCP
Il motore di pubblicazione/sottoscrizione è in esecuzione. È quindi possibile pubblicare / sottoscrivere utilizzando l'API (application programming interface). L'interfaccia di pubblicazione / sottoscrizione accodata non è in esecuzione, pertanto qualsiasi messaggio inserito nelle code monitorate dall'interfaccia di pubblicazione / sottoscrizione accodata non viene utilizzato. Questa impostazione viene utilizzata per compatibilità con WebSphere Message Broker V6 o versioni precedenti utilizzando questo gestore code, poiché deve leggere le stesse code da cui l'interfaccia di pubblicazione / sottoscrizione accodata normalmente legge.

PSMDS
Il motore di pubblicazione/sottoscrizione e l'interfaccia di pubblicazione/sottoscrizione in coda non sono in esecuzione. Non è quindi possibile pubblicare / sottoscrivere utilizzando l'API (application programming interface). I messaggi di pubblicazione / sottoscrizione inseriti nelle code monitorate dall'interfaccia di pubblicazione / sottoscrizione accodata non vengono utilizzati.

PSMEN
Il motore di pubblicazione/sottoscrizione e l'interfaccia di pubblicazione/sottoscrizione in coda sono in esecuzione. È quindi possibile pubblicare / sottoscrivere utilizzando l'API (application programming interface) e le code monitorate dall'interfaccia di pubblicazione / sottoscrizione accodata. Questo è il valore predefinito iniziale del gestore code.

Per stabilire il valore di questo attributo, utilizzare il selettore PSMODE con la chiamata MQINQ.

IBM i *QMgrDesc (stringa di caratteri a 64 byte) su IBM i*

La descrizione del gestore code.

Questo è un campo che può essere utilizzato per commenti descrittivi. Il contenuto del campo non ha alcun significato per il gestore code, ma il gestore code potrebbe richiedere che il campo contenga solo caratteri che possono essere visualizzati. Non può contenere caratteri null; se necessario, viene riempito a destra con spazi. In un'installazione DBCS, questo campo può contenere caratteri DBCS (con una lunghezza massima di 64 byte).

Nota: Se questo campo contiene caratteri non presenti nella serie di caratteri del gestore code (come definito dall'attributo del gestore code **CodedCharSetId**), tali caratteri potrebbero essere tradotti in modo non corretto se questo campo viene inviato a un altro gestore code.

Su IBM i, il valore predefinito è vuoto.

Per determinare il valore di questo attributo, utilizzare il selettore CAQMD con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQMD.

IBM i *QMgrIdentifier (stringa di caratteri a 48 - byte) su IBM i*

Identificativo univoco generato internamente del gestore code.

Questo è un nome univoco generato internamente per il gestore code.

Per determinare il valore di questo attributo, utilizzare il selettore CAQMID con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQMID.

IBM i *QMgrName (stringa di caratteri a 48 byte) su IBM i*

È il nome del gestore code.

Questo è il nome del gestore code locale, vale a dire il nome del gestore code a cui l'applicazione è collegata.

I primi 12 caratteri del nome vengono utilizzati per creare un identificativo di messaggio univoco (consultare il campo *MDMID* descritto in “MQMD (Message Descriptor) su IBM i” a pagina 1140). I gestori code che possono comunicare tra loro devono quindi avere nomi diversi nei primi 12 caratteri, in modo che gli ID messaggio siano univoci nella rete del gestore code.

Per determinare il valore di questo attributo, utilizzare il selettore CAQMN con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQMN.

IBM i *RemoteEvent (numero intero con segno a 10 cifre) su IBM i*

Controlla se vengono generati eventi di errore remoti.

Il valore è uno dei seguenti:

EVRDIS
Report eventi disabilitato.

EVRENA
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore IARMTE con la chiamata MQINQ.

IBM i *RepositoryName (stringa di caratteri a 48 byte) su IBM i*

Il nome del cluster per cui questo gestore code fornisce i servizi del repository.

Questo è il nome di un cluster per il quale questo gestore code fornisce un servizio gestore repository. Se il gestore code fornisce questo servizio per più di un cluster, *RepositoryNamelist* specifica il nome di un oggetto elenco nomi che identifica i cluster e *RepositoryName* è vuoto. Almeno uno tra *RepositoryName* e *RepositoryNamelist* deve essere vuoto.

Per determinare il valore di questo attributo, utilizzare il selettore CARPN con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNQMN.

IBM i *RepositoryNamelist (stringa di caratteri a 48 byte) su IBM i*

Il nome dell'oggetto elenco nomi che contiene i nomi dei cluster per cui questo gestore code fornisce i servizi repository.

Questo è il nome di un oggetto elenco nomi che contiene i nomi dei cluster per cui questo gestore code fornisce un servizio gestore repository. Se il gestore code fornisce questo servizio solo per un cluster, l'oggetto elenco nomi contiene un unico nome. In alternativa, *RepositoryName* può essere utilizzato per indicare il nome del cluster, nel qual caso *RepositoryNamelist* è vuoto. Almeno uno tra *RepositoryName* e *RepositoryNamelist* deve essere vuoto.

Per determinare il valore di questo attributo, utilizzare il selettore CARPNL con la chiamata MQINQ. La lunghezza di questo attributo è fornita da LNNLN.

IBM i *SSLEvent (stringa di caratteri) su IBM i*

Determina se vengono generati eventi TLS.

Il valore è uno dei seguenti:

- EVRENA (MQINQ/PCF/config event) ENABLED (MQSC): vengono generati eventi TLS (ovvero, viene generato l'evento RC2371).
- EVRDIS (MQINQ/PCF/config event) DISABLED (MQSC): gli eventi TLS non vengono generati. Questo è il valore predefinito iniziale del gestore code.

Per determinare il valore di questo attributo, utilizzare il selettore IASSLE con la chiamata MQINQ.

IBM i *SSLKeyResetConteggio (intero) il IBM i*

Determina il numero totale di byte non codificati inviati e ricevuti all'interno di una conversazione TLS, prima che la chiave segreta venga rinegoziata. Il numero di byte include le informazioni di controllo inviate da MCA (message channel agent).

Questo valore viene utilizzato solo dagli MCA del canale TLS che avviano le comunicazioni da questo gestore code (ossia, l'MCA del canale mittente in un accoppiamento di canali mittente e destinatario).

Se il valore di questo attributo è maggiore di 0 e gli heartbeat del canale sono abilitati per un canale, anche la chiave segreta viene rinegoziata prima che i dati vengano inviati o ricevuti dopo un heartbeat del canale. Il numero di byte fino a quando non viene reimpostata la successiva rinegoziazione della chiave segreta dopo ogni rinegoziazione riuscita.

Il valore può essere compreso tra 0 e 999 999 999. Il valore 0 per questo attributo indica che la chiave segreta non viene mai rinegoziata. Se si specifica un conteggio di reimpostazione della chiave segreta TLS compreso tra 1 byte e 32 KB, i canali TLS utilizzeranno un conteggio di reimpostazione della chiave segreta di 32 KB. Ciò consente di evitare il costo di elaborazione di un numero eccessivo di reimpostazioni della chiave che si verificherebbe per valori di reimpostazione della chiave segreta TLS di piccole dimensioni.

Quando il server SSL è un gestore code IBM MQ e sono abilitati sia la reimpostazione della chiave segreta che gli heartbeat del canale, la rinegoziazione si verifica immediatamente dopo ogni heartbeat del canale.

Per stabilire il valore di questo attributo, utilizzare il selettore IASSRC con la chiamata MQINQ.

IBM i *StartStopEvento (numero intero con segno a 10 cifre) il IBM i*

Controlla se vengono generati eventi di avvio e arresto.

Questo attributo può avere uno dei seguenti valori:

EVRDIS
Report eventi disabilitato.

EVRENA
Segnalazione eventi abilitata.

Per ulteriori informazioni sugli eventi, consultare Event monitoring.

Per determinare il valore di questo attributo, utilizzare il selettore IASSE con la chiamata MQINQ.

IBM i *SyncPoint (numero intero con segno a 10 cifre) su IBM i*

Disponibilità del punto di sincronizzazione.

Ciò indica se il gestore code locale supporta le unità di lavoro e di sincronizzazione con le chiamate MQGET, MQPUT e MQPUT1 .

SAVL
Unità di lavoro e punto di sincronizzazione disponibile.

SPNAVL
Unità di lavoro e punto di sincronizzazione non disponibile.

Per determinare il valore di questo attributo, utilizzare il selettore IASYNC con la chiamata MQINQ.

IBM i *TraceRouteRegistrazione (numero intero con segno a 10 cifre) su IBM i*

Controlla se le informazioni sui messaggi vengono registrate durante il loro flusso attraverso un gestore code.

Il valore è uno dei seguenti:

- RECDD: non è consentito accodare i messaggi di instradamento alla traccia
- RECDQ: i messaggi vengono inseriti in una coda denominata fissa
- RECDM: determina l'utilizzo del messaggio (questa è l'impostazione predefinita iniziale)

Per evitare che il messaggio di instradamento della traccia rimanga nel sistema, impostare un valore di scadenza maggiore di zero e specificare l'opzione del prospetto RODISC. Per evitare che i messaggi di report o di risposta rimangano nel sistema, impostare l'opzione di report ROPDAE. Per ulteriori informazioni, consultare “Opzioni di report e indicatori di messaggi su IBM i” a pagina 1473.

Per determinare il valore di questo attributo, utilizzare il selettore IATRGI con la chiamata MQINQ.

IBM i *TreeLifeOra (numero intero con segno a 10 cifre) su IBM i*

La durata, in secondi, degli argomenti non amministrativi.

Gli argomenti non di gestione sono quelli creati quando un'applicazione pubblica o sottoscrive una stringa di argomenti che non esiste come nodo di gestione. quando questo nodo non amministrativo non ha più alcuna sottoscrizione attiva, il presente parametro determina per quanto tempo il gestore code attenderà prima di rimuovere il nodo in questione. Solo gli argomenti non amministrativi in uso da parte di una sottoscrizione permanente persistono a seguito del riciclo del gestore code.

Specificare un valore fra 0 e 604 000. Un valore uguale a 0 indica che gli argomenti non amministrativi non vengono rimossi dal gestore code. Il valore predefinito iniziale del gestore code è 1800.

Per determinare il valore di questo attributo, utilizzare il selettore IATRLFT con la chiamata MQINQ.

IBM i *TriggerInterval (numero intero con segno a 10 cifre) su IBM i*

Intervallo messaggio trigger.

Questo è un intervallo di tempo (in millisecondi) utilizzato per limitare il numero di messaggi trigger. Ciò è rilevante solo quando *TriggerType* è TTFRST. In questo caso, i messaggi di trigger vengono normalmente generati solo quando un messaggio adatto arriva sulla coda e la coda era precedentemente vuota. In determinate circostanze, tuttavia, è possibile generare un ulteriore messaggio di trigger con l'attivazione TTFRST anche se la coda non era vuota. Questi messaggi di trigger aggiuntivi non vengono generati più spesso di ogni *TriggerInterval* millisecondi.

Per ulteriori informazioni sull'attivazione, vedi Trigger dei canali.

Il valore è compreso tra zero e 999 999 999. Il valore predefinito è 999 999 999.

Per determinare il valore di questo attributo, utilizzare il selettore IATRGI con la chiamata MQINQ.

Applicazioni

Queste informazioni descrivono i programmi di esempio forniti con IBM MQ for IBM i per RPG. Inoltre, scopri come creare applicazioni eseguibili dai programmi che scrivi.

Creazione dell'applicazione

Le pubblicazioni IBM i descrivono come creare applicazioni eseguibili dai programmi scritti. Questo argomento descrive le attività aggiuntive e le modifiche alle attività standard, che è necessario eseguire quando si creano applicazioni IBM MQ for IBM i da eseguire in IBM i.

Oltre a codificare le chiamate MQI nel proprio codice di origine, è necessario aggiungere le istruzioni di linguaggio appropriate per includere i file di copia IBM MQ for IBM i per il linguaggio RPG. È necessario acquisire familiarità con il contenuto di questi file; i loro nomi e una breve descrizione del contenuto sono forniti nel testo seguente.

IBM i *IBM MQ copia file su IBM i*

IBM MQ for IBM i fornisce i file di copia per assistere l'utente nella scrittura delle applicazioni nel linguaggio di programmazione RPG. Sono adatti per l'utilizzo con WebSphere Development toolset (5722 WDS) ILE RPG 4 Compiler.

I file di copia forniti da IBM MQ for IBM i per assistere nella scrittura delle uscite dei canali sono descritti in Programmi di uscita dei canali per i canali di messaggistica.

I nomi dei file di copia IBM MQ for IBM i per RPG hanno il prefisso CMQ. Hanno un suffisso di G o H. Esistono file di copia separati contenenti le costanti denominate e un file per ognuna delle strutture. I file di copia sono elencati in “Considerazioni sulla lingua” a pagina 1036.

Nota: Per ILE RPG/400, vengono forniti come membri del fileQRPGLESRC nella libreria QMQM.

Le dichiarazioni di struttura non contengono istruzioni DS . Ciò consente all'applicazione di dichiarare una struttura dati (o una struttura dati a più ricorrenze) codificando l'istruzione DS e utilizzando l'istruzione / COPY da copiare nel resto della dichiarazione:

Per ILE RPG/400 l'istruzione è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7
D* Declare an MQMD data structure
D MQMD       DS
D/COPY CMQMDG
```

Preparazione dei programmi da eseguire

Per creare un'applicazione IBM MQ for IBM i eseguibile, è necessario compilare il codice sorgente scritto.

Per fare ciò per ILE RPG/400, è possibile utilizzare i tipici comandi IBM i , CRTRPGMOD e CRTPGM.

Dopo aver creato il *MODULE, è necessario specificare `BNDSRVPGM(QMQM/LIBMQM)` nel comando CRTPGM. Ciò include le diverse procedure IBM MQ nel programma.

Assicurarsi che la libreria contenente i file di copia (QMQM) si trovi nell'elenco librerie quando si esegue la compilazione.

Per ulteriori informazioni relative alle considerazioni sulla programmazione, incluse le modalità client, consultare “Considerazioni sulla lingua” a pagina 1036.

Interfacce con il gestore del punto di sincronizzazione esterno IBM i

IBM MQ for IBM i utilizza il controllo del commit IBM i nativo come coordinatore del punto di sincronizzazione esterno.

Consultare *IBM i Programming: Backup and Recovery Guide* per ulteriori informazioni sulle funzionalità di controllo del commit di IBM i.

Per avviare le funzioni di controllo del commit IBM i , utilizzare il comando del sistema STRCMTCTL. Per terminare il controllo di sincronia, utilizzare il comando di sistema ENDCMTCTL.

Nota: Il valore predefinito di *Ambito definizione di commit* è *ACTGRP. Deve essere definito come *JOB per IBM MQ per IBM i. Ad esempio:

```
STRCMTCTL LCKLVL(*ALL) CMTSCOPE(*JOB)
```

Se si richiama MQPUT, MQPUT1o MQGET, specificando PMSYP o GMSYP, dopo aver avviato il controllo del commit, IBM MQ for IBM i si aggiunge come una risorsa di commit API alla definizione di commit. Questa è in genere la prima chiamata di questo tipo in un lavoro. Mentre ci sono delle risorse di commit API registrate in una particolare definizione di commit, non è possibile terminare il controllo di commit per tale definizione.

IBM MQ for IBM i rimuove la relativa registrazione come una risorsa di commit API quando ci si disconnette dal gestore code, purché non vi siano operazioni MQI in sospeso nell'unità di lavoro corrente.

Se ci si disconnette dal gestore code mentre ci sono operazioni MQPUT, MQPUT1o MQGET in sospeso nell'unità di lavoro corrente, IBM MQ for IBM i rimane registrato come una risorsa di commit API in modo che venga notificato il successivo commit o rollback. Quando il punto di sincronizzazione successivo viene raggiunto, IBM MQ esegue il commit o il rollback delle modifiche come richiesto. È possibile per un'applicazione disconnettere e riconnettersi a un gestore code durante un'unità di lavoro attiva ed eseguire ulteriori operazioni MQGET e MQPUT all'interno della stessa unità di lavoro (questa è una disconnessione in sospeso).

Se si tenta di immettere un comando di sistema ENDCMTCTL per tale definizione di commit, viene emesso il messaggio CPF8355 , che indica che le modifiche in sospeso erano attive. Questo messaggio viene visualizzato anche nella registrazione lavoro al termine del lavoro. Per evitare ciò, assicurarsi di eseguire il commit o il rollback di tutte le operazioni IBM MQ in sospeso e di disconnettersi dal gestore code. Pertanto, l'utilizzo dei comandi COMMIT o ROLLBACK prima di ENDCMTCTL dovrebbe consentire il corretto completamento del controllo di commit finale.

Quando il controllo di commit IBM i viene utilizzato come un coordinatore del punto di sincronizzazione esterno, le chiamate MQCMIT, MQBACK e MQBEGIN potrebbero non essere emesse. Le chiamate a tali funzioni non riescono con il codice di errore RC2012.

Per eseguire il commit o il rollback (ovvero, per eseguire il backout) della propria unità di lavoro, utilizzare uno dei linguaggi di programmazione che supporta il controllo del commit. Ad esempio:

- Comandi CL: COMMIT e ROLLBACK
- Funzioni di programmazione ILE C: _Rcommit e _Rrollback
- RPG/400: COMMIT e ROLBK
- COBOL/400: COMMIT e ROLLBACK

Syncpoint in applicazioni CICS per IBM i

IBM MQ for IBM i partecipa alle unità di lavoro con CICS. È possibile utilizzare MQI in un'applicazione CICS per inserire e richiamare messaggi all'interno dell'unità di lavoro corrente.

È possibile utilizzare il comando EXEC CICS SYNCPOINT per stabilire un punto di sincronizzazione che includa operazioni IBM MQ for IBM i . Per ripristinare tutte le modifiche fino al punto di sincronizzazione precedente, è possibile utilizzare il comando EXEC CICS SYNCPOINT ROLLBACK.

Se si utilizza MQPUT, MQPUT1o MQGET con l'opzione PMSYP o GMSYP, impostata in un'applicazione CICS , non è possibile scollegarsi CICS fino a quando IBM MQ for IBM i non ha rimosso la registrazione come risorsa di commit API. Pertanto, è necessario eseguire il commit o il backout di tutte le operazioni put o get in sospeso prima di disconnettersi dal gestore code. Ciò consentirà di scollegare CICS.

Programmi di esempio su IBM i

Questo argomento descrive i programmi di esempio forniti con IBM MQ for IBM i per RPG. Gli esempi mostrano gli utilizzi tipici di MQI (Message Queue Interface).

Gli esempi non sono destinati a dimostrare tecniche di programmazione generali, quindi è stato omesso il controllo degli errori che si potrebbe voler includere in un programma di produzione. Tuttavia, questi esempi sono adatti per essere utilizzati come base per i propri programmi di accodamento messaggi.

Il codice sorgente per tutti gli esempi viene fornito con il prodotto; questa sorgente include commenti che spiegano le tecniche di accodamento dei messaggi dimostrate nei programmi.

Esiste una serie di programmi di esempio ILE:

1. **Programmi che utilizzano chiamate con prototipo a MQI (chiamate associate statiche)**

   L'origine esiste in QMQMSAMP/QRPGLESRC. I membri sono denominati AMQ3xxx4, dove xxx indica la funzione di esempio. I membri di copia esistono in QMQM/QRPGLESRC. Ogni nome membro ha un suffisso G o H.

Tabella 811 a pagina 1456 fornisce un elenco completo dei programmi di esempio forniti con IBM MQ for IBM ie mostra i nomi dei programmi in ogni linguaggio di programmazione supportato. Si noti che i loro nomi iniziano tutti con il prefisso AMQ, il quarto carattere nel nome indica il linguaggio di programmazione.

Tabella 811. Nomi dei programmi di esempio

	RPG (ILE)
Esempi di inserimento	AMQ3PUT4
Sfoglia esempi	AMQ3GBR4
Ottieni esempi	AMQ3GET4
Richiedi esempi	AMQ3REQ4
Esempi Echo	AMQ3ECH4
Interroga esempi	AMQ3INQ4
Imposta esempi	AMQ3SET4
Esempio di controllo trigger	AMQ3TRG4
Esempio di server trigger	AMQ3SRV4

Oltre a questi, l'opzione di esempio IBM MQ for IBM i include un file di dati di esempio, AMQSDATA, che può essere utilizzato come input per alcuni programmi di esempio e programmi CL di esempio che dimostrano le attività di amministrazione. Gli esempi CL sono descritti in Amministrazione IBM i . È possibile utilizzare il programma di esempio CL per creare code da utilizzare con i programmi di esempio descritti in questo argomento.

Per informazioni su come eseguire i programmi di esempio, vedere “Preparazione ed esecuzione dei programmi di esempio su IBM i .” a pagina 1457.

Funzioni illustrate nei programmi di esempio su IBM i

Una tabella che mostra le tecniche dimostrate dai programmi di esempio IBM MQ for IBM i .

Alcune tecniche si verificano in più di un programma di esempio, ma solo un programma è elencato nella tabella. Tutti gli esempi aprono e chiudono le code utilizzando le chiamate MQOPEN e MQCLOSE, quindi queste tecniche non sono elencate separatamente nella tabella.

Tabella 812. Programmi di esempio che dimostrano l'utilizzo di MQI	
Tecnica	**RPG (ILE)**
Utilizzo delle chiamate MQCONN e MQDISC	AMQ3ECH4 o AMQ3INQ4
Connessione e disconnessione implicite	AMQ3PUT4
Inserimento di messaggi utilizzando la chiamata MQPUT	AMQ3PUT4
Inserimento di un singolo messaggio utilizzando la chiamata MQPUT1	AMQ3ECH4 o AMQ3INQ4
Risposta a un messaggio di richiesta	AMQ3INQ4
Ricezione messaggi (nessuna attesa)	AMQ3GBR4
Ricezione dei messaggi (attendere con un limite di tempo)	AMQ3GET4
Ricezione di messaggi (con conversione dati)	AMQ3ECH4
Esplorazione di una coda	AMQ3GBR4
Utilizzo di una coda di input condivisa	AMQ3INQ4
Utilizzo di una coda di immissione esclusiva	AMQ3REQ4
Utilizzo della chiamata MQINQ	AMQ3INQ4
Utilizzo della chiamata MQSET	AMQ3SET4
Utilizzo di una coda di risposta	AMQ3REQ4
Richiesta di messaggi di eccezione	AMQ3REQ4
Accettazione di un messaggio troncato	AMQ3GBR4
Utilizzo di un nome coda risolto	AMQ3GBR4
Elaborazione trigger	AMQ3SRV4 o AMQ3TRG4

Nota: Tutti i programmi di esempio producono un file di spool che contiene i risultati dell'elaborazione.

Preparazione ed esecuzione dei programmi di esempio su IBM i .

Prima di poter eseguire i programmi di esempio IBM MQ for IBM i , è necessario compilarli come qualsiasi altra applicazione IBM MQ for IBM i . A tale scopo, è possibile utilizzare i comandi IBM i CRTRPGMOD e CRTPGM.

Quando si creano i programmi AMQ3xxx4 , è necessario specificare BNDSRVPGM (QMQM/LIBMQM) nel comando CRTPGM. Questa operazione include le varie procedure IBM MQ nel programma.

I programmi di esempio vengono forniti nella libreria QMQMSAMP come membri di QRPGLESRC. Essi utilizzano i file di copia forniti nella libreria QMQM, quindi verificare che questa libreria si trovi nell'elenco librerie quando vengono compilati. Il compilatore RPG fornisce messaggi informativi poiché gli esempi non utilizzano molte delle variabili dichiarate nei file di copia.

Esecuzione dei programmi di esempio

È possibile utilizzare le proprie code quando si eseguono gli esempi oppure è possibile compilare ed eseguire AMQSAMP4 per creare alcune code di esempio. L'origine per questo programma viene fornita nel file QCLSRC nella libreria QMQMSAMP. Può essere compilato utilizzando il comando CRTCLPGM.

Per richiamare uno dei programmi di esempio, utilizzare un comando come:

```
CALL PGM(QMQMSAMP/AMQ3PUT4) PARM('Queue_Name','Queue_Manager_Name')
```

Dove Queue_Name e Queue_Manager_Name devono avere una lunghezza di 48 caratteri, che si ottiene riempiendo Queue_Name e Queue_Manager_Name con il numero richiesto di spazi.

Per i programmi di esempio Inquire e Set, le definizioni del campione create da AMQSAMP4 causano l'attivazione delle versioni C di questi campioni. Se si desidera attivare le versioni RPG, è necessario modificare le definizioni di processo SYSTEM.SAMPLE.ECHOPROCESS e SYSTEM.SAMPLE.INQPROCESS e SYSTEM.SAMPLE.SETPROCESS. È possibile utilizzare il comando CHGMQMPRC (descritto in Modifica processo MQ (CHGMQMPRC)) per eseguire tale operazione o modificare ed eseguire AMQSAMP4 con la definizione alternativa.

Il programma di esempio Put su IBM i

Il programma di esempio Put, AMQ3PUT4, inserisce i messaggi su una coda utilizzando la chiamata MQPUT.

Per avviare il programma, richiamare il programma e fornire il nome della coda di destinazione come parametro del programma. Il programma inserisce una serie di messaggi fissi sulla coda; questi messaggi vengono presi dal blocco di dati alla fine del codice sorgente del programma. Un programma di inserimento di esempio è AMQ3PUT4 nella libreria QMQMSAMP.

Utilizzando questo programma di esempio, il comando è:

```
CALL PGM(QMQMSAMP/AMQ3PUT4) PARM('Queue_Name','Queue_Manager_Name')
```

Dove Queue_Name e Queue_Manager_Name devono avere una lunghezza di 48 caratteri, che si ottiene riempiendo Queue_Name e Queue_Manager_Name con il numero richiesto di spazi.

Progettazione del programma di esempio Put

Il programma utilizza la chiamata MQOPEN con l'opzione OOOUT per aprire la coda di destinazione per inserire i messaggi. I risultati vengono emessi in un file di spool. Se non riesce ad aprire la coda, il programma scrive un messaggio di errore contenente il codice di errore restituito dalla chiamata MQOPEN. Per semplificare il programma, in questa e nelle successive chiamate MQI, il programma utilizza i valori predefiniti per molte opzioni.

Per ogni riga di dati contenuta nel codice di origine, il programma legge il testo in un buffer e utilizza la chiamata MQPUT per creare un messaggio datagramma contenente il testo di tale riga. Il programma continua fino a quando non raggiunge la fine dell'input o la chiamata MQPUT ha esito negativo. Se il programma raggiunge la fine dell'input, chiude la coda utilizzando la chiamata MQCLOSE.

Il programma di esempio Sfoglia su IBM i

Il programma di esempio Sfoglia, AMQ3GBR4, sfoglia i messaggi su una coda utilizzando la chiamata MQGET.

Il programma richiama le copie di tutti i messaggi nella coda specificata quando si richiama il programma; i messaggi rimangono nella coda. È possibile utilizzare la coda fornita SYSTEM.SAMPLE.LOCAL; eseguire prima il programma di esempio Put per inserire alcuni messaggi nella coda. È possibile utilizzare la coda SYSTEM.SAMPLE.ALIAS, che è un nome alias per la stessa coda locale. Il programma continua fino a quando non raggiunge la fine della coda o una chiamata MQI non riesce.

Un esempio di comando per richiamare il programma RPG è:

```
CALL PGM(QMQMSAMP/AMQ3GBR4) PARM('Queue_Name','Queue_Manager_Name')
```

Dove Queue_Name e Queue_Manager_Name devono avere una lunghezza di 48 caratteri, che si ottiene riempiendo Queue_Name e Queue_Manager_Name con il numero richiesto di spazi. Pertanto, se si utilizza SYSTEM.SAMPLE.LOCAL come coda di destinazione, sono necessari 29 caratteri vuoti.

Progettazione del programma di esempio Sfoglia

Il programma apre la coda di destinazione utilizzando la chiamata MQOPEN con opzione OOBRW. Se non è possibile aprire la coda, il programma scrive un messaggio di errore nel file di spool, contenente il codice motivo restituito dalla chiamata MQOPEN.

Per ogni messaggio sulla coda, il programma utilizza la chiamata MQGET per copiare il messaggio dalla coda, quindi visualizza i dati contenuti nel messaggio. La chiamata MQGET utilizza queste opzioni:

GMBRWN
Dopo la chiamata MQOPEN, il cursore di esplorazione viene posizionato in modo logico prima del primo messaggio nella coda, quindi questa opzione restituisce il *primo* messaggio quando la chiamata viene eseguita per la prima volta.

GMNWT
Il programma non attende se non ci sono messaggi nella coda.

GMATM
La chiamata MQGET specifica un buffer di dimensione fissa. Se un messaggio è più lungo di questo buffer, il programma visualizza il messaggio troncato, insieme ad un'avvertenza che il messaggio è stato troncato.

Il programma dimostra in che modo è necessario cancellare i campi *MDMID* e *MDCID* della struttura MQMD dopo ogni chiamata MQGET poiché la chiamata imposta questi campi sui valori contenuti nel messaggio richiamato. La cancellazione di questi campi significa che le chiamate MQGET successive richiamano i messaggi nell'ordine in cui sono conservati nella coda.

Il programma continua fino alla fine della coda; qui, la chiamata MQGET restituisce il codice motivo RC2033 (nessun messaggio disponibile) e il programma visualizza un messaggio di avviso. Se la chiamata MQGET non riesce, il programma scrive un messaggio di errore che contiene il codice di errore nel file di spool.

Il programma chiude quindi la coda utilizzando la chiamata MQCLOSE.

Il programma di esempio Get su IBM i

Il programma di esempio Get, AMQ3GET4, riceve i messaggi da una coda utilizzando la chiamata MQGET.

Quando il programma viene richiamato, rimuove i messaggi dalla coda specificata. È possibile utilizzare la coda fornita SYSTEM.SAMPLE.LOCAL; eseguire prima il programma di esempio Put per inserire alcuni messaggi nella coda. È possibile utilizzare SYSTEM.SAMPLE.ALIAS , che è un nome alias per la stessa coda locale. Il programma continua fino a quando la coda non è vuota o una chiamata MQI non riesce.

Un esempio di comando per richiamare il programma RPG è:

```
CALL PGM(QMQMSAMP/AMQ3GET4) PARM('Queue_Name','Queue_Manager_Name')
```

dove `Queue_Name` e `Queue_Manager_Name` devono avere una lunghezza di 48 caratteri, che si ottiene riempiendo `Queue_Name` e `Queue_Manager_Name` con il numero richiesto di spazi. Pertanto, se si utilizza SYSTEM.SAMPLE.LOCAL come coda di destinazione, sono necessari 29 caratteri vuoti.

Progettazione del programma di esempio Get

Il programma apre la coda di destinazione per richiamare i messaggi; utilizza la chiamata MQOPEN con l'opzione OOINPQ. Se non è possibile aprire la coda, il programma scrive un messaggio di errore contenente il codice motivo restituito dalla chiamata MQOPEN nel file di spool.

Per ogni messaggio sulla coda, il programma utilizza una chiamata MQGET per rimuovere il messaggio dalla coda; visualizza quindi i dati contenuti nel messaggio. La chiamata MQGET utilizza l'opzione GMWT, specificando un intervallo di attesa (*GMWI*) di 15 secondi, in modo che il programma attenda per questo periodo se non è presente alcun messaggio nella coda. Se non arriva alcun messaggio prima della scadenza di questo intervallo, la chiamata ha esito negativo e restituisce il codice di errore RC2033 (nessun messaggio disponibile).

Il programma dimostra in che modo è necessario cancellare i campi *MDMID* e *MDCID* della struttura MQMD dopo ogni chiamata MQGET poiché la chiamata imposta questi campi sui valori contenuti nel messaggio richiamato. La cancellazione di questi campi significa che le chiamate MQGET successive richiamano i messaggi nell'ordine in cui sono conservati nella coda.

La chiamata MQGET specifica un buffer di dimensione fissa. Se un messaggio è più lungo di questo buffer, la chiamata non riesce e il programma si arresta.

Il programma continua fino a quando la chiamata MQGET non restituisce il codice motivo RC2033 (nessun messaggio disponibile) o la chiamata MQGET non riesce. Se la chiamata ha esito negativo, il programma visualizza un messaggio di errore che contiene il codice di errore.

Il programma chiude quindi la coda utilizzando la chiamata MQCLOSE.

Il programma di esempio Request su IBM i

Il programma di esempio di richiesta, AMQ3REQ4, illustra l'elaborazione client/server. L'esempio è il client che inserisce i messaggi di richieste in una coda elaborata da un programma server. Attende che il programma del server inserisci un messaggio di risposta in una coda di risposta.

L'esempio Richiesta inserisce una serie di messaggi di richiesta su una coda utilizzando la chiamata MQPUT. Questi messaggi specificano SYSTEM.SAMPLE.REPLY come coda di risposta. Il programma attende i messaggi di risposta, quindi li visualizza. Le risposte vengono inviate solo se la coda di destinazione (che chiameremo *coda server*) viene elaborato da un'applicazione server o se un'applicazione viene attivata per tale scopo (i programmi di esempio Inquire e Set sono progettati per essere attivati). L'esempio attende 5 minuti per l'arrivo della prima risposta (per consentire l'attivazione di un'applicazione server) e 15 secondi per le risposte successive, ma può terminare senza ottenere alcuna risposta.

Per avviare il programma, richiamare il programma e fornire il nome della coda di destinazione come parametro del programma. Il programma inserisce una serie di messaggi fissi sulla coda; questi messaggi vengono presi dal blocco di dati alla fine del codice sorgente del programma.

Progettazione del programma di esempio Richiesta

Il programma apre la coda del server in modo che possa inserire i messaggi. Utilizza la chiamata MQOPEN con l'opzione OOOUT. Se non è in grado di aprire la coda, il programma visualizza un messaggio di errore contenente il codice motivo restituito dalla chiamata MQOPEN.

Il programma apre quindi la coda di risposta denominata SYSTEM.SAMPLE.REPLY in modo che possa ricevere i messaggi di risposta. Per questo, il programma utilizza la chiamata MQOPEN con l'opzione OOINPX. Se non è in grado di aprire la coda, il programma visualizza un messaggio di errore contenente il codice motivo restituito dalla chiamata MQOPEN.

Per ogni riga di input, il programma legge il testo in un buffer e utilizza la chiamata MQPUT per creare un messaggio di richiesta contenente il testo di quella riga. In questa chiamata, il programma utilizza l'opzione di prospetto ROEXCD per richiedere che tutti i messaggi di prospetto inviati sul messaggio di richiesta includano i primi 100 byte dei dati del messaggio. Il programma continua fino a quando non raggiunge la fine dell'input o la chiamata MQPUT ha esito negativo.

Il programma utilizza quindi la chiamata MQGET per rimuovere i messaggi di risposta dalla coda e visualizza i dati contenuti nelle risposte. La chiamata MQGET usa l'opzione GMWT, specificando un intervallo di attesa (*GMWI*) di 5 minuti per la prima risposta (per consentire l'attivazione di un'applicazione server) e 15 secondi per le risposte successive. Il programma attende questi periodi se non è presente alcun messaggio sulla coda. Se non arriva alcun messaggio prima della scadenza di questo intervallo, la chiamata ha esito negativo e restituisce il codice di errore RC2033 (nessun messaggio disponibile). La chiamata utilizza anche l'opzione GMATM, quindi i messaggi più lunghi della dimensione del buffer dichiarata vengono troncati.

Il programma dimostra in che modo è necessario cancellare i campi *MDMID* e *MDCOD* della struttura MQMD dopo ogni chiamata MQGET poiché la chiamata imposta questi campi sui valori contenuti nel messaggio richiamato. La cancellazione di questi campi significa che le chiamate MQGET successive richiamano i messaggi nell'ordine in cui sono conservati nella coda.

Il programma continua fino a quando la chiamata MQGET non restituisce il codice motivo RC2033 (nessun messaggio disponibile) o la chiamata MQGET non riesce. Se la chiamata ha esito negativo, il programma visualizza un messaggio di errore che contiene il codice di errore.

Il programma chiude quindi sia la coda del server che la coda di risposta utilizzando la chiamata MQCLOSE. La Tabella 813 a pagina 1461 mostra le modifiche al programma di esempio Echo necessarie per eseguire i programmi di esempio Inquire e Set.

Nota: I dettagli per il programma di esempio Echo sono inclusi come riferimento.

Tabella 813. Dettagli del programma di esempio Client / Server

Nome programma	**Coda SYSTEM/SAMPLE**	**Programma avviato**
Echo	Echo	AMQ3ECH4
Richiedi	INQ	AMQ3INQ4
Imposta	SET	AMQ3SET4

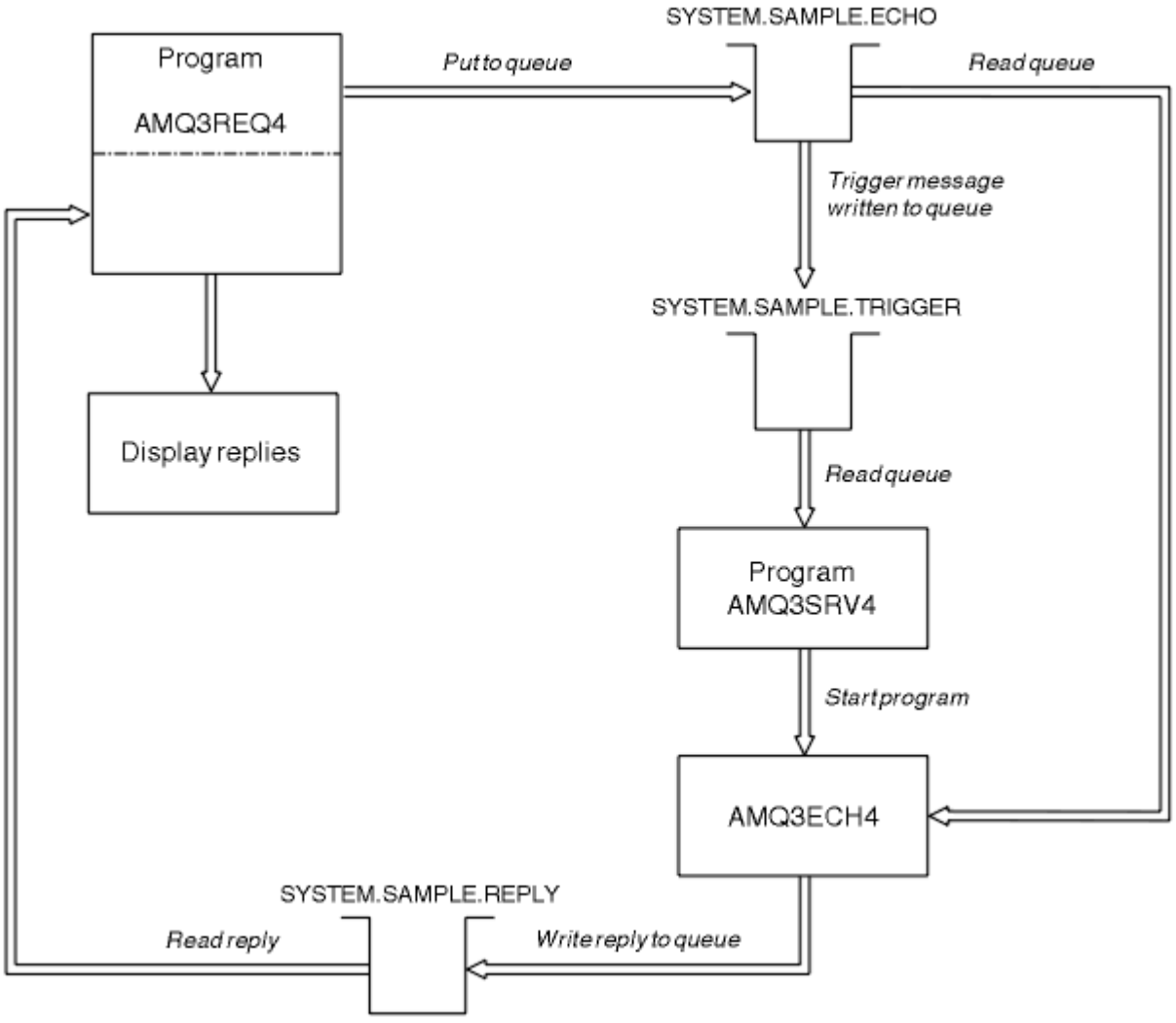


Figura 9. Diagramma di flusso del programma client / server di esempio (Echo)

IBM i *Utilizzo del trigger con l'esempio Richiesta su IBM i*

Per eseguire l'esempio utilizzando il trigger, avviare il programma del server del trigger, AMQ3SRV4, rispetto alla coda di avvio richiesta in un lavoro, quindi avviare AMQ3REQ4 in un altro lavoro.

Ciò significa che il server trigger è pronto quando il programma di esempio Richiesta invia un messaggio.

Nota:

1. Gli esempi utilizzano la coda SYSTEM SAMPLE TRIGGER come coda di avvio per SYSTEM.SAMPLE.ECHO, SYSTEM.SAMPLE.INQo code locali SYSTEM.SAMPLE.SET . In alternativa, è possibile definire la propria coda di avvio.
2. Le definizioni di esempio create da AMQSAMP4 causano l'attivazione della versione C dell'esempio. Se si desidera attivare la versione RPG, è necessario modificare le definizioni di processo SYSTEM.SAMPLE.ECHOPROCESS e SYSTEM.SAMPLE.INQPROCESS e SYSTEM.SAMPLE.SETPROCESS. È possibile utilizzare il comando CHGMQMPRC (per ulteriori dettagli, consultare Modifica processo MQ (CHGMQMPRC)) oppure modificare ed eseguire la propria versione di AMQSAMP4.
3. È necessario compilare il programma del server trigger dall'origine fornita in QMQMSAMP/QRPGLESRC.

A seconda del processo di trigger che si desidera eseguire, è necessario richiamare AMQ3REQ4 con il parametro che specifica i messaggi di richiesta da inserire in una delle seguenti code del server di esempio:

- SYSTEM.SAMPLE.ECHO (per i programmi di esempio Echo)
- SYSTEM.SAMPLE.INQ (per i programmi di esempio Inquire)
- SYSTEM.SAMPLE.SET (per i programmi di esempio Set)

Un grafico di flusso per SYSTEM.SAMPLE.ECHO è mostrato in Figura 9 a pagina 1461. Utilizzando l'esempio, il comando per emettere la richiesta del programma RPG a questo server è:

```
CALL PGM(QMQMSAMP/AMQ3REQ4) PARM('SYSTEM.SAMPLE.ECHO
+ 30 blank characters','Queue_Manager_Name')
```

perché il nome della coda e il nome del gestore code devono essere lunghi 48 caratteri.

Nota: Questa coda di esempio ha un tipo di trigger FIRST, quindi se ci sono già messaggi nella coda prima di eseguire l'esempio Richiesta, le applicazioni server non vengono attivate dai messaggi inviati.

Se si desidera provare ulteriori esempi, è possibile provare le seguenti variazioni:

- Utilizzare AMQ3TRG4 invece di AMQ3SRV4 per inoltrare il lavoro, ma i potenziali ritardi di inoltro del lavoro potrebbero rendere meno semplice seguire ciò che sta accadendo.
- Utilizzare SYSTEM.SAMPLE.INQ e SYSTEM.SAMPLE.SET . Utilizzando il file di dati di esempio, i comandi per emettere le richieste di programma RPG a questi server sono:

  ```
  CALL PGM(QMQMSAMP/AMQ3INQ4) PARM('SYSTEM.SAMPLE.INQ
  + 31 blank characters')
  CALL PGM(QMQMSAMP/AMQ3SET4) PARM('SYSTEM.SAMPLE.SET
  + 31 blank characters')
  ```

  poiché il nome coda deve essere lungo 48 caratteri.

  Queste code di esempio hanno anche un tipo di trigger FIRST.

Il programma di esempio Echo su IBM i

I programmi di esempio Echo restituiscono il messaggio inviato ad una coda di risposta. Il programma è denominato AMQ3ECH4

Perché il processo di attivazione funzioni, è necessario assicurarsi che il programma di esempio Echo che si desidera utilizzare venga attivato dai messaggi che arrivano sulla coda SYSTEM.SAMPLE.ECHO. A tale scopo, specificare il nome del programma di esempio Echo che si desidera utilizzare nel campo *ApplId* della definizione del processo SYSTEM.SAMPLE.ECHOPROCESS. (Per fare ciò, è possibile utilizzare il comando CHGMQMPRC, descritto in Amministrazione di IBM i .) La coda di esempio ha un tipo di trigger FIRST, quindi se ci sono già dei messaggi nella coda prima di eseguire l'esempio Richiesta, l'esempio Echo non viene attivato dai messaggi inviati.

Una volta impostata la definizione correttamente, avviare prima AMQ3SRV4 in un lavoro, quindi avviare AMQ3REQ4 in un'altro lavoro. È possibile utilizzare AMQ3TRG4 invece di AMQ3SRV4, ma i potenziali ritardi di inoltro dei lavori potrebbero rendere meno semplice seguire ciò che sta accadendo.

Utilizzare i programmi di esempio Richiesta per inviare messaggi alla coda SYSTEM.SAMPLE.ECHO. I programmi di esempio Echo inviano un messaggio di risposta contenente i dati nel messaggio di richiesta alla coda di risposta specificata nel messaggio di richiesta.

Progettazione del programma di esempio Echo

Quando il programma viene attivato, si connette esplicitamente al gestore code predefinito utilizzando la chiamata MQCONN. Sebbene ciò non sia necessario su IBM i, ciò significa che è possibile utilizzare lo stesso programma su altre piattaforme senza modificare il codice sorgente.

Il programma apre quindi la coda denominata nella struttura del messaggio trigger che è stata passata quando è stato avviato. (Per chiarezza, chiameremo questa *coda di richiesta*.) Il programma utilizza la chiamata MQOPEN per aprire questa coda per l'input condiviso.

Il programma utilizza la chiamata MQGET per rimuovere i messaggi da questa coda. Questa chiamata utilizza le opzioni GMATM e GMWT, con un intervallo di attesa di 5 secondi. Il programma verifica il descrittore di ciascun messaggio per verificare se si tratta di un messaggio di richiesta; in caso contrario, il programma elimina il messaggio e visualizza un messaggio di avvertenza.

Per ogni messaggio di richiesta rimosso dalla coda di richiesta, il programma utilizza la chiamata MQPUT per inserire un messaggio di risposta nella coda di risposta. Questo messaggio contiene il contenuto del messaggio di richiesta.

Quando non ci sono messaggi rimanenti sulla coda di richiesta, il programma chiude tale coda e si disconnette dal gestore code.

Questo programma può anche rispondere ai messaggi inviati alla coda da piattaforme diverse da IBM i, anche se non viene fornito alcun esempio per questa situazione. Per far funzionare il programma ECHO:

- Scrivere un programma, specificando correttamente i campi *Format*, *Encoding*e *CCSID* , per inviare messaggi di richiesta di testo.

  Il programma ECHO richiede al gestore code di eseguire la conversione dei dati del messaggio, se necessario.
- Specificare CONVERT (*YES) sul canale di invio IBM MQ for IBM i , se il programma scritto non fornisce una conversione simile per la risposta.

Il programma di esempio Inquiire su IBM i

Il programma di esempio di Inquire, AMQ3INQ4, interroga alcuni degli attributi di una coda utilizzando la chiamata MQINQ.

Il programma è progettato per essere eseguito come un programma attivato, quindi il suo unico input è una struttura MQTMC (trigger message). Questa struttura contiene il nome di una coda di destinazione con attributi su cui eseguire l'interrogazione.

Perché il processo di attivazione funzioni, è necessario accertarsi che il programma di esempio Inquire sia attivato dai messaggi in arrivo sulla coda SYSTEM.SAMPLE.INQ. Per eseguire questa operazione, specificare il nome del programma di esempio Inquire nel campo *ApplId* di SYSTEM.SAMPLE.INQPROCESS definizione del processo. (Per questo, è possibile utilizzare il comando CHGMQMPRC, descritto in Modifica processo MQ (CHGMQMPRC)). La coda di esempio ha un tipo di trigger FIRST, quindi se ci sono già dei messaggi nella coda prima di eseguire l'esempio Richiesta, l'esempio Inquire non viene attivato dai messaggi inviati.

Una volta impostata la definizione correttamente, avviare prima AMQ3SRV4 in un lavoro, quindi avviare AMQ3REQ4 in un'altro lavoro. È possibile utilizzare AMQ3TRG4 invece di AMQ3SRV4, ma i potenziali ritardi di inoltro dei lavori potrebbero rendere meno semplice seguire ciò che sta accadendo.

Utilizzare il programma di esempio Richiesta per inviare messaggi di richiesta, ciascuno contenente solo un nome coda, alla coda SYSTEM.SAMPLE.INQ. Per ogni messaggio di richiesta, il programma di esempio

Inquire invia un messaggio di risposta contenente informazioni sulla coda specificata nel messaggio di richiesta. Le risposte vengono inviate alla coda di risposta specificata nel messaggio di richiesta.

Progettazione del programma di esempio Inquire

Quando il programma viene attivato, si connette esplicitamente al gestore code predefinito utilizzando la chiamata MQCONN. Sebbene non sia necessario su IBM i, questa funzione di progettazione significa che è possibile utilizzare lo stesso programma su altre piattaforme senza modificare il codice sorgente.

Il programma apre quindi la coda denominata nella struttura del messaggio trigger che è stata passata quando è stato avviato. (Per chiarezza, chiameremo questa *coda di richiesta*.) Il programma utilizza la chiamata MQOPEN per aprire questa coda per l'input condiviso.

Il programma utilizza la chiamata MQGET per rimuovere i messaggi da questa coda. Questa chiamata utilizza le opzioni GMATM e GMWT, con un intervallo di attesa di 5 secondi. Il programma verifica il descrittore di ogni messaggio per verificare se si tratta di un messaggio di richiesta; in caso contrario, il programma elimina il messaggio e visualizza un messaggio di avvertenza.

Per ogni messaggio di richiesta rimosso da una coda di richiesta, il programma legge il nome della coda (che chiameremo *coda di destinazione*) contenuto nei dati e apre tale coda utilizzando il richiamo MQOPEN con l'opzione OOINQ. Il programma utilizza quindi la chiamata MQINQ per richiedere informazioni sui valori degli attributi **InhibitGet**, **CurrentQDepth**e **OpenInputCount** della coda di destinazione.

Se la chiamata MQINQ ha esito positivo, il programma utilizza la chiamata MQPUT per inserire un messaggio di risposta nella coda di risposta. Questo messaggio contiene i valori dei tre attributi.

Se la chiamata MQOPEN o MQINQ ha esito negativo, il programma utilizza la chiamata MQPUT per inserire un messaggio *report* nella coda di risposta. Nel campo *MDFB* del descrittore del messaggio di questo messaggio di report è presente il codice motivo restituito dalla chiamata MQOPEN o MQINQ, a seconda di quale non è riuscito.

Dopo la chiamata MQINQ, il programma chiude la coda di destinazione utilizzando la chiamata MQCLOSE.

Quando non ci sono messaggi rimanenti sulla coda di richiesta, il programma chiude tale coda e si disconnette dal gestore code.

Programma di esempio Set su IBM i

Il programma di esempio Set, AMQ3SET4, inibisce le operazioni di inserimento in una coda utilizzando la chiamata MQSET per modificare l'attributo **InhibitPut** della coda.

Il programma è concepito per essere eseguito come programma attivato, quindi il suo unico input è una struttura MQTMC (trigger message) che contiene il nome di una coda di destinazione con gli attributi su cui è necessario eseguire l'interrogazione.

Affinché il processo di attivazione funzioni, è necessario assicurarsi che il programma di esempio Set sia attivato dai messaggi in arrivo sulla coda SYSTEM.SAMPLE.SET. A tale scopo, specificare il nome del Programma di esempio Set nel campo *ApplId* della definizione del processo SYSTEM.SAMPLE.SETPROCESS. (Per questo, è possibile utilizzare il comando CHGMQMPRC, descritto in Amministrazione IBM i .) La coda di esempio ha un tipo di trigger FIRST, quindi se ci sono già messaggi nella coda prima di eseguire l'esempio Richiesta, l'esempio Imposta non viene attivato dai messaggi inviati.

Una volta impostata la definizione correttamente, avviare prima AMQ3SRV4 in un lavoro, quindi avviare AMQ3REQ4 in un'altro lavoro. È possibile utilizzare AMQ3TRG4 invece di AMQ3SRV4, ma i potenziali ritardi di inoltro dei lavori potrebbero rendere meno semplice seguire ciò che sta accadendo.

Utilizzare il programma di esempio Richiesta per inviare messaggi di richiesta, ciascuno contenente solo un nome coda, alla coda SYSTEM.SAMPLE.SET. Per ogni messaggio di richiesta, il programma di esempio Set invia un messaggio di risposta contenente una conferma che le operazioni di inserimento sono state inibite sulla coda specificata. Le risposte vengono inviate alla coda di risposta specificata nel messaggio di richiesta.

Progettazione del programma di esempio Set

Quando il programma viene attivato, si connette esplicitamente al gestore code predefinito utilizzando la chiamata MQCONN. Sebbene non sia necessario su IBM i, ciò significa che è possibile utilizzare lo stesso programma su altre piattaforme senza modificare il codice sorgente.

Il programma apre quindi la coda denominata nella struttura del messaggio trigger che è stata passata quando è stato avviato. (Per chiarezza, chiameremo questa *coda di richiesta*.) Il programma utilizza la chiamata MQOPEN per aprire questa coda per l'input condiviso.

Il programma utilizza la chiamata MQGET per rimuovere i messaggi da questa coda. Questa chiamata utilizza le opzioni GMATM e GMWT, con un intervallo di attesa di 5 secondi. Il programma verifica il descrittore di ciascun messaggio per verificare se si tratta di un messaggio di richiesta; in caso contrario, il programma elimina il messaggio e visualizza un messaggio di avvertenza.

Per ogni messaggio di richiesta rimosso da una coda di richiesta, il programma legge il nome della coda (che chiameremo *coda di destinazione*) contenuto nei dati e apre tale coda utilizzando la chiamata MQOPEN con opzione OOSET. Quindi, il programma utilizza la chiamata MQSET per impostare il valore dell'attributo **InhibitPut** della coda di destinazione su QAPUTI.

Se la chiamata MQSET ha esito positivo, il programma utilizza la chiamata MQPUT per inserire un messaggio di risposta nella coda di risposta. Questo messaggio contiene la stringa PUT `inhibited`.

Se la chiamata MQOPEN o MQSET non riesce, il programma utilizza la chiamata MQPUT per inserire un messaggio *report* nella coda di risposta. Nel campo *MDFB* del descrittore del messaggio di questo messaggio di report è presente il codice motivo restituito dalla chiamata MQOPEN o MQSET, a seconda di quale ha avuto esito negativo.

Dopo la chiamata MQSET, il programma chiude la coda di destinazione utilizzando la chiamata MQCLOSE.

Quando non ci sono messaggi rimanenti sulla coda di richiesta, il programma chiude tale coda e si disconnette dal gestore code.

Programmi di esempio Trigger su IBM i

IBM MQ for IBM i fornisce due programmi di esempio Trigger scritti in ILE/RPG.

I programmi sono:

AMQ3TRG4
: Questo è un controllo trigger per l'ambiente IBM i . Inoltra un lavoro IBM i per l'applicazione da avviare, ma ciò significa che vi sono ulteriori costi di elaborazione associati a ciascun messaggio trigger.

AMQ3SRV4
: Questo è un server di trigger per l'ambiente IBM i . Per ogni messaggio di trigger, questo server esegue il comando di avvio nel proprio lavoro per avviare l'applicazione specificata. Il server trigger può richiamare le transazioni CICS .

Le versioni in linguaggio C di questi esempi sono disponibili anche come programmi eseguibili nella libreria QMQM, denominata AMQSTRG4 e AMQSERV4.

Il controllo del trigger di esempio AMQ3TRG4 su IBM i

AMQ3TRG4 è un controllo trigger. Prende un parametro: il nome della coda di iniziazione che deve servire. AMQSAMP4 definisce una coda di avvio di esempio, SYSTEM.SAMPLE.TRIGGER, che è possibile utilizzare quando si provano i programmi di esempio.

AMQ3TRG4 inoltra un lavoro IBM i per ogni messaggio trigger valido che riceve dalla coda di iniziazione.

Progettazione del controllo trigger

Il controllo trigger apre la coda di iniziazione e riceve i messaggi dalla coda, specificando un intervallo di attesa illimitato.

Il controllo del trigger inoltra un lavoro IBM i per avviare l'applicazione specificata nel messaggio del trigger e trasmette una struttura MQTMC (una versione carattere del messaggio del trigger). I dati di ambiente nel messaggio trigger vengono utilizzati come parametri di inoltro del job.

Infine, il programma chiude la coda di iniziazione.

Il server trigger di esempio AMQ3SRV4
AMQ3SRV4 è un server trigger. Prende un parametro: il nome della coda di iniziazione che deve servire. AMQSAMP4 definisce una coda di avvio di esempio, SYSTEM.SAMPLE.TRIGGER, che è possibile utilizzare quando si provano i programmi di esempio.

Per ciascun messaggio trigger, AMQ3SRV4 esegue un comando di avvio nel proprio lavoro per avviare l'applicazione specificata.

Utilizzando la coda di trigger di esempio, il comando da emettere è:

```
CALL PGM(QMQM/AMQ3SRV4) PARM('Queue Name')
```

Dove `Queue Name` deve avere una lunghezza di 48 caratteri, che si ottiene riempiendo il nome della coda con il numero richiesto di spazi. Pertanto, se si utilizza SYSTEM.SAMPLE.TRIGGER come coda di destinazione, saranno necessari 28 caratteri vuoti.

Progettazione del server trigger

La progettazione del server trigger è simile a quella del controllo trigger, ad eccezione del server trigger:

- Consente applicazioni CICS e IBM i
- Non utilizza i dati di ambiente dal messaggio trigger
- Richiama le applicazioni IBM i nel proprio lavoro (o utilizza STRCICSUSR per avviare le applicazioni CICS) invece di inoltrare un lavoro IBM i
- Apre la coda di avvio per l'input condiviso, in modo che molti server trigger possano essere eseguiti contemporaneamente

Nota: I programmi avviati da AMQ3SRV4 non devono utilizzare la chiamata MQDISC perché questo arresterà il server trigger. Se i programmi avviati da AMQ3SRV4 utilizzano la chiamata MQCONN, riceveranno il codice di errore RC2002 .

Fine dei programmi di esempio Trigger su IBM i
Un programma di controllo trigger può essere terminato dall'opzione 2 di sysrequest (ENDRQS) o inibendo le ricezioni dalla coda trigger.

Se viene utilizzata la coda di trigger di esempio, il comando è:

```
CHGMQMQ QNAME('SYSTEM.SAMPLE.TRIGGER') GETENBL(*NO)
```

Nota: Per avviare nuovamente l'attivazione su questa coda, è necessario immettere il comando:

```
CHGMQMQ QNAME('SYSTEM.SAMPLE.TRIGGER') GETENBL(*YES)
```

Esecuzione degli esempi utilizzando le code remote su IBM i

È possibile dimostrare l'accodamento remoto eseguendo gli esempi sui gestori code di messaggi connessi.

Il programma AMQSAMP4 fornisce una definizione locale di una coda remota (SYSTEM.SAMPLE.REMOTE) che utilizza un gestore code remoto denominato 0THER. Per utilizzare questa definizione di esempio, modificare 0THER nel nome del secondo gestore code di messaggi che si desidera utilizzare. È inoltre necessario impostare un canale di messaggi tra i due gestori code di messaggi; per informazioni su come eseguire questa operazione, consultare Programmi di uscita canale per canali di messaggistica.

Il programma di esempio di richiesta inserisce il proprio nome gestore code locale nel campo *MDRM* dei messaggi inviati. Gli esempi Inquire e Set inviano i messaggi di risposta alla coda e al gestore code dei messaggi denominati nei campi *MDRQ* e *MDRM* dei messaggi di richiesta elaborati.

Codici di ritorno per IBM i (ILE RPG)

Queste informazioni descrivono i codici di ritorno associati a MQI e MQAI.

I codici di ritorno associati a:

- I comandi PCF (Programmable Command Format) sono elencati in Riferimento ai formati dei comandi programmabili.
- Le chiamate C++ sono elencate in Utilizzo di C++.

Per ogni chiamata, un codice di completamento e un codice motivo vengono restituiti dal gestore code o da una routine di uscita, per indicare l'esito positivo o negativo della chiamata.

Le applicazioni non devono dipendere da errori controllati in un ordine specifico, tranne dove specificamente indicato. Se più di un codice di completamento o di errore può derivare da una chiamata, l'errore particolare riportato dipende dall'implementazione.

Codici di completamento per IBM i (ILE RPG)

Il parametro del codice di completamento (*CMPCOD*) consente al chiamante di vedere rapidamente se la chiamata è stata completata correttamente, è stata completata parzialmente o non è riuscita.

CCOK
(MQCC_OK su altre piattaforme)

Completamento con esito positivo.

La chiamata è stata completata completamente; tutti i parametri di output sono stati impostati. Il parametro **REASON** ha sempre il valore RCNONE in questo caso.

AVVCCN
(MQCC_WARN su altre piattaforme)

Avvertenza (completamento parziale).

La chiamata è stata completata parzialmente. Alcuni parametri di output potrebbero essere stati impostati in aggiunta ai parametri di output *CMPCOD* e *REASON* . Il parametro **REASON** fornisce ulteriori informazioni sul completamento parziale.

CCNON RIUSCITO
(MQCC_FAIL su altre piattaforme)

Chiamata fallita.

L'elaborazione della chiamata non è stata completata e lo stato del gestore code è di norma invariato; le eccezioni sono riportate in modo specifico. I parametri di output *CMPCOD* e *REASON* sono stati impostati; gli altri parametri non vengono modificati, tranne dove indicato.

Il motivo potrebbe essere un errore nel programma applicativo o potrebbe essere il risultato di una situazione esterna al programma, ad esempio l'autorizzazione dell'utente potrebbe essere stata revocata. Il parametro **REASON** fornisce ulteriori informazioni sull'errore.

Codici di errore per IBM i (ILE RPG)

Il parametro del codice di errore (*REASON*) è una qualifica del parametro del codice di completamento (*CMPCOD*).

Se non vi è alcun motivo speciale per notificare, viene restituito RCNONE. Una chiamata riuscita restituisce CCOK e RCNONE.

Se il codice di completamento è CCWARN o CCFAIL, il gestore code riporta sempre un motivo valido; i dettagli vengono forniti sotto ogni descrizione di chiamata.

Quando le routine di uscita utente impostano codici di completamento e motivi, devono rispettare queste regole. Inoltre, i valori dei motivi speciali definiti dalle uscite utente devono essere inferiori a zero, per garantire che non siano in conflitto con i valori definiti dal gestore code. Le uscite possono impostare i motivi già definiti dal gestore code, dove sono appropriati.

I codici di errore si verificano anche in:

- Il campo *DLREA* della struttura MQDLH.
- Il campo *MDFB* della struttura di MQMD

Per un elenco completo dei codici di errore, consultare Codici di errore e completamento API.

Per trovare il codice motivo IBM i in tale elenco, rimuovere la "RC" dalla parte anteriore, ad esempio RC2002 diventa 2002. Anche i codici di completamento ci sono mostrati come sono su altre piattaforme:

Tabella 814. Nomi dei codici di errore su IBM i e su altre piattaforme

IBM i	**Altre piattaforme**
CCOK	MQCC_OK
AVVCCN	AVVERTENZA_MQCC
CCNON RIUSCITO	MQCC_NON RIUSCITA

Regole per la convalida delle opzioni MQI per IBM i (ILE RPG)

Questo argomento fornisce informazioni sulle situazioni che producono un codice di errore RC2046 da una chiamata MQOPEN, MQPUT, MQPUT1, MQGET o MQCLOSE.

Chiamata MQOPEN su IBM i

Per le opzioni della chiamata MQOPEN:

- È necessario specificare *almeno uno* dei seguenti:
  - OOBRW
  - OOINPQ
  - OOINPX
  - OOINPS
  - OOINQ
  - OOOUT
  - OSET
- È consentito solo *uno* dei seguenti:
  - OOINPQ
  - OOINPX
  - OOINPS
- È consentito solo *uno* dei seguenti:
  - OOBNDO
  - DN OBN
  - OOBNDQ

Nota: Le opzioni elencate in precedenza si escludono reciprocamente. Tuttavia, poiché il valore di OOBNDQ è zero, specificandolo con una delle due altre opzioni di bind non si ottiene il codice motivo RC2046. OOBNDQ viene fornito per aiutare la documentazione del programma.

- Se viene specificato OOSAVA, è necessario specificare anche una delle opzioni OOINP*.
- Se viene specificata una delle opzioni OOSET* o OOPAS*, è necessario specificare anche OOOUT.

Chiamata MQPUT su IBM i

Per le opzioni put - message:

- La combinazione di PMSYP e PMNSYP non è consentita.
- È consentito solo *uno* dei seguenti:
  - PMDEFC
  - NOC PM1
  - PMPASA
  - PMPASI
  - PMSETA
  - PMSETI
- PMALTU non è consentito (è valido solo sulla chiamata MQPUT1).

Chiamata MQPUT1 IBM i

Per le opzioni put - message, le regole sono le stesse della chiamata MQPUT, ad eccezione delle opzioni seguenti:

- PMALTU è consentito.
- PMLOGO non è consentito.

Chiamata MQGET su IBM i

Per le opzioni get - message:

- È consentita solo *una* delle seguenti opzioni:
  - GMNSYP
  - GMSYP
  - GMPSYP
- È consentita solo *una* delle seguenti opzioni:
  - GMBRWF
  - GMBRWC
  - GMBRWN
  - GMMUC
- GMSYP non è consentito con nessuna delle seguenti opzioni:
  - GMBRWF
  - GMBRWC
  - GMBRWN
  - GMLK
  - GMUNLK
- GMPSYP non è consentito con una delle seguenti opzioni:
  - GMBRWF

  - GMBRWC
  - GMBRWN
  - GMCMM
  - GMUNLK
- Se viene specificato GMLK, è necessario specificare anche una delle seguenti opzioni:
  - GMBRWF
  - GMBRWC
  - GMBRWN
- Se GMUNLK è specificato, sono consentite solo le seguenti opzioni:
  - GMNSYP
  - GMNWT

Chiamata MQCLOSE su IBM i

- Per le opzioni della chiamata MQCLOSE. La combinazione di CODEL e COPURG non è consentita.
- È consentito solo uno dei seguenti:
  - COKPSB
  - CORMSB

Chiamata MQSUB su IBM i

Per le opzioni della chiamata MQSUB:

- È necessario specificare almeno uno dei seguenti valori:
- È necessario specificare almeno uno dei seguenti valori:
  - SOALT
  - SORES
  - SOCRT
- È consentito solo uno dei seguenti:
  - SODUR
  - SONDUR

  Nota: Le opzioni elencate in precedenza si escludono reciprocamente. Tuttavia, poiché il valore di SOnDUR è zero, specificarlo con SODUR non risulta nel codice di errore RC2046. SONDUR viene fornito per aiutare la documentazione del programma.
- La combinazione di SOGRP e SOMAN non è consentita.
- SOGRP richiede la specifica di SOSCID.
- È consentito solo uno dei seguenti: SOAUID SOFUID
- La combinazione di SONEWP e SOPUBR non è consentita.
- SONEWP è consentito solo in combinazione con SOCRT.
- È consentito solo uno dei seguenti:
  - CHRSOW
  - SOWTOP

Codifiche macchina su IBM i

Utilizzare queste informazioni per conoscere la struttura del campo *MDENC* nel descrittore del messaggio.

Per ulteriori informazioni sul descrittore del messaggio, consultare “MQMD (Message Descriptor) su IBM i” a pagina 1140.

Il campo *MDENC* è un numero intero a 32 bit diviso in quattro sottocampi separati; questi sottocampi identificano:

- La codifica utilizzata per i numeri interi binari
- La codifica utilizzata per i numeri interi decimali compressi
- La codifica utilizzata per i numeri a virgola mobile
- Bit riservati

Ogni sottocampo è identificato da una maschera di bit che ha 1 bit nelle posizioni corrispondenti al sottocampo e 0 bit altrove. I bit sono numerati in modo che il bit 0 sia il bit più significativo e il bit 31 il bit meno significativo. Sono definite le seguenti maschere:

ITIMSC
Maschera per la codifica binario - intero.

Questo sottocampo occupa le posizioni da 28 a 31 nel campo *MDENC* .

ENDMSK
Maschera per la codifica packed - decimal - integer.

Questo sottocampo occupa le posizioni da 24 a 27 all'interno del campo *MDENC* .

ENFMSK
Maschera per la codifica a virgola mobile.

Questo campo secondario occupa le posizioni da 20 a 23 nel campo *MDENC* .

RNRMK
Maschera per bit riservati.

Questo sottocampo occupa le posizioni da 0 a 19 all'interno del campo *MDENC* .

IBM i Codifica numero intero binario su IBM i .

Valori validi per la codifica binario - intero.

I seguenti valori sono validi per la codifica binario - intero:

ENIUNDO
Codifica numero intero non definita.

I numeri interi binari vengono rappresentati utilizzando una codifica non definita.

ENINOR
Codifica numero intero normale.

I numeri interi binari sono rappresentati nel modo convenzionale:

- Il byte meno significativo nel numero ha l'indirizzo più alto di uno dei byte nel numero; il byte più significativo ha l'indirizzo più basso.
- Il bit meno significativo in ogni byte è accanto al byte con il successivo indirizzo più alto; il bit più significativo in ogni byte è accanto al byte con il successivo indirizzo più basso.

ENIREV
Codifica numero intero inversa.

I numeri interi binari sono rappresentati nello stesso modo di ENINOR, ma con i byte disposti in ordine inverso. I bit all'interno di ogni byte sono disposti allo stesso modo di ENINOR.

IBM i Codifica numero intero decimale compresso su IBM i

Valori validi per la codifica di numeri interi decimali compressi

I seguenti valori sono validi per la codifica numero intero decimale compresso:

ENDUND
Codifica decimale compressa non definita.

I numeri interi decimali compressi vengono rappresentati utilizzando una codifica non definita.

ENDNOR
Codifica decimale compressa normale.

I numeri interi decimali compressi sono rappresentati nel modo convenzionale:

- Ogni cifra decimale nel formato stampabile del numero è rappresentata in decimale compresso da una singola cifra esadecimale compresa nell'intervallo compreso tra X' 0 'e X' 9'. Ogni cifra esadecimale occupa 4 bit e quindi ogni byte nel numero decimale compresso rappresenta due cifre decimali nella forma stampabile del numero.
- Il byte meno significativo nel numero decimale compresso è il byte che contiene la cifra decimale meno significativa. All'interno di quel byte, i 4 bit più significativi contengono la cifra decimale meno significativa e i 4 bit meno significativi contengono il segno. Il segno è X'C '(positivo), X 'D' (negativo) o X'F ' (senza segno).
- Il byte meno significativo nel numero ha l'indirizzo più alto di uno dei byte nel numero; il byte più significativo ha l'indirizzo più basso.
- Il bit meno significativo in ogni byte è accanto al byte con il successivo indirizzo più alto; il bit più significativo in ogni byte è accanto al byte con il successivo indirizzo più basso.

TERMINA
Codifica decimale compresso inversa.

I numeri interi decimali compressi sono rappresentati nello stesso modo di ENDNOR, ma con i byte disposti in ordine inverso. I bit all'interno di ogni byte sono disposti nello stesso modo di ENDNOR.

IBM i Codifica a virgola mobile su IBM i

Valori validi per la codifica a virgola mobile

I seguenti valori sono validi per la codifica a virgola mobile:

ENFUND
Codifica a virgola mobile non definita.

I numeri a virgola mobile vengono rappresentati utilizzando una codifica non definita.

ENFNOR
Normale codifica a virgola mobile IEEE (Institute of Electrical and Electronics Engineers).

I numeri a virgola mobile sono rappresentati utilizzando il formato a virgola mobile IEEE standard, con i byte disposti come segue:

- Il byte meno significativo nella mantissa ha l'indirizzo più alto di uno qualsiasi dei byte nel numero; il byte contenente l'esponente ha l'indirizzo più basso
- Il bit meno significativo in ogni byte è accanto al byte con l'indirizzo superiore successivo; il bit più significativo in ciascun byte è accanto al byte con l'indirizzo inferiore successivo

I dettagli della codifica float IEEE possono essere trovati nello standard IEEE 754.

ENFREV
Codifica a virgola mobile IEEE inversa.

I numeri a virgola mobile sono rappresentati nello stesso modo di ENFNOR, ma con i byte disposti in ordine inverso. I bit all'interno di ogni byte sono disposti nello stesso modo di ENFNOR.

ENF390
Codifica a virgola mobile dell'architettura System/390 .

I numeri a virgola mobile vengono rappresentati utilizzando il formato a virgola mobile System/390 standard; questo viene utilizzato anche da System/370.

IBM i Costruzione di codifiche su IBM i

Per creare un valore per il campo *MDENC* in MQMD, è necessario aggiungere le costanti pertinenti che descrivono le codifiche richieste.

Assicurarsi di combinare solo una delle codifiche ENI* con una delle codifiche END* e una delle codifiche ENF*.

IBM i Analisi delle codifiche su IBM i

Il campo *MDENC* contiene sottocampi; per questo motivo, le applicazioni che devono esaminare la codifica integer, packed decimal o float devono utilizzare la tecnica descritta in questo argomento.

Utilizzo dell'aritmetica

Le seguenti operazioni devono essere eseguite utilizzando l'aritmetica dei numeri interi:

1. Selezionare uno dei seguenti valori, in base al tipo di codifica richiesto:
   - 1 per la codifica di numeri interi binari
   - 16 per la codifica del numero intero decimale compresso
   - 256 per la codifica a virgola mobile

   Richiamare il valore A.
2. Dividere il valore del campo *MDENC* per A ; richiamare il risultato B.
3. Dividere B per 16; richiamare il risultato C.
4. Moltiplicare C per 16 e sottrarre da B ; richiamare il risultato D.
5. Moltiplicare D per A ; richiamare il risultato E.
6. E è la codifica richiesta e può essere testata per l'uguaglianza con ciascuno dei valori validi per quel tipo di codifica.

IBM i Riepilogo delle codifiche dell'architettura della macchina su IBM i

Una tabella che riepiloga le codifiche per le architetture della macchina.

Le codifiche per le architetture delle macchine sono mostrate in Tabella 815 a pagina 1473.

Tabella 815. Riepilogo delle codifiche per le architetture delle macchine

Architettura macchina	Codifica numero intero binario	Codifica numero intero decimale compresso	Codifica a virgola mobile
IBM i	normale	normale	IEEE normale
Intel x86	Inverso	Inverso	IEEE invertito
PowerPC	normale	normale	IEEE normale
System/390	normale	normale	System/390

IBM i Opzioni di report e indicatori di messaggi su IBM i

Questo argomento riguarda i campi *MDREP* e *MDMFL* che fanno parte del descrittore del messaggio MQMD specificato nelle chiamate MQGET, MQPUT e MQPUT1 .

Per ulteriori informazioni sul descrittore del messaggio, consultare “MQMD (Message Descriptor) su IBM i” a pagina 1140. Queste informazioni descrivono:

- La struttura del campo del report e il modo in cui il gestore code lo elabora
- In che modo un'applicazione deve analizzare il campo del prospetto
- La struttura del campo message - flags

Struttura del campo del report

Il campo *MDREP* è un numero intero a 32 bit diviso in tre sottocampi separati.

Questi sottocampi identificano:

- Opzioni di report rifiutate se il gestore code locale non le riconosce
- Opzioni di report sempre accettate, anche se il gestore code locale non le riconosce
- Opzioni di report accettate solo se sono soddisfatte determinate altre condizioni

Ogni sottocampo è identificato da una maschera di bit che ha 1 bit nelle posizioni corrispondenti al sottocampo e 0 bit altrove. Si noti che i bit in un campo secondario non sono necessariamente adiacenti. I bit sono numerati in modo che il bit 0 sia il bit più significativo e il bit 31 il bit meno significativo. Le seguenti maschere sono definite per identificare i sottocampi:

RORUM
Maschera per le opzioni di report non supportate rifiutate.

Questa maschera identifica le posizioni di bit all'interno del campo *MDREP* in cui le opzioni di report non supportate dal gestore code locale causeranno l'esito negativo della chiamata MQPUT o MQPUT1 con codice di completamento CCFAIL e codice di errore RC2061.

Questo sottocampo occupa le posizioni bit 3 e da 11 a 13.

ROAUM
Maschera per le opzioni di report non supportate accettate.

Questa maschera identifica le posizioni di bit all'interno del campo *MDREP* in cui le opzioni di report non supportate dal gestore code locale verranno tuttavia accettate nelle chiamate MQPUT o MQPUT1 . In questo caso, viene restituito il codice di completamento CCWARN con codice motivo RC2104 .

Questo sottocampo occupa le posizioni da 0 a 2, da 4 a 10 e da 24 a 31.

Le seguenti opzioni del report sono incluse in questo campo secondario:

- ROCMTC
- RODLQ
- RODISC
- ROEXC
- ROEXCD
- ROEXCF
- ROEXP
- ROEXPD
- ROEXPF
- RONAN
- RONMI
- RONONE
- ROPAN
- ROPCI
- RAMMI

ROAUXM
Maschera per le opzioni di report non supportate che sono accettate solo in determinate circostanze.

Questa maschera identifica le posizioni di bit all'interno del campo *MDREP* in cui le opzioni di report non supportate dal gestore code locale verranno tuttavia accettate sulle chiamate MQPUT o MQPUT1 *purché* siano soddisfatte entrambe le condizioni riportate di seguito:

- Il messaggio è destinato a un gestore code remoto.
- L'applicazione non sta inserendo il messaggio direttamente in una coda di trasmissione locale (ossia, la coda identificata dai campi *ODMN* e *ODON* nel descrittore oggetto specificato nella chiamata MQOPEN o MQPUT1 non è una coda di trasmissione locale).

Il codice di completamento CCWARN con codice di errore RC2104 viene restituito se queste condizioni sono soddisfatte e CCFAIL con codice di errore RC2061 in caso contrario.

Questo sottocampo occupa le posizioni di bit da 14 a 23.

Le seguenti opzioni del report sono incluse in questo campo secondario:

- ROCOA
- ROCOAD
- ROCOAF
- ROCOD
- ARROTONDA
- ROCODF

Se sono presenti opzioni specificate nel campo *MDREP* che il gestore code non riconosce, il gestore code controlla ciascun campo secondario utilizzando l'operazione AND bit per bit per combinare il campo *MDREP* con la maschera per tale campo secondario. Se il risultato di tale operazione è diverso da zero, vengono restituiti il codice di completamento e i codici di errore descritti in precedenza.

Se viene restituito CCWARN, non viene definito quale codice di errore viene restituito se esistono altre condizioni di avvertenza.

La possibilità di specificare e accettare opzioni di report non riconosciute dal gestore code locale è utile quando è necessario inviare un messaggio con un'opzione di report che verrà riconosciuto ed elaborato da un gestore code *remoto* .

IBM i Analisi del campo del prospetto su IBM i

Il campo MDREP contiene campi secondari. Per questo motivo, alcune applicazioni devono verificare se il mittente del messaggio ha richiesto un particolare report. Tali applicazioni devono utilizzare la tecnica descritta in questo argomento.

Utilizzo dell'aritmetica

Le seguenti operazioni devono essere eseguite utilizzando l'aritmetica dei numeri interi:

1. Selezionare uno dei seguenti valori, in base al tipo di report da controllare:
   - Report ROCOA per COA
   - Report ROCOD per COD
   - ROEXC per report di eccezione
   - report ROEXP per scadenza

   Richiamare il valore A.
2. Dividere il campo *MDREP* per A ; richiamare il risultato B.
3. Dividere B per 8 ; richiamare il risultato C.
4. Moltiplicare C per 8 e sottrarre da B ; richiamare il risultato D.
5. Moltiplicare D per A ; richiamare il risultato E.
6. Verificare E per verificare l'uguaglianza con ciascuno dei valori possibili per quel tipo di report.

Ad esempio, se A è ROEXC, verificare E per verificare l'uguaglianza con ciascuno dei seguenti elementi per determinare cosa è stato specificato dal mittente del messaggio:

- RONONE
- ROEXC
- ROEXCD
- ROEXCF

I test possono essere eseguiti in qualsiasi ordine sia più conveniente per la logica dell'applicazione.

Il seguente pseudocodice illustra questa tecnica per i messaggi di report di eccezioni:

```
A = ROEXC
B = Report/A
C = B/8
D = B - C*8
E = D*A
```

Un metodo simile può essere utilizzato per verificare le opzioni ROPMI o ROPCI; selezionare come valore A qualunque di queste due costanti sia appropriato e procedere come descritto in precedenza, ma sostituendo il valore 8 nei passi precedenti con il valore 2.

IBM i Struttura del campo message - flags su IBM i

Il campo *MDMFL* è un numero intero a 32 bit diviso in tre sottocampi separati.

Questi sottocampi identificano:

- Indicatori di messaggio rifiutati se il gestore code locale non li riconosce
- Indicatori di messaggi sempre accettati, anche se il gestore code locale non li riconosce
- Indicatori di messaggio accettati solo se sono soddisfatte determinate altre condizioni

Nota: Tutti i campi secondari in *MDMFL* sono riservati per l'utilizzo da parte del gestore code.

Ogni sottocampo è identificato da una maschera di bit che ha 1 bit nelle posizioni corrispondenti al sottocampo e 0 bit altrove. I bit sono numerati in modo che il bit 0 sia il bit più significativo e il bit 31 il bit meno significativo. Le seguenti maschere sono definite per identificare i sottocampi:

MFRUM
Maschera per indicatori di messaggi non supportati rifiutati.

Questa maschera identifica le posizioni di bit all'interno del campo *MDMFL* in cui gli indicatori di messaggio non supportati dal gestore code locale causeranno l'esito negativo della chiamata MQPUT o MQPUT1 con codice di completamento CCFAIL e codice di errore RC2249.

Questo sottocampo occupa le posizioni da 20 a 31.

I seguenti indicatori di messaggio sono inclusi in questo campo secondario:

- MFLMIG
- MFLSEG
- MMIG
- MFSEG
- MFSEGA
- MFSEGI

MFAUM
Maschera per gli indicatori di messaggi non supportati accettati.

Questa maschera identifica le posizioni dei bit all'interno del campo *MDMFL* in cui gli indicatori dei messaggi non supportati dal gestore code locale verranno comunque accettati nelle chiamate MQPUT o MQPUT1 . Il codice di completamento è CCOK.

Questo sottocampo occupa le posizioni di bit da 0 a 11.

MFAUXM
Maschera per gli indicatori di messaggi non supportati accettati solo in determinate circostanze.

Questa maschera identifica le posizioni dei bit all'interno del campo *MDMFL* in cui gli indicatori dei messaggi non supportati dal gestore code locale verranno tuttavia accettati nelle chiamate MQPUT o MQPUT1 *purché* siano soddisfatte entrambe le condizioni riportate di seguito:

- Il messaggio è destinato a un gestore code remoto.
- L'applicazione non sta inserendo il messaggio direttamente in una coda di trasmissione locale (ossia, la coda identificata dai campi *ODMN* e *ODON* nel descrittore oggetto specificato nella chiamata MQOPEN o MQPUT1 non è una coda di trasmissione locale).

Il codice di completamento CCOK viene restituito se queste condizioni sono soddisfatte e CCFAIL con codice motivo RC2249 in caso contrario.

Questo sottocampo occupa le posizioni bit da 12 a 19.

Se ci sono indicatori specificati nel campo *MDMFL* che il gestore code non riconosce, il gestore code controlla ogni campo secondario a turno utilizzando l'operazione AND bit per bit per combinare il campo *MDMFL* con la maschera per tale campo secondario. Se il risultato di tale operazione è diverso da zero, vengono restituiti il codice di completamento e i codici di errore descritti in precedenza.

IBM i Conversione dati su IBM i

Questo topic descrive l'interfaccia per l'uscita di conversione dati e l'elaborazione eseguita dal gestore code quando è richiesta la conversione dati.

L'uscita conversione dati viene richiamata come parte dell'elaborazione della chiamata MQGET. Viene utilizzato per convertire i dati del messaggio dell'applicazione nella rappresentazione richiesta dall'applicazione ricevente. La conversione dei dati del messaggio dell'applicazione è facoltativa e richiede che l'opzione GMCONV sia specificata nella chiamata MQGET.

Sono descritti i seguenti aspetti della conversione dei dati:

- L'elaborazione eseguita dal gestore code in risposta all'opzione GMCONV; consultare “Elaborazione conversione su IBM i” a pagina 1477.
- Convenzioni di elaborazione utilizzate dal gestore code quando si elabora un formato integrato; queste convenzioni sono consigliate anche per le uscite scritte dall'utente. Consultare “Convenzioni di elaborazione su IBM i” a pagina 1479.
- Considerazioni speciali per la conversione dei messaggi di report; consultare “Conversione dei messaggi di report su IBM i” a pagina 1483.
- I parametri passati all'exit di conversione dati; consultare “MQCONVX (Data conversion exit) su IBM i” a pagina 1494.
- Una chiamata che può essere utilizzata dall'uscita per convertire i dati carattere tra rappresentazioni differenti; consultare “MQXCNVC (Converti caratteri) su IBM i” a pagina 1489.
- Il parametro della struttura dati specifico per l'uscita; consultare “MQDXP (parametro di uscita conversione dati) su IBM i” a pagina 1484.

IBM i Elaborazione conversione su IBM i

Queste informazioni descrivono l'elaborazione eseguita dal gestore code in risposta all'opzione GMCONV.

Il gestore code esegue le seguenti azioni se l'opzione GMCONV è specificata nella chiamata MQGET e c'è un messaggio da restituire all'applicazione:

1. Se si verifica una o più delle seguenti condizioni, non è necessaria alcuna conversione:

- I dati del messaggio sono già nella serie di caratteri e nella codifica richiesta dall'applicazione che emette la chiamata MQGET. L'applicazione deve impostare i campi *MDCSI* e *MDENC* nel parametro **MSGDSC** della chiamata MQGET sui valori richiesti, prima di emettere la chiamata.
- La lunghezza dei dati del messaggio è zero.
- La lunghezza del parametro **BUFFER** della chiamata MQGET è zero.

In questi casi, il messaggio viene restituito senza conversione all'applicazione che emette la chiamata MQGET; i valori *MDCSI* e *MDENC* nel parametro **MSGDSC** sono impostati sui valori nelle informazioni di controllo nel messaggio e la chiamata viene completata con una delle seguenti combinazioni di codice di completamento e codice motivo:

Codice di completamento
Codice di errore

CCOK
RCNONE

AVVCCN
RC2079

AVVCCN
RC2080

Le seguenti operazioni vengono eseguite solo se la serie di caratteri o la codifica dei dati del messaggio differiscono dal valore corrispondente nel parametro **MSGDSC** e vi sono dati da convertire:

1. Se il campo *MDFMT* nelle informazioni di controllo nel messaggio ha il valore FMNONE, il messaggio viene restituito non convertito, con codice di completamento CCWARN e codice motivo RC2110.

   In tutti gli altri casi l'elaborazione della conversione continua.
2. Il messaggio viene rimosso dalla coda e collocato in un buffer temporaneo della stessa dimensione del parametro **BUFFER** . Per le operazioni di ricerca, il messaggio viene copiato nel buffer temporaneo, invece di essere rimosso dalla coda.
3. Se il messaggio deve essere troncato per adattarsi al buffer, viene eseguito quanto segue:
   - Se l'opzione GMATM non è stata specificata, il messaggio viene restituito non convertito, con codice di completamento CCWARN e codice di errore RC2080.
   - Se l'opzione GMATM *è stata* specificata, il codice di completamento è impostato su CCWARN, il codice motivo è impostato su RC2079e l'elaborazione della conversione continua.
4. Se il messaggio può essere inserito nel buffer senza troncare o se è stata specificata l'opzione GMATM, viene eseguita la seguente operazione:
   - Se il formato è un formato integrato, il buffer viene passato al servizio di conversione dati del gestore code.
   - Se il formato non è un formato incorporato, il buffer viene passato a un'uscita scritta dall'utente che ha lo stesso nome del formato. Se non è possibile trovare l'uscita, il messaggio viene restituito non convertito, con codice di completamento CCWARN e codice di errore RC2110.

   Se non si verifica alcun errore, l'output del servizio di conversione dati o dell'uscita scritta dall'utente è il messaggio convertito, più il codice di completamento e il codice motivo da restituire all'applicazione che emette la chiamata MQGET.
5. Se la conversione ha esito positivo, il gestore code restituisce il messaggio convertito all'applicazione. In tal caso, il codice di completamento e il codice motivo restituiti dalla chiamata MQGET saranno generalmente una delle seguenti combinazioni:

   Codice di completamento
   Codice di errore

   CCOK
   RCNONE

   AVVCCN
   RC2079

Tuttavia, se la conversione viene eseguita da un'uscita scritta dall'utente, possono essere restituiti altri codici di errore, anche quando la conversione ha esito positivo.

Se la conversione ha esito negativo (per qualsiasi motivo), il gestore code restituisce il messaggio non convertito all'applicazione, con i campi *MDCSI* e *MDENC* nel parametro **MSGDSC** impostati sui valori nelle informazioni di controllo nel messaggio e con il codice di completamento CCWARN.

IBM i Convenzioni di elaborazione su IBM i

Quando si converte un formato integrato, il gestore code segue le convenzioni di elaborazione descritte in questo argomento.

Si consiglia di applicare queste convenzioni alle uscite scritte dall'utente, anche se ciò non viene applicato dal gestore code. I formati integrati convertiti dal gestore code sono i seguenti:

- MN FMADM
- FMMDE
- FMCICS
- FMPCF
- FMCMD1
- MMRM
- FMCMD2
- FMRF
- FMDLH
- FMRFH2
- FMDH
- FMSTR
- FMEVNT
- FMTM
- FMIMS
- FMXQH
- FMIMVS

1. Se il messaggio si espande durante la conversione e supera la dimensione del parametro **BUFFER** , viene eseguito quanto segue:
   - Se l'opzione GMATM non è stata specificata, il messaggio viene restituito non convertito, con codice di completamento CCWARN e codice di errore RC2120.
   - Se l'opzione GMATM *è stata* specificata, il messaggio viene troncato, il codice di completamento viene impostato su CCWARN, il codice di origine errore viene impostato su RC2079e l'elaborazione della conversione continua.
2. Se si verifica un troncamento (prima o durante la conversione), è possibile che il numero di byte validi restituiti nel parametro **BUFFER** sia *inferiore* alla lunghezza del buffer.

   Ciò può verificarsi, ad esempio, se un numero intero a 4 byte o un carattere DBCS si trova alla fine del buffer. L'elemento incompleto delle informazioni non viene convertito e quindi i byte nel messaggio restituito non contengono informazioni valide. Ciò può verificarsi anche se un messaggio troncato prima della conversione si riduce durante la conversione.

   Se il numero di byte validi restituiti è inferiore alla lunghezza del buffer, i byte inutilizzati alla fine del buffer vengono impostati su valori null.
3. Se una schiera o una stringa si trova a cavallo della fine del buffer, viene convertita la maggior parte dei dati possibile; non viene convertito solo il particolare elemento della schiera o il carattere DBCS incompleto - vengono convertiti i caratteri o gli elementi della schiera precedenti.

4. Se si verifica il troncamento (prima o durante la conversione), la lunghezza restituita per il parametro **DATLEN** è la lunghezza del messaggio *non convertito* prima del troncamento.
5. Quando le stringhe vengono convertite tra serie di caratteri a byte singolo (SBCS), serie di caratteri a byte doppio (DBCS) o serie di caratteri a più byte (MBCS), le stringhe possono espandersi o contrarsi.
   - Nei formati PCF FMADMN, FMEVNT e FMPCF, le stringhe nelle strutture MQCFST e MQCFSL si espandono o si contraggono come necessario per adattare la stringa dopo la conversione.

     Per la struttura dell'elenco di stringhe MQCFSL, le stringhe nell'elenco potrebbero espandersi o contrarsi in base a quantità differenti. Se ciò si verifica, il gestore code riempirebbe le stringhe più corte con spazi vuoti per renderle della stessa lunghezza della stringa più lunga dopo la conversione.
   - Nel formato FMRMH, le stringhe indirizzate dai campi RMSEO, RMSNO, RMDEOe RMDNO si espandono o si contraggono in base alle necessità per contenere le stringhe dopo la conversione.
   - Nel formato FMRFH, il campo RFNVS si espande o si contrae come necessario per adattare le coppie nome - valore dopo la conversione.
   - Nelle strutture con dimensioni di campo fisse, il gestore code consente alle stringhe di espandersi o contrarsi all'interno dei relativi campi fissi, se non si perdono informazioni significative. A questo proposito, gli spazi vuoti finali e i caratteri che seguono il primo carattere null nel campo vengono trattati come non significativi.
     - Se la stringa si espande, ma solo i caratteri non significativi devono essere scartati per contenere la stringa convertita nel campo, la conversione ha esito positivo e la chiamata viene completata con CCOK e codice di errore RCNONE (supponendo che non vi siano altri errori).
     - Se la stringa si espande, ma la stringa convertita richiede caratteri significativi da eliminare per rientrare nel campo, il messaggio viene restituito non convertito e la chiamata viene completata con CCWARN e codice di errore RC2190.

       Nota: Il codice di errore RC2190 determina in questo caso se è stata specificata l'opzione GMATM.
     - Se la stringa si contrae, il gestore code riempiendo la stringa con spazi vuoti fino alla lunghezza del campo.
6. Per i messaggi costituiti da una o più strutture di intestazione IBM MQ seguite da dati utente, è possibile convertire una o più strutture di intestazione, mentre il resto del messaggio non lo è. Tuttavia, con due eccezioni, i campi MDCSI e MDENC in ogni struttura di intestazione indicano sempre correttamente la serie di caratteri e la codifica dei dati che seguono la struttura di intestazione.

   Le due eccezioni sono le strutture MQCIH e MQIIH, in cui i valori nei campi MDCSI e MDENC in tali strutture non sono significativi. Per tali strutture, i dati che seguono la struttura si trovano nella stessa serie di caratteri e nella stessa codifica della struttura MQCIH o MQIIH.
7. Se i campi MDCSI o MDENC nelle informazioni di controllo del messaggio da richiamare o nel parametro **MSGDSC** specificano valori non definiti o non supportati, il gestore code potrebbe ignorare l'errore se non è necessario utilizzare il valore non definito o non supportato nella conversione del messaggio.

   Ad esempio, se il campo MDENC nel messaggio specifica una codifica a virgola mobile non supportata, ma il messaggio contiene solo dati interi o contiene dati a virgola mobile che non richiedono la conversione (poiché le codifiche a virgola mobile di origine e di destinazione sono identiche), l'errore potrebbe essere diagnosticato o meno.

   Se l'errore viene diagnosticato, il messaggio non viene convertito, con codice di completamento CCWARN e uno dei codici di errore RC2111, RC2112, RC2113, RC2114 o RC2115, RC2116, RC2117, RC2118 (come appropriato); i campi MDCSI e MDENC nel parametro **MSGDSC** sono impostati sui valori nelle informazioni di controllo nel messaggio.

   Se l'errore non viene diagnosticato e la conversione viene completata correttamente, i valori restituiti nei campi MDCSI e MDENC nel parametro **MSGDSC** sono quelli specificati dall'applicazione che emette la chiamata MQGET.

8. In tutti i casi, se il messaggio viene restituito all'applicazione non convertita, il codice di completamento è impostato su CCWARN e i campi `MDCSI` e `MDENC` nel parametro **MSGDSC** sono impostati sui valori appropriati per i dati non convertiti. Questo viene fatto anche per FMNONE.

   Il parametro **REASON** è impostato su un codice che indica il motivo per cui non è stato possibile eseguire la conversione, a meno che non sia stato necessario troncare anche il messaggio; i codici di errore relativi al troncamento hanno la precedenza sui codici di errore relativi alla conversione. (Per determinare se un messaggio troncato è stato convertito, controllare i valori restituiti nei campi `MDCSI` e `MDENC` nel parametro **MSGDSC** .)

   Quando viene diagnosticato un errore, viene restituito un codice motivo specifico o il codice motivo generale RC2119. Il codice motivo restituito dipende dalle capacità diagnostiche del servizio di conversione dati sottostante.
9. Se viene restituito il codice di completamento CCWARN e più di un codice di errore è rilevante, l'ordine di precedenza è il seguente:
   a. Il seguente motivo ha la precedenza su tutti gli altri:
      - RC2079
   b. Avanti in precedenza è il seguente motivo:
      - RC2110
   c. L'ordine di precedenza all'interno dei codici di errore rimanenti non è definito.
10. Al completamento della chiamata MQGET:
    - Il seguente codice di errore indica che il messaggio è stato convertito correttamente:
      – RCNONE
    - Il seguente codice di errore indica che il messaggio *potrebbe* essere stato convertito correttamente (controllare i campi `MDCSI` e `MDENC` nel parametro **MSGDSC** per scoprirlo):
      – RC2079
    - Tutti gli altri codici di errore indicano che il messaggio non è stato convertito.

La seguente elaborazione è specifica per i formati integrati; non è applicabile ai formati definiti dall'utente:

1. Tranne che per i formati seguenti:
   - MN FMADM
   - FMEVNT
   - FMIMVS
   - FMPCF
   - FMSTR

   nessuno dei formati integrati può essere convertito da o in serie di caratteri che non dispongono di caratteri SBCS per i caratteri validi nei nomi delle code. Se viene effettuato un tentativo di eseguire tale conversione, il messaggio viene restituito non convertito, con codice di completamento CCWARN e codice motivo RC2111 o RC2115, come appropriato.

   La serie di caratteri Unicode UTF-16 è un esempio di una serie di caratteri che non dispone di caratteri SBCS per i caratteri validi nei nomi coda.
2. Se i dati del messaggio per un formato integrato vengono troncati, i campi all'interno del messaggio che contengono lunghezze di stringhe o conteggi di elementi o strutture non vengono regolati per riflettere la lunghezza dei dati restituiti all'applicazione; i valori restituiti per tali campi all'interno dei dati del messaggio sono i valori applicabili al messaggio prima del troncamento.

   Quando si elaborano messaggi come un messaggio FMADMN troncato, è necessario prestare attenzione per assicurarsi che l'applicazione non tenti di accedere ai dati oltre la fine dei dati restituiti.
3. Se il nome formato è FMDLH, i dati del messaggio iniziano con una struttura MQDLH e questo può essere seguito da zero o più byte di dati del messaggio dell'applicazione. Il formato, la serie di

caratteri e la codifica dei dati del messaggio dell'applicazione sono definiti dai campi DLFMT, DLCSIe DLENC nella struttura MQDLH all'inizio del messaggio. Dal momento che la struttura MQDLH e i dati del messaggio dell'applicazione possono avere diverse serie di caratteri e codifiche, è possibile che una, un'altra o entrambe le strutture MQDLH e i dati del messaggio dell'applicazione richiedano la conversione.

Il gestore code converte prima la struttura MQDLH, come necessario. Se la conversione ha esito positivo o se la struttura MQDLH non richiede la conversione, il gestore code controlla i campi DLCSI e DLENC nella struttura MQDLH per verificare se è richiesta la conversione dei dati del messaggio dell'applicazione. Se la conversione è richiesta, il gestore code richiama l'uscita scritta dall'utente con il nome fornito dal campo DLFMT nella struttura MQDLH oppure esegue la conversione stessa (se DLFMT è il nome di un formato integrato).

Se la chiamata MQGET restituisce un codice di completamento CCWARN e il codice di errore è uno di quelli che indicano che la conversione non è riuscita, si applica una delle seguenti condizioni:

- Non è possibile convertire la struttura MQDLH. In questo caso, anche i dati del messaggio dell'applicazione non saranno stati convertiti.
- La struttura MQDLH è stata convertita, ma non i dati del messaggio dell'applicazione.

L'applicazione può esaminare i valori restituiti nei campi MDCSI e MDENC nel parametro **MSGDSC** e quelli nella struttura MQDLH, per determinare quale dei precedenti si applica.

4. Se il nome del formato è FMXQH, i dati del messaggio iniziano con una struttura MQXQH e questo può essere seguito da zero o più byte di dati aggiuntivi. Questi dati aggiuntivi sono in genere i dati del messaggio dell'applicazione (che possono essere di lunghezza zero), ma possono essere presenti anche una o più ulteriori strutture di intestazione IBM MQ , all'inizio dei dati aggiuntivi.

   La struttura MQXQH deve essere nella serie di caratteri e nella codifica del gestore code. Il formato, la serie di caratteri e la codificazione dei dati che seguono la struttura MQXQH sono forniti dai campi MDFMT, MDCSIe MDENC nella struttura MQMD contenuta in MQXQH. Per ogni successiva struttura di intestazione IBM MQ presente, i campi MDFMT, MDCSIe MDENC nella struttura descrivono i dati che seguono tale struttura; tali dati sono un'altra struttura di intestazione IBM MQ o i dati del messaggio dell'applicazione.

   Se l'opzione GMCONV viene specificata per un messaggio FMXQH, i dati del messaggio dell'applicazione e alcune strutture dell'intestazione MQ vengono convertiti, ma i dati nella struttura MQXQH non lo sono. Al ritorno dalla chiamata MQGET, quindi:

   - I valori dei campi MDFMT, MDCSIe MDENC nel parametro **MSGDSC** , descrivono i dati nella struttura MQXQH e non i dati del messaggio dell'applicazione; pertanto, i valori non saranno uguali a quelli specificati dall'applicazione che ha emesso la chiamata MQGET.

     Il risultato è che un'applicazione che riceve ripetutamente i messaggi da una coda di trasmissione con l'opzione GMCONV specificata deve reimpostare i campi MDCSI e MDENC nel parametro **MSGDSC** sui valori necessari per i dati del messaggio dell'applicazione, prima di ogni chiamata MQGET.
   - I valori dei campi MDFMT, MDCSIe MDENC presenti nell'ultima struttura di intestazioni MQ descrivono i dati del messaggio dell'applicazione. Se non sono presenti altre strutture di intestazione IBM MQ , i dati del messaggio dell'applicazione vengono descritti da questi campi nella struttura MQMD all'interno della struttura MQXQH. Se la conversione ha esito positivo, i valori saranno gli stessi specificati nel parametro **MSGDSC** dall'applicazione che ha emesso la chiamata MQGET.

   Se il messaggio è un messaggio dell'elenco di distribuzione, la struttura MQXQH è seguita da una struttura MQDH (più i relativi array di record MQOR e MQPMR), che a sua volta può essere seguita da zero o più ulteriori strutture di intestazione IBM MQ e zero o più byte di dati del messaggio dell'applicazione. Come la struttura MQXQH, la struttura MQDH deve essere nella serie di caratteri e nella codifica del gestore code e non viene convertita nella chiamata MQGET, anche se è specificata l'opzione GMCONV.

   L'elaborazione delle strutture MQXQH e MQDH descritte in precedenza è principalmente destinata all'utilizzo da parte degli agent del canale dei messaggi quando ricevono i messaggi dalle code di trasmissione.

IBM i Conversione dei messaggi di report su IBM i

Un messaggio di report può contenere diverse quantità di dati del messaggio dell'applicazione, in base alle opzioni del report specificate dal mittente del messaggio originale.

In particolare, un messaggio di report può contenere:

1. Nessun dato del messaggio dell'applicazione
2. Alcuni dati del messaggio dell'applicazione dal messaggio originale

   Ciò si verifica quando il mittente del messaggio originale specifica RO* D e il messaggio è più lungo di 100 byte.
3. Tutti i dati del messaggio dell'applicazione dal messaggio originale

   Ciò si verifica quando il mittente del messaggio originale specifica RO* F o specifica RO* D e il messaggio è di 100 byte o più breve.

Quando il gestore code o l'agent del canale dei messaggi genera un messaggio di report, copia il nome del formato dal messaggio originale nel campo *MDFMT* nelle informazioni di controllo nel messaggio di report. Il nome del formato nel messaggio del report potrebbe quindi implicare una lunghezza di dati diversa dalla lunghezza presente nel messaggio del report (casi 1 e 2 descritti in precedenza).

Se l'opzione GMCONV viene specificata quando viene richiamato il messaggio di report:

- Per il caso 1 descritto in precedenza, l'uscita di conversione dati non verrà richiamata (poiché il messaggio di report non avrà dati).
- Per il caso 3 descritto precedentemente, il nome del formato implica correttamente la lunghezza dei dati del messaggio.
- Ma per il caso 2 descritto in precedenza, l'uscita di conversione dati verrà richiamata per convertire un messaggio che è *più breve* della lunghezza implicita dal nome del formato.

  Inoltre, il codice di errore inoltrato all'uscita generalmente sarà RCNONE (ossia, il codice di errore non indicherà che il messaggio è stato troncato). Ciò si verifica perché i dati del messaggio sono stati troncati dal *mittente* del messaggio di report e non dal gestore code del ricevente in risposta alla chiamata MQGET.

A causa di queste possibilità, l'uscita di conversione dati non deve utilizzare il nome del formato per dedurre la lunghezza dei dati ad essa passati; al contrario, l'uscita deve controllare la lunghezza dei dati forniti ed essere preparata a convertire meno dati della lunghezza implicita dal nome del formato. Se i dati possono essere convertiti correttamente, il codice di completamento CCOK e il codice motivo RCNONE devono essere restituiti dall'uscita. La lunghezza dei dati del messaggio da convertire viene passata all'exit come parametro **INLEN** .

Interfaccia di programmazione sensibile al prodotto

Se un messaggio di report contiene informazioni su un'attività che ha avuto luogo, è noto come report di attività. Esempi di attività sono:

- un MCA che invia un messaggio da una coda su un canale
- un MCA che riceve un messaggio da un canale e lo immette in coda
- un messaggio non recapitabile MCA che accoda un messaggio non recapitabile
- un MCA che riceve un messaggio da una coda e lo elimina
- un gestore di messaggi non instradabili che riposiziona un messaggio su una coda
- il server dei comandi che elabora una richiesta PCF - un broker che elabora una richiesta di pubblicazione
- un'applicazione utente che riceve un messaggio da una coda - un'applicazione utente che sfoglia un messaggio su una coda

Qualsiasi applicazione, incluso il gestore code, può aggiungere alcuni dati del messaggio al report di attività seguendo l'intestazione del report. La quantità di dati che devono essere forniti se alcuni vengono

inviati non è fissa e viene decisa dall'applicazione. Le informazioni restituite dovrebbero essere utili per l'applicazione che elabora il report di attività. I report di attività del gestore code restituiscono le strutture di intestazione IBM MQ standard (che iniziano con 'MQH') contenute nel messaggio originale. Ciò include, ad esempio, le intestazioni MQRFH2 incluse nel messaggio originale. Inoltre, il gestore code restituirà un'intestazione MQCFH trovata, ma non i parametri PCF associati ad essa. Ciò dà alle applicazioni di controllo un'idea di cosa fosse il messaggio.

IBM i MQDXP (parametro di uscita conversione dati) su IBM i

Blocco del parametro di uscita conversione dati.

Panoramica

Scopo: La struttura di MQDXP è un parametro che il gestore code passa all'uscita di conversione dati quando l'uscita viene richiamata per convertire i dati del messaggio come parte dell'elaborazione della chiamata MQGET. Consultare la descrizione della chiamata MQCONVX per dettagli sull'exit di conversione dati.

Serie di caratteri e codifica: i dati del carattere in MQDXP si trovano nella serie di caratteri del gestore code locale; ciò viene fornito dall'attributo del gestore code **CodedCharSetId** . I dati numerici in MQDXP sono nella codifica della macchina nativa; ciò viene fornito da ENNAT.

Utilizzo: l'uscita potrebbe modificare solo i campi *DXLEN*, *DXCC*, *DXREA* e *DXRES* in MQDXP; le modifiche agli altri campi vengono ignorate. Tuttavia, il campo *DXLEN* non può essere modificato se il messaggio che si sta convertendo è un segmento che contiene solo una parte di un messaggio logico.

Quando il controllo ritorna al gestore code dall'uscita, il gestore code verifica i valori restituiti in MQDXP. Se i valori restituiti non sono validi, il gestore code continua l'elaborazione come se l'exit avesse restituito XRFAIL in *DXRES* ; tuttavia, il gestore code ignora i valori dei campi *DXCC* e *DXREA* restituiti dall'exit in questo caso e utilizza invece i valori che tali campi avevano in *input* per l'exit. I seguenti valori in MQDXP causano questa elaborazione:

- Campo *DXRES* non XROK e non XRFAIL
- Campo *DXCC* non CCOK e non CCWARN
- Il campo *DXLEN* minore di zero oppure il campo *DXLEN* è stato modificato quando il messaggio da convertire è un segmento che contiene solo una parte di un messaggio logico.
- “Campi” a pagina 1484
- “Dichiarazione RPG (copia file CMQDXPH)” a pagina 1488

Campi

La struttura MQDXP contiene i seguenti campi; i campi sono descritti in ordine alfabetico :

DXAOP (numero intero con segno a 10 cifre)

Opzioni dell'applicazione.

Questa è una copia del campo *GMOPT* della struttura MQGMO specificata dall'applicazione che emette la chiamata MQGET. È possibile che l'uscita debba esaminarli per verificare se è stata specificata l'opzione GMATM.

Questo è un campo di immissione per l'uscita.

DXCC (numero intero con segno a 10 cifre)

Codice di completamento.

Quando viene richiamata l'exit, questo contiene il codice di completamento che verrà restituito all'applicazione che ha emesso la chiamata MQGET, se l'exit sceglie di non eseguire alcuna operazione. È sempre CCWARN, perché il messaggio è stato troncato oppure il messaggio richiede la conversione e questa operazione non è stata ancora eseguita.

In fase di output dall'uscita, questo campo contiene il codice di completamento da restituire all'applicazione nel parametro **CMPCOD** della chiamata MQGET; sono validi solo CCOK e CCWARN. Consultare la descrizione del campo *DXREA* per suggerimenti su come l'uscita deve impostare questo campo sull'output.

Questo è un campo di immissione / emissione per l'uscita.

DXCSI (numero intero con segno a 10 cifre)

Serie di caratteri richiesta dall'applicazione.

Questo è il CCSID (coded character set identifier) della serie di caratteri richiesta dall'applicazione che emette la chiamata MQGET; consultare il campo *MDCSI* nella struttura MQMD per ulteriori dettagli. Se l'applicazione specifica il valore speciale CSQM nella chiamata MQGET, il gestore code lo modifica con l'identificativo della serie di caratteri utilizzata dal gestore code, prima di richiamare l'uscita.

Se la conversione ha esito positivo, l'uscita deve copiarlo nel campo *MDCSI* nel descrittore del messaggio.

Questo è un campo di immissione per l'uscita.

DXENC (numero intero con segno a 10 cifre)

Codifica numerica richiesta dall'applicazione.

Si tratta della codifica numerica richiesta dall'applicazione che emette la chiamata MQGET; per ulteriori dettagli, consultare il campo *MDENC* nella struttura MQMD.

Se la conversione ha esito positivo, l'uscita deve copiarlo nel campo *MDENC* nel descrittore del messaggio.

Questo è un campo di immissione per l'uscita.

DXHCN (numero intero con segno a 10 cifre)

Handle di connessione.

Questo è un handle di connessione che può essere utilizzato sulla chiamata MQXCNVC. Questo handle non è necessariamente uguale a quello specificato dall'applicazione che ha emesso la chiamata MQGET.

DXLEN (numero intero con segno a 10 cifre)

Lunghezza in byte dei dati del messaggio.

Quando si richiama l'uscita, questo campo contiene la lunghezza originale dei dati del messaggio dell'applicazione. Se il messaggio è stato troncato per adattarsi al buffer fornito dall'applicazione, la dimensione del messaggio fornito all'exit sarà *minore* del valore di *DXLEN*. La dimensione del messaggio fornito all'uscita è sempre fornita dal parametro **INLEN** dell'uscita, indipendentemente da eventuali troncamenti.

Il troncamento è indicato dal campo *DXREA* con il valore RC2079 sull'input all'uscita.

La maggior parte delle conversioni non dovrà modificare questa lunghezza, ma un'uscita può farlo se necessario; il valore impostato dall'uscita viene restituito all'applicazione nel parametro **DATLEN** della chiamata MQGET. Tuttavia, questa lunghezza non può essere modificata se il messaggio che si sta convertendo è un segmento che contiene solo una parte di un messaggio logico. Ciò è dovuto al fatto che la modifica della lunghezza potrebbe causare l'errore degli offset dei segmenti successivi nel messaggio logico.

Tenere presente che, se l'uscita desidera modificare la lunghezza dei dati, tenere presente che il gestore code ha già deciso se i dati del messaggio rientreranno nel buffer dell'applicazione, in base alla lunghezza dei dati *non convertiti* . Questa decisione determina se il messaggio viene rimosso dalla coda (o il cursore di ricerca spostato, per una richiesta di ricerca) e non viene influenzato da alcuna modifica alla lunghezza dei dati causata dalla conversione. Per questo motivo si consiglia che le uscite di conversione non causino una modifica nella lunghezza dei dati del messaggio dell'applicazione.

Se la conversione dei caratteri implica una modifica della lunghezza, una stringa può essere convertita in un'altra stringa con la stessa lunghezza in byte, troncando gli spazi finali o riempiendo gli spazi come necessario.

L'uscita non viene richiamata se il messaggio non contiene dati del messaggio dell'applicazione; quindi *DXLEN* è sempre maggiore di zero.

Questo è un campo di immissione / emissione per l'uscita.

DXREA (numero intero con segno a 10 cifre)

Codice di errore *DXCC*.

Quando viene richiamata l'exit, questo contiene il codice motivo che verrà restituito all'applicazione che ha emesso la chiamata MQGET, se l'exit sceglie di non eseguire alcuna operazione. Tra i possibili valori sono RC2079, che indica che il messaggio è stato troncato per adattarsi al buffer fornito dall'applicazione, e RC2119, che indica che il messaggio richiede la conversione ma non è stato ancora eseguito.

All'output dall'uscita, questo campo contiene il motivo per cui deve essere restituito all'applicazione nel parametro **REASON** della chiamata MQGET; si consiglia di:

- Se *DXREA* aveva il valore RC2079 sull'input per l'uscita, i campi *DXREA* e *DXCC* non devono essere modificati, a prescindere dal fatto che la conversione abbia esito positivo o negativo.

  (Se il campo *DXCC* non è CCOK, l'applicazione che richiama il messaggio può identificare un errore di conversione confrontando i valori restituiti *MDENC* e *MDCSI* nel descrittore del messaggio con i valori richiesti; al contrario, l'applicazione non può distinguere un messaggio troncato da un messaggio che ha appena adattato il buffer. Per questo motivo, RC2079 deve essere restituito in preferenza a uno qualsiasi dei motivi che indicano un errore di conversione.)
- Se *DXREA* aveva un altro valore nell'input per l'uscita:
  - Se la conversione ha esito positivo, *DXCC* deve essere impostato su CCOK e *DXREA* su RCNONE.
  - Se la conversione ha esito negativo o se il messaggio si espande e deve essere troncato per adattarsi al buffer, *DXCC* deve essere impostato su CCWARN (o lasciato invariato) e *DXREA* deve essere impostato su uno dei valori i nel seguente elenco, per indicare la natura dell'errore.

    Notare che, se il messaggio dopo la conversione è troppo grande per il buffer, deve essere troncato solo se l'applicazione che ha emesso la chiamata MQGET ha specificato l'opzione GMATM:

    - Se ha specificato tale opzione, deve essere restituito il motivo RC2079 .
    - Se non è stata specificata tale opzione, il messaggio deve essere restituito non convertito, con codice di errore RC2120.

I codici di errore nel seguente elenco sono consigliati per l'utilizzo da parte dell'uscita per indicare il motivo per cui la conversione non è riuscita, ma l'uscita può restituire altri valori dalla serie di codici RC*, se ritenuto appropriato. Inoltre, l'intervallo di valori compresi tra RC0900 e RC0999 viene allocato per l'utilizzo da parte dell'exit per indicare le condizioni che l'exit desidera comunicare con l'applicazione che emette la chiamata MQGET.

Nota: Se il messaggio non può essere convertito correttamente, l'uscita deve restituire XRFAIL nel campo *DXRES* , in modo che il gestore code restituisca il messaggio non convertito. Ciò è vero indipendentemente dal codice di errore restituito nel campo *DXREA* .

RC0900
(900, X'384 ') Valore più basso per il codice di errore definito dall'applicazione.

RC0999
(999, X'3E7') Valore più alto per il codice di errore definito dall'applicazione.

RC2120
(2120, X'848 ') Dati convertiti troppo grandi per il buffer.

RC2119
(2119, X'847 ') Dati del messaggio non convertiti.

RC2111
(2111, X'83F') Identificativo serie di caratteri codificati origine non valido.

RC2113
(2113, X'841 ') Codifica decimale compresso nel messaggio non riconosciuta.

RC2114
(2114, X'842 ') La codifica a virgola mobile nel messaggio non è stata riconosciuta.

RC2112
(2112, X'840 ') Numero intero di origine non riconosciuto.

RC2115
(2115, X'843 ') Identificativo serie di caratteri codificati di destinazione non valido.

RC2117
(2117, X'845 ') La codifica decimale compresso specificata dal ricevitore non è stata riconosciuta.

RC2118
(2118, X'846 ') La codifica a virgola mobile specificata dal ricevitore non è stata riconosciuta.

RC2116
(2116, X'844 ') Codifica numero intero di destinazione non riconosciuta.

RC2079
(2079, X'81F') Messaggio troncato restituito (elaborazione completata).

Questo è un campo di immissione / emissione per l'uscita.

DXRES (numero intero con segno a 10 cifre)

Risposta dall'uscita.

Questo valore viene impostato dall'uscita per indicare l'esito positivo o meno della conversione. Deve essere uno dei seguenti:

XROK
Conversione riuscita correttamente.

Se l'uscita specifica questo valore, il gestore code restituisce quanto segue all'applicazione che ha emesso la chiamata MQGET:

- Il valore del campo *DXCC* sull'output dall'uscita
- Il valore del campo *DXREA* sull'output dall'uscita
- Il valore del campo *DXLEN* sull'output dall'uscita
- Il contenuto del buffer di output dell'exit *OUTBUF*. Il numero di byte restituiti è il minore del parametro **OUTLEN** dell'exit e il valore del campo *DXLEN* sull'output dall'exit

Se i campi *MDENC* e *MDCSI* nel parametro del descrittore del messaggio dell'uscita sono *entrambi* non modificati, il gestore code restituisce:

- Il valore dei campi *MDENC* e *MDCSI* nella struttura MQDXP in *input* per l'uscita

Se uno o entrambi i campi *MDENC* e *MDCSI* nel parametro del descrittore del messaggio dell'exit sono stati modificati, il gestore code restituisce:

- Il valore dei campi *MDENC* e *MDCSI* nel parametro del descrittore del messaggio dell'uscita sull'output dall'uscita
-

XRFAIL
Conversione non riuscita.

Se l'uscita specifica questo valore, il gestore code restituisce quanto segue all'applicazione che ha emesso la chiamata MQGET:

- Il valore del campo *DXCC* sull'output dall'uscita
- Il valore del campo *DXREA* sull'output dall'uscita

- Il valore del campo *DXLEN* in *input* per l'exit
- Il contenuto del buffer di input dell'uscita *INBUF*. Il numero di byte restituiti viene fornito dal parametro **INLEN**

Se l'uscita ha modificato *INBUF*, i risultati non sono definiti.

DXRES è un campo di output dall'uscita.

DXSID (stringa di caratteri a 4 byte)

Identificatore struttura.

Il valore deve essere:

DXSIDV
Identificativo per la struttura del parametro di uscita conversione dati.

Questo è un campo di immissione per l'uscita.

DXVER (numero intero con segno a 10 cifre)

Numero di versione della struttura.

Il valore deve essere:

DXVER1
Numero di versione per la struttura del parametro di uscita conversione dati.

La seguente costante specifica il numero di versione della versione corrente:

DXVERC
Versione corrente della struttura del parametro di uscita conversione dati.

Nota: Quando viene introdotta una nuova versione di questa struttura, il layout della parte esistente non viene modificato. L'uscita deve quindi controllare che il campo *DXVER* sia uguale o superiore alla versione più bassa che contiene i campi che l'uscita deve utilizzare.

Questo è un campo di immissione per l'uscita.

DXXOP (numero intero con segno a 10 cifre)

Riservato.

Questo è un campo riservato; il valore è `0`.

Dichiarazione RPG (copia file CMQDXPH)

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQDXP Structure
D*
D* Structure identifier
D  DXSID                  1      4
D* Structure version number
D  DXVER                  5      8I 0
D* Reserved
D  DXXOP                  9     12I 0
D* Application options
D  DXAOP                 13     16I 0
D* Numeric encoding required by application
D  DXENC                 17     20I 0
D* Character set required by application
D  DXCSI                 21     24I 0
D* Length in bytes of message data
D  DXLEN                 25     28I 0
D* Completion code
D  DXCC                  29     32I 0
D* Reason code qualifying DXCC
D  DXREA                 33     36I 0
D* Response from exit
D  DXRES                 37     40I 0
D* Connection handle
D  DXHCN                 41     44I 0
```

IBM i MQXCNVC (Converti caratteri) su IBM i

La chiamata MQXCNVC converte i caratteri da una serie di caratteri a un altro.

Questa chiamata fa parte di DCI (Data Conversion Interface) IBM MQ , che è una delle interfacce del framework IBM MQ . Nota: questa chiamata può essere utilizzata solo da un'uscita di conversione dati.

Sintassi

```
MQXCNVC(HCONN, OPTS, SRCCSI, SRCLEN, SRCBUF, TGTCSI, TGTLEN,
   TGTBUF, DATLEN, CMPCOD, REASON)
```

Parametri

La chiamata MQXCNVC ha i parametri seguenti:

HCONN (numero intero con segno a 10 cifre) - input

Handle di connessione.

Questo handle rappresenta la connessione al gestore code. Normalmente dovrebbe essere l'handle passato all'uscita di conversione dati nel campo DXHCN della struttura MQDXP; questo handle non è necessariamente uguale all'handle specificato dall'applicazione che ha emesso la chiamata MQGET.

In IBM i, è possibile specificare il seguente valore speciale per HCONN:

HCDEFH
Handle di connessione predefinito.

OPTS (numero intero con segno a 10 cifre) - immissione

Opzioni che controllano l'azione di MQXCNVC.

È possibile specificare zero o più delle opzioni descritte più avanti in questa sezione. Se ne è richiesta più di una, è possibile aggiungere i valori (non aggiungere la stessa costante più di una volta).

Opzione di conversione predefinita: la seguente opzione controlla l'utilizzo della conversione dei caratteri predefinita:

DCCDEF
Conversione predefinita.

Questa opzione specifica che è possibile utilizzare la conversione caratteri predefinita se una o entrambe le serie di caratteri specificate nella chiamata non sono supportate. Ciò consente al gestore code di utilizzare una serie di caratteri predefinita specificata dall'installazione che si avvicina alla serie di caratteri specificata, durante la conversione della stringa.

Nota: Il risultato dell'utilizzo di un set di caratteri approssimativo per convertire la stringa è che alcuni caratteri potrebbero essere convertiti in modo non corretto. Ciò può essere evitato utilizzando nella stringa solo i caratteri comuni sia alla serie di caratteri specificata che alla serie di caratteri predefinita.

I set di caratteri predefiniti vengono definiti da un'opzione di configurazione quando il gestore code viene installato o riavviato.

Se DCCDEF non viene specificato, il gestore code utilizza solo le serie di caratteri specificate per convertire la stringa e la chiamata ha esito negativo se una o entrambe le serie di caratteri non sono supportate.

Opzione di riempimento: la seguente opzione consente al gestore code di riempire la stringa convertita con spazi vuoti o eliminare caratteri finali non significativi, in modo che la stringa convertita si adatti al buffer di destinazione:

DCCFIL
Riempimento buffer di destinazione.

Questa opzione richiede che la conversione avvenga in modo tale che il buffer di destinazione venga riempito completamente:

- Se la stringa si contrae quando viene convertita, vengono aggiunti spazi finali per riempire il buffer di destinazione.
- Se la stringa si espande quando viene convertita, i caratteri finali non significativi vengono eliminati per rendere la stringa convertita adatta al buffer di destinazione. Se questa operazione può essere eseguita correttamente, la chiamata viene completata con CCOK e codice di errore RCNONE.

  Se il numero di caratteri finali non significativi è troppo basso, la maggior parte della stringa viene inserita nel buffer di destinazione e la chiamata viene completata con CCWARN e codice di errore RC2120.

  I caratteri non significativi sono:

  – Spazi vuoti finali
  – I caratteri che seguono il primo carattere null nella stringa (ma escludendo il primo carattere null stesso)
- Se la stringa, TGTCSI e TGTLEN sono tali che il buffer di destinazione non può essere impostato completamente con caratteri validi, la chiamata ha esito negativo con CCFAIL e codice di errore RC2144. Ciò può verificarsi quando TGTCSI è una serie di caratteri DBCS puri (come UTF-16), ma TGTLEN specifica una lunghezza che è un numero dispari di byte.
- TGTLEN può essere minore o maggiore di SRCLEN. Al ritorno da MQXCNVC, DATLEN ha lo stesso valore di TGTLEN.

Se questa opzione non viene specificata:

- La stringa è consentita per contrarre o espandere all'interno del buffer di destinazione come richiesto. I caratteri finali non significativi non vengono aggiunti o eliminati.

  Se la stringa convertita si adatta al buffer di destinazione, la chiamata viene completata con CCOK e codice di errore RCNONE.

  Se la stringa convertita è troppo grande per il buffer di destinazione, la maggior parte della stringa viene inserita nel buffer di destinazione e la chiamata viene completata con CCWARN e codice di errore RC2120. Notare che in questo caso possono essere restituiti meno di *TGTLEN* byte.
- TGTLEN può essere minore o maggiore di SRCLEN. Alla restituzione da MQXCNVC, DATLEN è minore o uguale a TGTLEN.

Opzioni di codifica: le seguenti opzioni possono essere utilizzate per specificare le codifiche intere delle stringhe di origine e di destinazione. La codifica rilevante viene utilizzata solo quando l'identificativo della serie di caratteri corrispondente indica che la rappresentazione della serie di caratteri nella memoria principale dipende dalla codifica utilizzata per i numeri interi binari. Ciò riguarda solo alcune serie di caratteri multibyte (ad esempio, le serie di caratteri UTF-16).

La codifica viene ignorata se la serie di caratteri è una serie di caratteri a byte singolo (SBCS) o una serie di caratteri a più byte con rappresentazione nella memoria principale che non dipende dalla codifica del numero intero.

È necessario specificare solo uno dei valori DCCS*, combinato con uno dei seguenti valori DCCT*:

DCCSNA
La codifica di origine è quella predefinita per l'ambiente e il linguaggio di programmazione.

DCCSNO
La codifica origine è normale.

DCCSRE
La codifica di origine è invertita.

DCCSUN
La codifica di origine non è definita.

DCCTNA
La codifica di destinazione è quella predefinita per l'ambiente e il linguaggio di programmazione.

NCCCC
La codifica di destinazione è normale.

DCCTRE
La codifica di destinazione è invertita.

DCCTUN
La codifica di destinazione non è definita.

I valori di codifica definiti precedentemente possono essere aggiunti direttamente al campo OPTS . Tuttavia, se la codifica di origine o di destinazione viene ottenuta dal campo MDENC in MQMD o in un'altra struttura, è necessario eseguire la seguente elaborazione:

1. La codifica di numeri interi deve essere estratta dal campo MDENC eliminando le codifiche a virgola mobile e decimale compresso; consultare “Analisi delle codifiche su IBM i” a pagina 1473 per dettagli su come eseguire questa operazione.
2. La codifica di numeri interi risultante dal passo 1 deve essere moltiplicata per il fattore appropriato prima di essere aggiunta al campo OPTS . Questi fattori sono:

   DCCSFA
   Fattore per la codifica di origine

   DCCTFA
   Fattore per la codifica di destinazione

Se non specificato, le opzioni di codifica vengono impostate per default su non definito (DCC* UN). Nella maggior parte dei casi, ciò non influisce sul corretto completamento della chiamata MQXCNVC. Tuttavia, se la serie di caratteri corrispondente è una serie di caratteri multibyte con rappresentazione dipendente dalla codifica (ad esempio, una serie di caratteri UTF-16), la chiamata ha esito negativo con il codice di errore RC2112 o RC2116 come appropriato.

Opzione predefinita: se nessuna delle opzioni descritte precedentemente viene specificata, è possibile utilizzare la seguente opzione:

DCCNON
Nessuna opzione specificata.

DCCNON è definito per aiutare la documentazione del programma. Non è previsto che questa opzione venga utilizzata con altre, ma poiché il suo valore è zero, tale utilizzo non può essere rilevato.

SRCCSI (numero intero con segno a 10 cifre) - input

CCSID (Coded character set identifier) della stringa prima della conversione.

Questo è il CCSID (coded character set identifier) della stringa di immissione in SRCBUF.

SRCLEN (numero intero con segno a 10 cifre) - immissione

Lunghezza della stringa prima della conversione.

Questa è la lunghezza in byte della stringa di input in SRCBUF ; deve essere zero o maggiore.

SRCBUF (stringa di caratteri a 1 byte x SRCLEN) - immissione

Stringa da convertire.

Questo è il buffer contenente la stringa da convertire da una serie di caratteri ad un'altra.

TGTCSI (numero intero con segno a 10 cifre) - input

CCSID (coded character set identifier) della stringa dopo la conversione.

Questo è il CCSID (coded character set identifier) della serie di caratteri in cui SRCBUF deve essere convertito.

TGTLEN (numero intero con segno a 10 cifre) - input

Lunghezza del buffer di output.

È la lunghezza in byte del buffer di output TGTBUF ; deve essere zero o maggiore. Può essere minore o maggiore di SRCLEN.

TGTBUF (stringa di caratteri a 1 byte x TGTLEN) - output

Stringa dopo la conversione.

Questa è la stringa dopo che è stata convertita nella serie di caratteri definita da TGTCSI. La stringa convertita può essere più corta o più lunga della stringa non convertita. Il parametro **DATLEN** indica il numero di byte validi restituiti.

DATLEN (numero intero con segno a 10 cifre) - emissione

Lunghezza della stringa di output.

Questa è la lunghezza della stringa restituita nel buffer di output TGTBUF. La stringa convertita può essere più corta o più lunga della stringa non convertita.

CMPCOD (numero intero con segno a 10 cifre) - output

Codice di completamento.

Il valore è uno dei seguenti:

CCOK
Completamento con esito positivo.

AVVCCN
Avvertenza (completamento parziale).

CCNON RIUSCITO
Chiamata fallita.

REASON (numero intero con segno a 10 cifre) - output

Codice di errore CMPCOD.

Se CMPCOD è CCOK:

RCNONE
(0, X'000 ') Nessun motivo per segnalare.

Se CMPCOD è CCWARN:

RC2120
(2120, X'848 ') Dati convertiti troppo grandi per il buffer.

Se CMPCOD è CCFAIL:

RC2010
(2010, X'7DA') Parametro di lunghezza dati non valido.

RC2150
(2150, X'866 ') Stringa DBCS non valida.

RC2018
(2018, X'7E2') Handle di connessione non valido.

RC2046
(2046, X'7FE') Opzioni non valide o non congruenti.

RC2102
(2102, X'836 ') Risorse di sistema insufficienti.

RC2145
(2145, X'861 ') Parametro del buffer di origine non valido.

RC2111
(2111, X'83F') Identificativo serie di caratteri codificati origine non valido.

RC2112
(2112, X'840 ') Numero intero di origine non riconosciuto.

RC2143
(2143, X'85F') Parametro di lunghezza origine non valido.

RC2071
(2071, X'817 ') Memoria disponibile insufficiente.

RC2146
(2146, X'862 ') Parametro buffer di destinazione non valido.

RC2115
(2115, X'843 ') Identificativo serie di caratteri codificati di destinazione non valido.

RC2116
(2116, X'844 ') Codifica numero intero di destinazione non riconosciuta.

RC2144
(2144, X'860 ') Parametro di lunghezza di destinazione non valido.

RC2195
(2195, X'893 ') Si è verificato un errore non previsto.

Per ulteriori informazioni su questi codici di errore, consultare “Codici di ritorno per IBM i (ILE RPG)” a pagina 1467.

Richiamo RPG (ILE)

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                    CALLP     MQXCNVC(HCONN : OPTS : SRCCSI :
C                                      SRCLEN : SRCBUF : TGTCSI :
C                                      TGTLEN : TGTBUF : DATLEN :
C                                      CMPCOD : REASON)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQXCNVC          PR                  EXTPROC('MQXCNVC')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Options that control the action of MQXCNVC
D OPTS                          10I 0 VALUE
D* Coded character set identifier of string before conversion
D SRCCSI                        10I 0 VALUE
D* Length of string before conversion
D SRCLEN                        10I 0 VALUE
D* String to be converted
D SRCBUF                          *   VALUE
D* Coded character set identifier of string after conversion
D TGTCSI                        10I 0 VALUE
D* Length of output buffer
D TGTLEN                        10I 0 VALUE
D* String after conversion
D TGTBUF                          *   VALUE
D* Length of output string
D DATLEN                        10I 0
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CMPCOD
D REASON                        10I 0
```

IBM i MQCONVX (Data conversion exit) su IBM i

Questa definizione di chiamata descrive i parametri passati all'uscita di conversione dati.

Nessun punto di ingresso denominato MQCONVX viene fornito dal gestore code (consultare la nota sull'uso “11” a pagina 1495).

Questa definizione fa parte di DCI (Data Conversion Interface) IBM MQ , che è una delle interfacce del framework IBM MQ .

Sintassi

MQCONVX *(MQDXP, MQMD, INLEN, INBUF, OUTLEN, OUTBUF)*

Note d'utilizzo

1. Un'uscita di conversione dati è un'uscita scritta dall'utente che riceve il controllo durante l'elaborazione di una chiamata MQGET. La funzione eseguita dall'uscita di conversione dati è definita dal provider dell'uscita; tuttavia, l'uscita deve essere conforme alle regole qui descritte e nella struttura di parametro associata MQDXP.

   I linguaggi di programmazione che possono essere utilizzati per un'uscita di conversione dati sono determinati dall'ambiente.
2. L'uscita viene richiamata solo se *tutte* le seguenti istruzioni sono vere:
   - L'opzione GMCONV è specificata nella chiamata MQGET
   - Il campo *MDFMT* nel descrittore del messaggio non è FMNONE
   - Il messaggio non è già nella rappresentazione richiesta; ovvero, uno o entrambi i *MDCSI* e *MDENC* del messaggio sono diversi dal valore specificato dall'applicazione nel descrittore del messaggio fornito nella chiamata MQGET
   - Il gestore code non ha ancora eseguito correttamente la conversione
   - La lunghezza del buffer dell'applicazione è maggiore di zero
   - La lunghezza dei dati del messaggio è maggiore di zero
   - Il codice di errore fino ad ora durante l'operazione MQGET è RCNONE o RC2079
3. Quando si sta scrivendo un'uscita, è necessario considerare la codifica dell'uscita in un modo che gli consenta di convertire i messaggi che sono stati troncati. I messaggi troncati possono verificarsi nei modi seguenti:
   - L'applicazione ricevente fornisce un buffer più piccolo del messaggio, ma specifica l'opzione GMATM sulla chiamata MQGET.

     In tal caso, il campo *DXREA* nel parametro **MQDXP** sull'input per l'exit avrà il valore RC2079.
   - Il mittente del messaggio lo ha troncato prima di inviarlo. Ciò può verificarsi con i messaggi di report, ad esempio (consultare “Conversione dei messaggi di report su IBM i” a pagina 1483 per ulteriori dettagli).

     In questo caso, il campo *DXREA* nel parametro **MQDXP** sull'input per l'uscita avrà il valore RCNONE (se l'applicazione ricevente ha fornito un buffer sufficientemente grande per il messaggio).

   Pertanto, il valore del campo *DXREA* sull'input per l'uscita non può essere sempre utilizzato per decidere se il messaggio è stato troncato.

   La caratteristica distintiva di un messaggio troncato è che la lunghezza fornita all'uscita nel parametro **INLEN** sarà *inferiore* alla lunghezza implicita dal nome formato contenuto nel campo

MDFMT nel descrittore del messaggio. L'uscita deve quindi controllare il valore di *INLEN* prima di tentare la conversione dei dati; l'uscita *non* deve presumere che sia stata fornita l'intera quantità di dati implicita dal nome del formato.

Se l'uscita non è stata scritta per convertire i messaggi troncati e **INLEN** è inferiore al valore previsto, l'uscita deve restituire XRFAIL nel campo *DXRES* del parametro **MQDXP** , con il campo *DXCC* impostato su CCWARN e il campo *DXREA* impostato su RC2110.

Se l'uscita *è stata scritta* per convertire i messaggi troncati, l'uscita deve convertire la maggior quantità possibile di dati (consultare la nota sull'utilizzo successiva), facendo attenzione a non tentare di esaminare o convertire i dati oltre la fine di *INBUF*. Se la conversione viene completata correttamente, l'uscita deve lasciare invariato il campo *DXREA* nel parametro **MQDXP** . Restituisce RC2079 se il messaggio è stato troncato dal gestore code del destinatario e RCNONE se il messaggio è stato troncato dal mittente del messaggio.

È anche possibile che un messaggio espanda *durante la conversione* , al punto in cui è più grande di *OUTBUF*. In questo caso l'uscita deve decidere se troncare il messaggio; il campo *DXAOP* nel parametro **MQDXP** indicherà se l'applicazione ricevente ha specificato l'opzione GMATM.

4. Generalmente, si consiglia di convertire tutti i dati nel messaggio fornito all'uscita in *INBUF* o che nessuno di essi sia convertito. Un'eccezione a questo, tuttavia, si verifica se il messaggio viene troncato, prima della conversione o durante la conversione; in questo caso, potrebbe esserci un elemento incompleto alla fine del buffer (ad esempio: un byte di un carattere a doppio byte o 3 byte di un numero intero a 4 byte). In questa situazione, si consiglia di omettere l'elemento incompleto e di impostare i byte inutilizzati in *OUTBUF* su null. Tuttavia, gli elementi completi o i caratteri all'interno di un array o di una stringa *devono* essere convertiti.
5. Quando un'uscita è necessaria per la prima volta, il gestore code tenta di caricare un oggetto con lo stesso nome del formato (a parte le estensioni). L'oggetto caricato deve contenere l'uscita che elabora i messaggi con tale nome formato. Si consiglia che il nome dell'uscita e il nome dell'oggetto che contiene l'uscita siano identici, sebbene non tutti gli ambienti lo richiedano.
6. Una nuova copia dell'uscita viene caricata quando un'applicazione tenta di recuperare il primo messaggio che utilizza *MDFMT* da quando l'applicazione si è connessa al gestore code. Una nuova copia potrebbe essere caricata anche in altre occasioni, se il gestore code ha eliminato una copia precedentemente caricata. Per questo motivo, un'uscita non deve tentare di utilizzare la memoria statica per comunicare le informazioni da un richiamo dell'uscita al successivo - l'uscita può essere scaricata tra i due richiami.
7. Se è presente un'uscita fornita dall'utente con lo stesso nome di uno dei formati integrati supportati dal gestore code, l'uscita fornita dall'utente non sostituisce la routine di conversione integrata. Le uniche circostanze in cui tale uscita viene richiamata sono:
   - Se la routine di conversione integrata non è in grado di gestire le conversioni verso o da *MDCSI* o *MDENC* coinvolti, oppure
   - Se la routine di conversione incorporata non è riuscita a convertire i dati (ad esempio, perché esiste un campo o un carattere che non può essere convertito).
8. L'ambito dell'uscita dipende dall'ambiente. I nomi *MDFMT* devono essere scelti in modo da ridurre il rischio di conflitti con altri formati. Si consiglia di iniziare con caratteri che identificano l'applicazione che definisce il nome del formato.
9. L'uscita di conversione dati viene eseguita in un ambiente simile a quello del programma che ha emesso la chiamata MQGET; l'ambiente include lo spazio di indirizzo e il profilo utente (dove applicabile). Il programma potrebbe essere un agente del canale dei messaggi che invia messaggi a un gestore code di destinazione che non supporta la conversione dei messaggi. L'uscita non può compromettere l'integrità del gestore code, poiché non viene eseguita nell'ambiente del gestore code.
10. L'unica chiamata MQI che può essere utilizzata dall'uscita è MQXCNVC; il tentativo di utilizzare altre chiamate MQI ha esito negativo con codice motivo RC2219o altri errori imprevedibili.
11. Nessun punto di entrata denominato MQCONVX viene fornito dal gestore code. Il nome dell'uscita deve essere uguale al nome del formato (il nome contenuto nel campo *MDFMT* in MQMD), sebbene non sia richiesto in tutti gli ambienti.

Parametri

La chiamata MQCONVX presenta i parametri seguenti:

MQDXP (MQDXP) - input/output

Blocco del parametro di uscita conversione dati.

Questa struttura contiene informazioni relative al richiamo dell'exit. L'uscita imposta le informazioni in questa struttura per indicare il risultato della conversione. Consultare "MQDXP (parametro di uscita conversione dati) su IBM i" a pagina 1484 per i dettagli dei campi in questa struttura.

MQMD (MQMD) - input / output

Descrittore del messaggio.

Nell'input per l'uscita, questo è il descrittore del messaggio che viene restituito all'applicazione se non viene eseguita alcuna conversione. Pertanto, contiene *MDFMT*, *MDENC*e *MDCSI* del messaggio non convertito contenuto in *INBUF*.

Nota: Il parametro **MQMD** inoltrato all'exit è sempre la versione più recente di MQMD supportata dal gestore code che richiama l'exit. Se l'uscita deve essere trasportabile tra ambienti differenti, l'uscita deve controllare il campo *MDVER* in *MQMD* per verificare che i campi a cui l'uscita deve accedere siano presenti nella struttura.

Su IBM i, l'uscita viene passata a version-2 MQMD.

In fase di output, l'uscita deve modificare i campi *MDENC* e *MDCSI* nei valori richiesti dall'applicazione, se la conversione ha avuto esito positivo; queste modifiche si rifletteranno nuovamente sull'applicazione. Tutte le altre modifiche apportate dall'exit alla struttura vengono ignorate; non vengono riportate nell'applicazione.

Se l'uscita restituisce XROK nel campo *DXRES* della struttura MQDXP, ma non modifica i campi *MDENC* o *MDCSI* nel descrittore del messaggio, il gestore code restituisce per tali campi i valori che i campi corrispondenti nella struttura MQDXP avevano nell'input dell'uscita.

INLEN (numero intero con segno a 10 cifre) - input

Lunghezza in byte di *INBUF*.

Si tratta della lunghezza del buffer di input *INBUF*e specifica il numero di byte che devono essere elaborati dall'uscita. *INLEN* è la lunghezza minore dei dati del messaggio prima della conversione e la lunghezza del buffer fornito dall'applicazione sulla chiamata MQGET.

Il valore è sempre maggiore di zero.

INBUF (stringa bit a 1 byte x INLEN) - input

Buffer contenente il messaggio non convertito.

Contiene i dati del messaggio prima della conversione. Se l'uscita non è in grado di convertire i dati, il gestore code restituisce il contenuto di questo buffer all'applicazione dopo che l'uscita è stata completata.

Nota: L'uscita non deve modificare *INBUF* ; se questo parametro viene modificato, i risultati non sono definiti.

OUTLEN (numero intero con segno a 10 cifre) - immissione

Lunghezza in byte di *OUTBUF*.

Questa è la lunghezza del buffer di output *OUTBUF*ed è uguale alla lunghezza del buffer fornito dall'applicazione sulla chiamata MQGET.

Il valore è sempre maggiore di zero.

OUTBUF (stringa bit a 1 byte x OUTLEN) - output

Buffer contenente il messaggio convertito.

All'output dell'uscita, se la conversione ha avuto esito positivo (come indicato dal valore XROK nel campo *DXRES* del parametro **MQDXP**), **OUTBUF** contiene i dati del messaggio da consegnare all'applicazione, nella rappresentazione richiesta. Se la conversione ha avuto esito negativo, tutte le modifiche che l'uscita ha apportato a questo buffer vengono ignorate.

Richiamo RPG (ILE)

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                   CALLP     exitname(MQDXP : MQMD : INLEN :
C                                      INBUF : OUTLEN : OUTBUF)
```

La definizione del prototipo per la chiamata è:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
Dexitname         PR                  EXTPROC('exitname')
D* Data-conversion exit parameter block
D MQDXP                          44A
D* Message descriptor
D MQMD                          364A
D* Length in bytes of INBUF
D INLEN                          10I 0 VALUE
D* Buffer containing the unconverted message
D INBUF                           *   VALUE
D* Length in bytes of OUTBUF
D OUTLEN                         10I 0 VALUE
D* Buffer containing the converted message
D OUTBUF                          *   VALUE
```

Fine dell'interfaccia di programmazione sensibile al prodotto

Riferimenti alle uscite utente, alle uscite API e ai servizi installabili

Utilizzare le linformation in questa sezione per sviluppare le uscite utente, le uscite API e le applicazioni dei servizi installabili:

- “Struttura MQIEP” a pagina 1497
- “Riferimento uscita conversione dati” a pagina 1501
- “MQ_PUBLISH_EXIT - Uscita pubblicazione” a pagina 1505
- “Chiamate di uscita canale e strutture dati” a pagina 1513
- “Riferimento uscita API” a pagina 1604
- “Informazioni di riferimento per l'interfaccia dei servizi installabili” a pagina 1665

Concetti correlati

Uscite utente, uscite API e servizi installabili IBM MQ

Attività correlate

Estensione delle funzioni del gestore code

Struttura MQIEP

La struttura MQIEP contiene un punto di immissione per ogni chiamata di funzione che le uscite possono effettuare.

Campi

StrucId

Tipo: MQCHAR4 - input

Identificatore struttura. Il valore è il seguente:

ID_STRUC_MQIEP

Versione
Tipo: MQLONG - input

Numero di versione della struttura. Il valore è il seguente:

MQIEP_VERSION_1
Numero di versione della struttura versione 1.

VERSIONE MQIEP_CURRENT_
Versione corrente della struttura.

StrucLength
Tipo: MQLONG

Dimensione della struttura MQIEP in byte. Il valore è il seguente:

MQIEP_LENGTH_1

Indicatori
Tipo: MQLONG

Fornisce informazioni sugli indirizzi della funzione. Un indicatore per indicare se la libreria è con thread può essere utilizzato con un indicatore per indicare se la libreria è una libreria client o server.

Il seguente valore viene utilizzato per non specificare alcuna informazione sulla libreria:

MQIEPF_NONE

Uno dei seguenti valori viene utilizzato per specificare se la libreria condivisa è con o senza thread:

MQIEPF_NON_THREADED_LIBRARY
Una libreria condivisa senza thread

Libreria MQIEPF_THREADED_LIBRARY
Una libreria condivisa con thread

Uno dei seguenti valori viene utilizzato per specificare se la libreria condivisa è una libreria condivisa del client o del server:

LIBRERIA_CLIENT_MQIEPF_
Una libreria condivisa del client

MQIEPF_LOCAL_LIBRARY
Una libreria condivisa del server

Riservato
Tipo: MQPTR

Chiamata MQBACK_Call
Tipo: PMQ_BACK_CALL

Indirizzo della chiamata MQBACK.

Richiama_MQBEGIN
Tipo: PMQ_BEGIN_CALL

Indirizzo della chiamata MQBEGIN.

Chiamata MQBUFMH
Tipo: PMQ_BUFMH_CALL

Indirizzo della chiamata MQBUFMH.

Richiama_MQCB
Tipo: PMQ_CB_CALL

Indirizzo della chiamata MQCB.

Richiama_MQCLOSE
Tipo: PMQ_CLOSE_CALL

Indirizzo della chiamata MQCLOSE.

Richiama_MQCMIT
Tipo: PMQ_CMIT_CALL

Indirizzo della chiamata MQCMIT.

Chiamata MQCONN_
Tipo: PMQ_CONN_CALL

Indirizzo della chiamata MQCONN.

Richiama_MQCONNX
Tipo: PMQ_CONNX_CALL

Indirizzo della chiamata MQCONNX.

Richiama_MQCRTMH_Call
Tipo: PMQ_CRTMH_CALL

Indirizzo della chiamata MQCRTMH.

Richiama_MQCTL
Tipo: PMQ_CTL_CALL

Indirizzo della chiamata MQCTL.

Chiamata MQDISC
Tipo: PMQ_DISC_CALL

Indirizzo della chiamata MQDISC.

Richiama_MQDLTMH_Call
Tipo: PMQ_DLTMH_CALL

Indirizzo della chiamata MQDLTMH.

Chiamata MQDLTMP
Tipo: PMQ_DLTMP_CALL

Indirizzo della chiamata MQDLTMP.

Richiama_MQGET
Tipo: PMQ_GET_CALL

Indirizzo della chiamata MQGET.

Richiama_MQINQ
Tipo: PMQ_INQ_CALL

Indirizzo della chiamata MQINQ.

Richiama_MQINQMP
Tipo: PMQ_INQMP_CALL

Indirizzo della chiamata MQINQMP.

Chiamata MQMHBUF_Call
Tipo: PMQ_MHBUF_CALL

Indirizzo della chiamata MQMHBUF.

Chiamata MQOPEN
Tipo: PMQ_OPEN_CALL

Indirizzo della chiamata MQOPEN.

Richiama_MQPUT
Tipo: PMQ_PUT_CALL

Indirizzo della chiamata MQPUT.

MQPUT1_Call
Tipo: PMQ_PUT1_CALL

Indirizzo della chiamata MQPUT1 .

Richiama_MQSET
Tipo: PMQ_SET_CALL

Indirizzo della chiamata MQSET.

Richiama_MQSETMP
Tipo: PMQ_SETMP_CALL

Indirizzo della chiamata MQSETMP.

MQSTAT_Chiamata
Tipo: PMQ_STAT_CALL

Indirizzo della chiamata MQSTAT.

MQSUB_Chiamata
Tipo: PMQ_SUB_CALL

Indirizzo della chiamata MQSUB.

Richiama_MQSUBRQ
Tipo: PMQ_SUBRQ_CALL

Indirizzo della chiamata MQSUBRQ.

Chiamata MQXCNVC_
Tipo: PMQ_XCNVC_CALL

Indirizzo della chiamata MQXCNVC.

Chiamata MQXCLWLN
Tipo: PMQ_XCLWLN_CALL

Indirizzo della chiamata MQXCLWLN.

Richiama_MQXDX
Tipo: PMQ_XDX_CALL

Indirizzo della chiamata MQXDX.

Richiama_MQXEP
Tipo: PMQ_XEP_CALL

Indirizzo della chiamata MQXEP.

Chiamata MQZEP
Tipo: PMQ_ZEP_CALL

Indirizzo della chiamata MQZEP.

Dichiarazione C

```
struct tagMQIEP {
   MQCHAR4          StrucId;          /* Structure identifier */
   MQLONG           Version;          /* Structure version number */
   MQLONG           StrucLength;      /* Structure length */
   MQLONG           Flags;            /* Flags */
   MQPTR            Reserved;         /* Reserved */
   PMQ_BACK_CALL    MQBACK_Call;      /* Address of MQBACK */
   PMQ_BEGIN_CALL   MQBEGIN_Call;     /* Address of MQBEGIN */
   PMQ_BUFMH_CALL   MQBUFMH_Call;     /* Address of MQBUFMH */
   PMQ_CB_CALL      MQCB_Call;        /* Address of MQCB */
   PMQ_CLOSE_CALL   MQCLOSE_Call;     /* Address of MQCLOSE */
   PMQ_CMIT_CALL    MQCMIT_Call;      /* Address of MQCMIT */
   PMQ_CONN_CALL    MQCONN_Call;      /* Address of MQCONN */
   PMQ_CONNX_CALL   MQCONNX_Call;     /* Address of MQCONNX */
   PMQ_CRTMH_CALL   MQCRTMH_Call;     /* Address of MQCRTMH */
   PMQ_CTL_CALL     MQCTL_Call;       /* Address of MQCTL */
   PMQ_DISC_CALL    MQDISC_Call;      /* Address of MQDISC */
   PMQ_DLTMH_CALL   MQDLTMH_Call;     /* Address of MQDLTMH */
```

```
    PMQ_DLTMP_CALL   MQDLTMP_Call;     /* Address of MQDLTMP */
    PMQ_GET_CALL     MQGET_Call;       /* Address of MQGET */
    PMQ_INQ_CALL     MQINQ_Call;       /* Address of MQINQ */
    PMQ_INQMP_CALL   MQINQMP_Call;     /* Address of MQINQMP */
    PMQ_MHBUF_CALL   MQMHBUF_Call;     /* Address of MQMHBUF */
    PMQ_OPEN_CALL    MQOPEN_Call;      /* Address of MQOPEN */
    PMQ_PUT_CALL     MQPUT_Call;       /* Address of MQPUT */
    PMQ_PUT1_CALL    MQPUT1_Call;      /* Address of MQPUT1 */
    PMQ_SET_CALL     MQSET_Call;       /* Address of MQSET */
    PMQ_SETMP_CALL   MQSETMP_Call;     /* Address of MQSETMP */
    PMQ_STAT_CALL    MQSTAT_Call;      /* Address of MQSTAT */
    PMQ_SUB_CALL     MQSUB_Call;       /* Address of MQSUB */
    PMQ_SUBRQ_CALL   MQSUBRQ_Call;     /* Address of MQSUBRQ */
    PMQ_XCLWLN_CALL  MQXCLWLN_Call;    /* Address of MQXCLWLN */
    PMQ_XCNVC_CALL   MQXCNVC_Call;     /* Address of MQXCNVC */
    PMQ_XDX_CALL     MQXDX_Call;       /* Address of MQXDX */
    PMQ_XEP_CALL     MQXEP_Call;       /* Address of MQXEP */
    PMQ_ZEP_CALL     MQZEP_Call;       /* Address of MQZEP */
};
```

Riferimento uscita conversione dati

Per z/OS, è necessario scrivere uscite di conversione dati in linguaggio assembler. Per altre piattaforme, si consiglia di utilizzare il linguaggio di programmazione C.

Per facilitare la creazione di un programma di uscita di conversione dati, vengono fornite le seguenti risorse:

- Un file di origine skeleton
- Una chiamata di conversione caratteri
- Un programma di utilità che crea un frammento di codice che esegue la conversione dei dati sulle strutture tipo di dati. Questo programma di utilità utilizza solo l'input C. Su z/OS, produce il codice assembler.

Per la procedura di scrittura dei programmi si veda:

- IBM i Scrittura di un programma di uscita conversione dati per IBM MQ for IBM i
- z/OS Scrittura di un programma di uscita conversione dati per IBM MQ for z/OS
- Scrittura di un'uscita di conversione dati per sistemi IBM MQ for AIX or Linux
- Scrittura di un'uscita di conversione dati per IBM MQ for Windows

File di origine della struttura

Questi possono essere utilizzati come punto di inizio quando si scrive un programma di uscita di conversione dati.

I file forniti sono elencati in Tabella 816 a pagina 1501.

Tabella 816. File di origine skeleton

Piattaforma	**File**
AIX AIX	amqsvfc0.c
IBM i IBM i	QMQMSAMP/QCSRC (AMQSVFC4)
Linux Linux	amqsvfc0.c
Sistemi Windows Windows	amqsvfc0.c

Tabella 816. File di origine skeleton (Continua)	
Piattaforma	**File**
z/OS z/OS	CSQ4BAX8 (“1” a pagina 1502) CSQ4BAX9 (“2” a pagina 1502) CSQ4CAX9 (“3” a pagina 1502)
Note: 1. Illustra la chiamata MQXCVNC. 2. Un wrapper per i frammenti di codice generati dal programma di utilità da utilizzare in tutti gli ambienti tranne CICS. 3. Un wrapper per i frammenti di codici generati dal programma di utilità da utilizzare nell'ambiente CICS .	

Converti chiamata caratteri

Utilizzare la chiamata MQXCNVC (conversione caratteri) dall'interno di un programma di uscita di conversione dati per convertire i dati del messaggio carattere da una serie di caratteri ad un altro. Per alcune serie di caratteri multibyte (ad esempio, le serie di caratteri UTF-16), è necessario utilizzare le opzioni appropriate.

Non è possibile effettuare altre chiamate MQI dall'interno dell'uscita; un tentativo di effettuare tale chiamata ha esito negativo con codice motivo MQRC_CALL_IN_PROGRESS.

Consultare “MQXCNVC - Converti caratteri” a pagina 945 per ulteriori informazioni sulla chiamata MQXCNVC e sulle opzioni appropriate.

Programma di utilità per la creazione del codice di uscita di conversione

Utilizzare queste informazioni per ulteriori informazioni sulla creazione del codice di uscita di conversione.

I comandi per la creazione del codice di uscita di conversione sono:

IBM i **IBM i**
CVTMQMDTA (Conversione del tipo di dati IBM MQ)

ALW **Sistemi AIX, Linux, and Windows**
`crtmqcvx` (Crea IBM MQ exit di conversione)

z/OS **z/OS**
CSQUCVX

Il comando per la piattaforma produce un frammento di codice che esegue la conversione dei dati sulle strutture del tipo di dati, da utilizzare nel programma di uscita conversione dati. Il comando prende un file che contiene una o più definizioni della struttura del linguaggio C. z/OS In z/OS, genera un dataset contenente i frammenti di codice assembler e le funzioni di conversione. Su altre piattaforme, genera un file con una funzione C per convertire ogni definizione di struttura. Su z/OS, il programma di utilità richiede l'accesso alla libreria di runtime LE/370 SCEERUN.

Richiamo del programma di utilità CSQUCVX su z/OS

z/OS

Figura 10 a pagina 1503 mostra un esempio di JCL utilizzato per richiamare il programma di utilità CSQUCVX.

```
//CVX    EXEC PGM=CSQUCVX
//STEPLIB  DD DISP=SHR,DSN=thlqual.SCSQANLE
//         DD DISP=SHR,DSN=thlqual.SCSQLOAD
//         DD DISP=SHR,DSN=le370qual.SCEERUN
//SYSPRINT DD SYSOUT=*
//CSQUINP  DD DISP=SHR,DSN=MY.MQSERIES.FORMATS(MSG1)
//CSQUOUT  DD DISP=OLD,DSN=MY.MQSERIES.EXITS(MSG1)
```

Figura 10. JCL di esempio utilizzato per richiamare il programma di utilità CSQUCVX

Istruzioni di definizione dati z/OS

z/OS

Il programma di utilità CSQUCVX richiede istruzioni DD con i seguenti nomi DD mostrati in Tabella 817 a pagina 1503:

Tabella 817. Nomi e descrizioni delle istruzioni di definizione dati

Istruzione DD	Descrizione
SYSPRINT	Specifica un dataset o una classe di spool di stampa per i report e i messaggi di errore.
CSQUINP	Specifica il dataset partizionato contenente le definizioni delle strutture dati da convertire.
CSQUOUT	Specifica il dataset partizionato in cui devono essere scritti i frammenti del codice di conversione. La lunghezza record logica (LRECL) deve essere 80 e il formato record (RECFM) deve essere FB.

Messaggi di errore nei sistemi AIX, Linux, and Windows

Il comando `crtmqcvx` restituisce i messaggi compresi tra AMQ7953 e AMQ7970.

Questi messaggi sono elencati in Messaggi e codici di errore *IBM MQ Messaggi*.

Esistono due tipi principali di errore:

- Gli errori principali, come gli errori di sintassi, quando l'elaborazione non può continuare.

  Viene visualizzato un messaggio sullo schermo che indica il numero di riga dell'errore nel file di input. Il file di output potrebbe essere stato creato parzialmente.
- Altri errori quando viene visualizzato un messaggio che indica che è stato rilevato un problema ma che l'analisi della struttura può continuare.

  Il file di output è stato creato e contiene informazioni di errore relative ai problemi che si sono verificati. Queste informazioni di errore sono precedute da `#error` in modo che il codice prodotto non venga accettato da alcun compilatore senza intervento per risolvere i problemi.

Sintassi valida

Il file di immissione per il programma di utilità deve essere conforme alla sintassi del linguaggio C.

Se non si ha familiarità con C, fare riferimento all' esempio C in questo argomento.

Inoltre, tenere presenti le seguenti regole:

- typedef viene riconosciuto solo prima della parola chiave struct.
- È richiesta una tag di struttura nelle dichiarazioni di struttura.
- È possibile utilizzare le parentesi quadre vuote [] per indicare una stringa o un array di lunghezza variabile alla fine di un messaggio.
- Gli array multidimensionali e gli array di stringhe non sono supportati.

- Vengono riconosciuti i seguenti tipi di dati aggiuntivi:
  - MQBOOL
  - MQBYTE
  - MQCAR
  - MQFLOAT32
  - MQFLOAT64
  - MQSHORT
  - MQLONG
  - MQINT8
  - MQUINT8
  - MQINT16
  - MQUINT16
  - MQINT32
  - MQUINT32
  - MQINT64
  - MQUINT64

  I campi MQCHAR sono convertiti in codepage, ma MQBYTE, MQINT8 e MQUINT8 non vengono toccati. Se la codifica è diversa, MQSHORT, MQLONG, MQINT16, MQUINT16, MQINT32, MQUINT32, MQINT64, MQUINT64, MQFLOAT32, MQFLOAT64 e MQBOOL vengono convertiti di conseguenza.
- Non utilizzare i seguenti tipi di dati:
  - doppio
  - puntatori
  - campi bit

  Ciò è dovuto al fatto che il programma di utilità per la creazione del codice di uscita di conversione non fornisce la funzione per convertire questi tipi di dati. Per superare questo, è possibile scrivere le proprie routine e richiamarle dall'uscita.

Altri punti da notare:

- Non utilizzare i numeri di sequenza nel dataset di input.
- Se ci sono campi per i quali si desidera fornire le proprie routine di conversione, dichiararli come MQBYTE e quindi sostituire le macro CMQXCFBA generate con il proprio codice di conversione.

Esempio C

```
struct TEST { MQLONG    SERIAL_NUMBER;
              MQCHAR    ID[5];
              MQINT16   VERSION;
              MQBYTE    CODE[4];
              MQLONG    DIMENSIONS[3];
              MQCHAR    NAME[24];
            } ;
```

Ciò corrisponde alle seguenti dichiarazioni in altri linguaggi di programmazione:

COBOL

```
      10 TEST.
       15 SERIAL-NUMBER  PIC S9(9) BINARY.
       15 ID             PIC X(5).
       15 VERSION        PIC S9(4) BINARY.
     * CODE IS NOT TO BE CONVERTED
```

```
        15 CODE           PIC X(4).
        15 DIMENSIONS     PIC S9(9) BINARY OCCURS 3 TIMES.
        15 NAME           PIC X(24).
```

System/390

```
TEST           EQU *
SERIAL_NUMBER  DS  F
ID             DS  CL5
VERSION        DS  H
CODE           DS  XL4
DIMENSIONS     DS  3F
NAME           DS  CL24
```

PL/I

Supportato solo su z/OS

```
 DCL 1 TEST,
       2 SERIAL_NUMBER  FIXED BIN(31),
       2 ID             CHAR(5),
       2 VERSION        FIXED BIN(15),
       2 CODE           CHAR(4),        /* not to be converted */
       2 DIMENSIONS(3)  FIXED BIN(31),
       2 NAME           CHAR(24);
```

MQ_PUBLISH_EXIT - Uscita pubblicazione

La chiamata MQ_PUBLISH_EXIT può esaminare e modificare i messaggi consegnati ai sottoscrittori.

Finalità

Utilizzare l'uscita di pubblicazione per esaminare e modificare i messaggi consegnati ai sottoscrittori:

- Esaminare il contenuto di un messaggio pubblicato per ciascun sottoscrittore
- Modificare il contenuto di un messaggio pubblicato per ciascun sottoscrittore
- Modificare la coda in cui viene inserito un messaggio
- Arrestare la consegna di un messaggio a un sottoscrittore

Questa uscita non è disponibile su IBM MQ for z/OS.

Sintassi

MQ_PUBLISH_EXIT (*ExitParms*, *PubContext*, *SubContext*)

Parametri

***ExitParms* (MQPSXP)- Input/Output**
ExitParms contiene informazioni sul richiamo dell'exit.

***PubContext* (MQPBC) - Input**
PubContext contiene informazioni contestuali sul publisher della pubblicazione.

***SubContext* (MQSBC) - Input/Output**
SubContext contiene informazioni contestuali sul sottoscrittore che riceve la pubblicazione.

MQPSXP - Struttura dei dati di uscita di pubblicazione

La struttura MQPSXP descrive le informazioni trasmesse e restituite dall'uscita di pubblicazione.

Tabella 818 a pagina 1506 riepiloga i campi nella struttura:

Tabella 818. Campi in MQPSXP	
Campo	**Descrizione**
StrucID	Identificativo struttura
Version	Numero di versione della struttura
ExitId	Tipo di uscita che viene richiamata
ExitReason	Motivo del richiamo dell'exit
ExitResponse	Risposta dall'uscita
ExitResponse2	Risposta secondaria dall'uscita
Feedback	Codice feedback
ExitUserArea	Esci dall'area utente
ExitData	Dati uscita
QMgrName	Nome del gestore code locale
Hconn	ID interno connessioni
MsgDescPtr	Indirizzo del descrittore del messaggio (MQMD)
MsgHandle	Gestione delle proprietà dei messaggi (MQHMSG)
MsgInPtr	Indirizzo del messaggio di input
MsgInLength	Lunghezza del messaggio di input
MsgOutPtr	Indirizzo del messaggio di output
MsgOutLength	Lunghezza del messaggio di output
pEntryPoints	Indirizzo della struttura MQIEP

Campi

***StrucID* (MQCHAR4)**

StrucID è l'identificativo della struttura. Il valore è il seguente:

MQPSXP_STRUCID
MQPSXP_STRUCID è l'identificativo per la struttura del parametro di uscita di pubblicazione. Per il linguaggio di programmazione C, viene definita anche la costante MQPSXP_STRUC_ID_ARRAY ; ha lo stesso valore di MQPSXP_STRUC_ID, ma è un array di caratteri anziché una stringa.

StrucID è un campo di immissione per l'uscita.

***Version* (MQLONG)**

Version è il numero di versione della struttura. Il valore è il seguente:

MQPSXP_VERSION_1
MQPSXP_VERSION_1 è la struttura del parametro di uscita pubblicazione della Versione 1. La costante MQPSXP_CURRENT_VERSION è definita anche con lo stesso valore.

Version è un campo di immissione per l'uscita.

***ExitId* (MQLONG)**

ExitId è il tipo di uscita che viene richiamato. Il valore è il seguente:

MQXT_PUBLISH_EXIT
Uscita di pubblicazione.

ExitId è un campo di immissione per l'uscita.

***ExitReason* (MQLONG)**
ExitReason è il motivo per richiamare l'uscita. I valori possibili sono:

MQXR_INIT
L'uscita per questa connessione viene richiamata per l'inizializzazione. L'uscita potrebbe acquisire e inizializzare le risorse di cui ha bisogno; ad esempio, la memoria principale.

MQXR_TERM
L'uscita per questa connessione viene richiamata perché l'uscita sta per essere arrestata. L'uscita deve liberare tutte le risorse che ha acquisito da quando è stata inizializzata; ad esempio, la memoria principale.

MQXR_PUBLICATION
L'uscita viene richiamata dal gestore code prima di inserire una pubblicazione in una coda messaggi di un sottoscrittore. L'uscita può modificare il messaggio, non inserirlo nella coda o interrompere la pubblicazione.

ExitReason è un campo di immissione per l'uscita.

***ExitResponse* (MQLONG)**
Impostare *ExitResponse* nell'uscita per specificare il modo in cui l'elaborazione deve continuare. *ExitResponse* è uno dei seguenti valori:

MQXCC_OK
Impostare MQXCC_OK per continuare l'elaborazione normalmente. Impostare MQXCC_OK in risposta a qualsiasi valore di ExitReason.

Se ExitReason ha il valore MQXR_PUBLICATION, i campi *DestinationQName* e *DestinationQMgrName* della struttura MQSBC identificano la destinazione a cui viene inviato il messaggio.

MQXCC_FAILED
Impostare MQXCC_FAILED per arrestare l'operazione di pubblicazione. Il codice di completamento MQCC_FAILED e il codice motivo 2557 (09FD) (RC2557): MQRC_PUBLISH_EXIT_ERROR viene impostato al ritorno dall'uscita.

MQXCC_SUPPRESS_FUNCTION
Impostare MQXCC_SUPPRESS_FUNCTION per arrestare la normale elaborazione del messaggio. Impostare MQXCC_SUPPRESS_FUNCTION solo se *ExitReason* ha il valore MQXR_PUBLICATION.

Il messaggio continua ad essere elaborato dal gestore code in base all'opzione MQRO_DISCARD_MSG nel campo *Report* nel descrizione del messaggio.

- Se viene specificata l'opzione MQRO_DISCARD_MSG , il messaggio non viene consegnato al sottoscrittore.
- Se l'opzione MQRO_DISCARD_MSG non viene specificata, il messaggio viene inserito nella coda di messaggi non recapitabili. Se non è presente una coda di messaggi non recapitabili o se il messaggio non può essere inserito correttamente nella coda di messaggi non recapitabili, la pubblicazione non viene consegnata al sottoscrittore. La consegna della pubblicazione ad altri sottoscrittori dipende dai valori degli attributi dell'oggetto argomento PMSGDLV e NPMSGDLV . Per una spiegazione di questi attributi, consultare le descrizioni dei parametri per il comando DEFINE TOPIC .

ExitResponse è un campo di output dall'uscita.

***ExitResponse2* (MQLONG)**
ExitResponse2 è riservato per un utilizzo futuro.

***Feedback* (MQLONG)**
Feedback è il codice di feedback da utilizzare se l'uscita restituisce MQXCC_SUPPRESS_FUNCTION in *ExitResponse*.

All'immissione dell'uscita, *Feedback* ha sempre il valore MQFB_NONE. Se l'uscita restituisce MQXCC_SUPPRESS_FUNCTION, impostare *Feedback* sul valore da utilizzare per il messaggio quando il gestore code lo inserisce nella coda dei messaggi non instradabili. Al ritorno

dall'uscita, se *Feedback* ha il valore originale MQFB_NONE, il gestore code imposta *Feedback* su MQFB_STOPPED_BY_PUBSUB_EXIT.

Feedback è un campo di immissione / emissione per l'uscita.

***ExitUserArea* (MQBYTE16)**
ExitUserArea è un campo disponibile per l'uscita. Ogni connessione ha un *ExitUserArea*separato. La lunghezza di *ExitUserArea* è fornita da MQ_EXIT_USER_AREA_LENGTH.

Il campo *ExitReason* ha il valore MQXR_INIT al primo richiamo dell'uscita. *ExitUserArea* viene inizializzato in MQXUA_NONE al primo richiamo dell'uscita per una connessione. Le modifiche successive a *ExitUserArea* vengono conservate nei richiami dell'exit.

ExitUserArea è un campo di immissione / emissione per l'uscita.

***ExitData* (MQCHAR32)**
ExitData sono dati di uscita fissi definiti dal parametro **PublishExitData** della stanza nel file di inizializzazione del gestore code. I dati vengono riempiti con spazi vuoti fino alla lunghezza completa del campo. Se non sono definiti dati di uscita fissi nel file di inizializzazione, *ExitData* è vuoto. La lunghezza di *ExitData* è fornita da MQ_EXIT_DATA_LENGTH.

ExitData è un campo di immissione per l'uscita.

***QMgrName* (MQCHAR48)**
QMgrName è il nome del gestore code locale. Il nome viene riempito con spazi vuoti fino alla lunghezza completa del campo. La lunghezza di questo campo è fornita da MQ_Q_MGR_NAME_LENGTH.

QMgrName è un campo di immissione per l'uscita.

***Hconn* (MQHCONN)**
Hconn è l'handle che rappresenta una connessione al gestore code. Utilizzare *Hconn* solo come parametro per le chiamate della funzione di proprietà del messaggio MQSETMP, MQINQMMPo MQDLTMP per gestire le proprietà del messaggio.

Hconn è un campo di immissione per l'uscita.

***MsgDescPtr* (PMQMD)**
MsgDescPtr è l'indirizzo del descrittore del messaggio (MQMD) del messaggio da elaborare ed è una copia di MQMD restituito dalla chiamata MQPUT. L'uscita può modificare il contenuto del descrittore del messaggio. Qualsiasi modifica al contenuto del descrittore del messaggio deve essere effettuata con attenzione. In particolare, nel caso in cui il campo *SubType* della struttura MQSBC sia di valore MQSUBTYPE_PROXY, il campo *CorrelId* nel descrittore del messaggio non deve essere modificato.

Nessun descrittore di messaggi viene passato all'exit se *ExitReason* è MQXR_INIT o MQXR_TERM ; in questi casi, *MsgDescPtr* è il puntatore null.

MsgDescPtr è un campo di immissione per l'uscita.

***MsgHandle* (MQHMSG)**
MsgHandle è l'handle delle proprietà del messaggio. Utilizzare *MsgHandle* solo con le chiamate della funzione di proprietà del messaggio MQSETMP, MQINQMMPo MQDLTMP per gestire le proprietà del messaggio.

MsgHandle è un campo di immissione per l'uscita.

***MsgInPtr* (PMQVOID)**
MsgInPtr è l'indirizzo dei dati del messaggio di input. Il contenuto del buffer indirizzato da *MsgInPtr* può essere modificato dall'uscita; consultare *MsgOutPtr* .

MsgInPtr è un campo di immissione per l'uscita.

***MsgInLength* (MQLONG)**
MsgInLength è la lunghezza in byte dei dati del messaggio passati all'uscita. L'indirizzo dei dati è fornito da *MsgInPtr*.

MsgInLength è un campo di immissione per l'uscita.

***MsgOutPtr* (PMQVOID)**
MsgOutPtr è l'indirizzo di un buffer contenente i dati del messaggio restituiti dall'exit. All'ingresso nell'uscita, *MsgOutPtr* è null. Al ritorno dall'uscita, se il valore è ancora null, il gestore code invia il messaggio specificato da *MsgInPtr*, con la lunghezza fornita da *MsgInLength*.

Se l'uscita modifica i dati del messaggio, utilizzare una delle seguenti procedure:

- Se la lunghezza dei dati non cambia, i dati possono essere modificati nel buffer indicato da *MsgInPtr*. In questo caso, non modificare *MsgOutPtr* e *MsgOutLength*.
- Se i dati modificati sono più brevi dei dati originali, i dati possono essere modificati nel buffer indirizzato da *MsgInPtr*. In questo caso, *MsgOutPtr* deve essere impostato sull'indirizzo del buffer del messaggio di input e *MsgOutLength* deve essere impostato sulla nuova lunghezza dei dati del messaggio.
- Se i dati modificati sono o potrebbero essere più lunghi dei dati originali, l'exit deve ottenere un nuovo buffer di messaggi. Copiare i dati modificati in essi. Impostare *MsgOutPtr* sull'indirizzo del nuovo buffer e impostare *MsgOutLength* sulla lunghezza dei nuovi dati del messaggio. L'uscita è responsabile della liberazione del buffer indirizzato da *MsgOutPtr* quando l'uscita viene successivamente richiamata.

Nota: *MsgOutPtr* è sempre il puntatore null sull'input all'uscita e non l'indirizzo di un buffer di messaggi precedentemente ottenuto. Per liberare il buffer ottenuto in precedenza, l'exit deve salvare l'indirizzo e la lunghezza. Salvare le informazioni in *ExitUserArea*o in un blocco di controllo il cui indirizzo è stato salvato in *ExitUserArea*.

MsgOutPtr è un campo di immissione / emissione per l'uscita.

***MsgOutLength* (MQLONG)**
MsgOutLength è la lunghezza in byte dei dati del messaggio restituiti dall'exit. All'immissione dell'uscita, questo campo è sempre zero. Al ritorno dall'uscita, questo campo viene ignorato se *MsgOutPtr* è null. Consultare *MsgOutPtr* per informazioni sulla modifica dei dati del messaggio.

MsgOutLength è un campo di immissione / emissione per l'uscita.

***pEntryPoints* (PMQIEP)**
pEntryPoints è l'indirizzo di una struttura MQIEP attraverso cui è possibile effettuare chiamate MQI e DCI.

Dichiarazione di linguaggio C - MQPSXP

```
typedef struct tagMQPSXP {
    MQCHAR4    StrucId;             /* Structure identifier */
    MQLONG     Version;             /* Structure version number */
    MQLONG     ExitId;              /* Type of exit */
    MQLONG     ExitReason;          /* Reason for invoking exit */
    MQLONG     ExitResponse;        /* Response from exit */
    MQLONG     ExitResponse2;       /* Reserved */
    MQLONG     Feedback;            /* Feedback code */
    MQBYTE16   ExitUserArea;        /* Exit user area */
    MQCHAR32   ExitData;            /* Exit data */
    MQCHAR48   QMgrName;            /* Name of local queue manager */
    MQHCONN    Hconn;               /* Connection handle */
    MQHMSG     MsgHandle;           /* Handle to message properties */
    PMQMD      MsgDescPtr;          /* Address of message descriptor */
    PMQVOID    MsgInPtr;            /* Address of input message data */
    MQLONG     MsgInLength;         /* Length of input message data */
    PMQVOID    MsgOutPtr;           /* Address of output message data */
    MQLONG     MsgOutLength;        /* Length of output message data */
    /* Ver:1 */
    PMQIEP     pEntryPoints;        /* Address of the MQIEP structure */
    /* Ver:2 */
}    MQPSXP;
```

MQPBC - Struttura dei dati del contesto di pubblicazione

La struttura MQPBC contiene le informazioni contestuali, relative al publisher della pubblicazione, passate all'uscita di pubblicazione.

Tabella 819 a pagina 1510 riepiloga i campi nella struttura:

Tabella 819. Campi in MQPBC	
Campo	**Descrizione**
StrucID	Identificativo struttura
Version	Numero di versione della struttura
PubTopicString	Pubblica stringa topic
MsgDescPtr	Indirizzo del descrittore del messaggio (MQMD)

Campi

***StrucID* (MQCHAR4)**
StrucID è l'identificativo della struttura. Il valore è il seguente:

MQPBC_STRUCID
MQPBC_STRUCID è l'identificativo per la struttura del contesto di pubblicazione. Per il linguaggio di programmazione C, viene definita anche la costante MQPBC_STRUC_ID_ARRAY ; ha lo stesso valore di MQPBC_STRUC_ID, ma è un array di caratteri anziché una stringa.

StrucID è un campo di immissione per l'uscita.

***Version* (MQLONG)**
Version è il numero di versione della struttura. Il valore è il seguente:

MQPBC_VERSION_1
MQPBC_VERSION_1 è la struttura del parametro di uscita pubblicazione della Versione 1.

MQPBC_VERSION_2
MQPBC_VERSION_2 è la struttura del parametro di uscita pubblicazione della Versione 2. La costante MQPBC_CURRENT_VERSION è definita anche con lo stesso valore.

Version è un campo di immissione per l'uscita.

***PubTopicString* (MQCHARV)**
PubTopicString è la stringa di argomenti in cui viene pubblicato.

PubTopicString è un campo di immissione per l'uscita.

***MsgDescPtr* (PMQMD)**
MsgDescPtr è l'indirizzo di una copia del descrittore del messaggio (MQMD) per il messaggio elaborato.

MsgDescPtr è un campo di immissione per l'uscita.

Dichiarazione del linguaggio C - MQPBC

```
typedef struct tagMQPBC {
  MQCHAR4    StrucId;             /* Structure identifier */
  MQLONG     Version;             /* Structure version number */
  MQCHARV    PubTopicString;      /* Publish topic string */
  PMQMD      MsgDescPtr;          /* Address of message descriptor */
} MQPBC;
```

MQSBC - Struttura dati contesto sottoscrizione

La struttura MQSBC contiene le informazioni contestuali, relative al sottoscrittore (subscriber) che riceve la pubblicazione, passate all'uscita di pubblicazione.

Tabella 820 a pagina 1511 riepiloga i campi nella struttura:

Tabella 820. Campi in MQSBC	
Campo	**Descrizione**
StrucID	Identificativo struttura
Version	Numero di versione della struttura
DestinationQMgrName	Nome del gestore code di destinazione
DestinationQName	Nome della coda di destinazione
SubType	Il tipo di sottoscrizione
SubOptions	Opzioni di sottoscrizione
ObjectName	Nome oggetto
ObjectString	Stringa oggetto
SubTopicString	Stringa argomento sottoscrizione
SubName	Nome sottoscrizione
SubId	Identificativo sottoscrizione
SelectionString	Indirizzo della stringa di selezione
SubLevel	Livello sottoscrizione
PSProperties	Proprietà di pubblicazione / sottoscrizione

Campi

***StrucID* (MQCHAR4)**
Identificatore struttura. Il valore è il seguente:

MQSBC_STRUCID
MQSBC_STRUCID è l'identificativo per la struttura del parametro di uscita di pubblicazione. Per il linguaggio di programmazione C, viene definita anche la costante MQSBC_STRUC_ID_ARRAY ; MQSBC_STRUC_ID_ARRAY ha lo stesso valore di MQSBC_STRUC_ID, ma è un array di caratteri anziché una stringa.

StrucID è un campo di immissione per l'uscita.

***Version* (MQLONG)**
Numero di versione della struttura. Il valore è il seguente:

MQSBC_VERSION_1
Struttura del parametro di uscita pubblicazione versione 1. La costante MQSBC_CURRENT_VERSION è definita anche con lo stesso valore.

Version è un campo di immissione per l'uscita.

***DestinationQMgrName* (MQCHAR48)**
DestinationQMgrName è il nome del gestore code a cui viene inviato il messaggio. Il nome viene riempito con spazi vuoti fino alla lunghezza completa del campo. Il nome può essere modificato dall'uscita. La lunghezza di questo campo è fornita da MQ_Q_MGR_NAME_LENGTH.

DestinationQMgrName è un campo di immissione / emissione per l'uscita; vedere nota.

***DestinationQName* (MQCHAR48)**
DestinationQName è il nome della coda a cui viene inviato il messaggio. Il nome viene riempito con spazi vuoti fino alla lunghezza completa del campo. Il nome può essere modificato dall'uscita. La lunghezza di questo campo è fornita da MQ_Q_NAME_LENGTH.

DestinationQName è un campo di immissione / emissione per l'uscita; vedere nota.

***SubType* (MQLONG)**
SubType indica il modo in cui è stata creata la sottoscrizione. I valori validi sono MQSUBTYPE_API, MQSUBTYPE_ADMIN e MQSUBTYPE_PROXY ; vedere Interroga stato sottoscrizione (Risposta).

SubType è un campo di immissione per l'uscita.

***SubOptions* (MQLONG)**
SubOptions sono le opzioni di sottoscrizione; consultare "Opzioni (MQLONG) per MQSD" a pagina 587 per la descrizione dei valori che questo campo può assumere.

SubOptions è un campo di immissione per l'uscita.

***ObjectName* (MQCHAR48)**
ObjectName è il nome dell'oggetto argomento come definito sul gestore code locale. La lunghezza di questo campo è fornita da MQ_TOPIC_NAME_LENGTH. Il nome dell'oggetto è il nome dell'oggetto argomento di gestione che il gestore code ha associato alla stringa argomento. Anche se il sottoscrittore (subscriber) ha fornito un oggetto argomento come parte della sottoscrizione, *ObjectName* potrebbe essere un oggetto argomento diverso. L'associazione di un oggetto argomento ad una sottoscrizione dipende dalla risoluzione completa di *SubTopicString*.

ObjectName è un campo di immissione per l'uscita.

***ObjectString* (MQCHARV)**
ObjectString è la stringa di argomenti completa della pubblicazione che è stata sottoscritta. Tutti i caratteri jolly nella stringa di sottoscrizione originale vengono risolti. È diverso dal campo MQSD Sottoscrizione *ObjectString* descritto in "ObjectString (MQCHARV) per MQSD" a pagina 597, che potrebbe contenere caratteri jolly ed è escluso da qualsiasi nome oggetto fornito dal sottoscrittore.

ObjectString è un campo di immissione per l'uscita.

***SubTopicString* (MQCHARV)**
SubTopicString è la stringa di argomenti completa fornita dal sottoscrittore. *SubTopicString* è la combinazione della stringa argomento definita in un oggetto argomento e una stringa argomento. Un sottoscrittore deve fornire un oggetto argomento, una stringa argomento o entrambi. Se il sottoscrittore fornisce una stringa di argomenti, potrebbe contenere caratteri jolly.

SubTopicString è un campo di immissione per l'uscita.

***SubName* (MQCHARV)**
SubName è il nome della sottoscrizione fornito dal sottoscrittore o è un nome generato.

SubName è un campo di immissione per l'uscita.

***SubId* (MQBYTE 24)**
SubId è l'identificativo di sottoscrizione interno univoco.

SubId è un campo di immissione per l'uscita.

***SelectionString* (MQCHARV)**
SelectionString è il criterio di selezione utilizzato durante la sottoscrizione di messaggi da un argomento; consultare Selettori.

SelectionString è un campo di immissione per l'uscita.

***SubLevel* (MQLONG)**
SubLevel è il livello di intercettazione associato alla sottoscrizione; consultare "SubLevel (MQLONG) per MQSD" a pagina 601 per ulteriori dettagli.

SubLevel è un campo di immissione per l'uscita.

***PSProperties* (MQLONG)**
PSProperties sono le proprietà di pubblicazione / sottoscrizione. Specificano il modo in cui le proprietà dei messaggi relative alla pubblicazione / sottoscrizione vengono aggiunte ai messaggi inviati a questa sottoscrizione. I valori possibili sono MQPSPROP_NONE, MQPSPROP_COMPAT, MQPSPROP_RFH2, MQPSPROP_MSGPROP. Consultare Parametri facoltativi (Modifica, Copia e Crea sottoscrizione) per una descrizione di questi valori.

PSProperties è un campo di immissione per l'uscita.

Nota: I controlli di autorizzazione vengono eseguiti solo sui valori originali di *DestinationQMgrName* e *DestinationQName* prima che vengano passati all'uscita di pubblicazione. Non viene eseguito alcun nuovo controllo di autorizzazione quando l'uscita modifica la coda di destinazione, modificando *DestinationQMgrName* o *DestinationQName*.

Dichiarazione del linguaggio C - MQSBC

```
typedef struct tagMQSBC {
  MQCHAR4    StrucId;             /* Structure identifier */
  MQLONG     Version;             /* Structure version number */
  MQCHAR48   DestinationQMgrName; /* Destination queue manager */
  MQCHAR48   DestinationQName;    /* Destination queue name */
  MQLONG     SubType;             /* Type of subscription */
  MQLONG     SubOptions;          /* Subscription options */
  MQCHAR48   ObjectName;          /* Object name */
  MQCHARV    ObjectString;        /* Object string */
  MQCHARV    SubTopicString;      /* Subscription topic string */
  MQCHARV    SubName;             /* Subscription name */
  MQBYTE24   SubId;               /* Subscription identifier */
  MQCHARV    SelectionString;     /* Subscription selection string */
  MQLONG     SubLevel;            /* Subscription level */
  MQLONG     PSProperties;        /* Publish/subscribe properties */
} MQSBC;
```

Chiamate di uscita canale e strutture dati

Questa raccolta di argomenti fornisce informazioni di riferimento sulle chiamate IBM MQ speciali e sulle strutture di dati che è possibile utilizzare quando si scrivono programmi di uscita del canale.

Queste informazioni sono relative all'interfaccia di programmazione sensibile al prodotto. È possibile scrivere uscite utente IBM MQ nei seguenti linguaggi di programmazione:

Tabella 821. Uscite utente IBM MQ : piattaforme e linguaggi di programmazione

Piattaforma	**Linguaggi di programmazione**
IBM MQ for z/OS	Assembler e C (che devono essere conformi all'ambiente di programmazione del sistema C per le uscite di sistema, descritto in *z/OS C/C++ Programming Guide*.)
IBM MQ for IBM i	ILE C, ILE COBOL e ILE RPG
Tutte le altre piattaforme IBM MQ	C

È inoltre possibile scrivere uscite utente in Java da utilizzare solo con applicazioni Java e JMS . Per ulteriori informazioni sulla creazione e l'utilizzo delle uscite canale con IBM MQ classes for Java, consultare Utilizzo delle uscite canale in IBM MQ classes for Java e per IBM MQ classes for JMS, consultare Utilizzo delle uscite canale con IBM MQ classes for JMS.

Non è possibile scrivere uscite utente IBM MQ in TAL o Visual Basic. Tuttavia, una dichiarazione per la struttura MQCD viene fornita in Visual Basic per l'utilizzo sulla chiamata MQCONNX da un programma IBM MQ MQI client .

In una serie di casi nelle descrizioni che seguono, i parametri sono schiere o stringhe di caratteri con una dimensione non fissa. Per questi parametri, viene utilizzata una "n" minuscola per rappresentare una costante numerica. Quando la dichiarazione per tale parametro è codificata, la "n" deve essere sostituita dal valore numerico richiesto. Per ulteriori informazioni sulle convenzioni utilizzate in queste descrizioni, consultare “Tipi di dati elementari” a pagina 236.

File di definizione dati

I file di definizione dati vengono forniti con IBM MQ per ciascuno dei linguaggi di programmazione supportati. Per i dettagli di questi file, consultare File di copia, intestazione, inclusione e modulo.

MQ_CHANNEL_EXIT - Uscita canale

La chiamata MQ_CHANNEL_EXIT descrive i parametri passati a ciascuna delle uscite del canale richiamate da Message Channel Agent.

Nessun punto di ingresso denominato MQ_CHANNEL_EXIT viene fornito dal gestore code; il nome MQ_CHANNEL_EXIT non è di particolare importanza poiché i nomi delle uscite del canale vengono forniti nella definizione di canale MQCD.

Esistono cinque tipi di uscita canale:

- Uscita di sicurezza canale
- Uscita messaggio canale
- Uscita di invio canale
- Uscita ricezione canale
- Uscita nuovo tentativo messaggio canale

I parametri sono simili per ogni tipo di uscita, e la descrizione qui fornita si applica a tutti, tranne dove specificamente indicato.

Sintassi

MQ_CHANNEL_EXIT *(ChannelExitParms, ChannelDefinition, DataLength, AgentBufferLength, AgentBuffer, ExitBufferLength, ExitBufferAddr)*

Parametri

La chiamata MQ_CHANNEL_EXIT ha i parametri seguenti.

Parametri ChannelExit(MQCXP) - input/output

Blocco parametro di uscita canale.

Questa struttura contiene ulteriori informazioni relative al richiamo dell'exit. L'uscita imposta le informazioni in questa struttura per indicare come procede l'MCA.

ChannelDefinition (MQCD) - input/output

Definizione di canale.

Questa struttura contiene parametri impostati dall'amministratore per controllare il comportamento del canale.

DataLength (MQLONG) - input/output

Lunghezza dei dati.

I dati dipendono dal tipo di uscita:

- Per un'uscita di sicurezza del canale, quando l'uscita viene richiamata, questo parametro contiene la lunghezza di qualsiasi messaggio di sicurezza nel campo *AgentBuffer* , se *ExitReason* è MQXR_SEC_MSG. È zero se non c'è alcun messaggio. L'uscita deve impostare questo campo sulla lunghezza di qualsiasi messaggio di sicurezza da inviare al relativo partner, se imposta *ExitResponse* su MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG. I dati del messaggio si trovano in *AgentBuffer* o *ExitBufferAddr*.

  Il contenuto dei messaggi di sicurezza è di esclusiva responsabilità delle uscite di sicurezza.
- Per un'uscita del messaggio del canale, quando viene richiamata l'uscita, questo parametro contiene la lunghezza del messaggio (inclusa l'intestazione della coda di trasmissione). L'uscita

deve impostare questo campo sulla lunghezza del messaggio in *AgentBuffer* o *ExitBufferAddr* che deve procedere. Deve essere maggiore o uguale alla lunghezza dell'intestazione della coda di trasmissione (MQXQH).

- Per un'uscita di ricezione o di invio del canale, quando viene richiamata l'uscita, questo parametro contiene la lunghezza della trasmissione. L'uscita deve impostare questo campo sulla lunghezza della trasmissione in *AgentBuffer* o *ExitBufferAddr* che deve procedere.

Se un'uscita di sicurezza invia un messaggio e non c'è alcuna uscita di sicurezza all'altra estremità del canale, oppure l'altra estremità imposta un *ExitResponse* di MQXCC_OK, l'uscita di inizializzazione viene richiamata nuovamente con MQXR_SEC_MSG e una risposta null (*DataLength* = 0).

AgentBufferLength (MQLONG) - input

Lunghezza del buffer dell'agent.

Questo parametro può essere maggiore di *DataLength* al richiamo.

Per i messaggi del canale, le uscite di invio e ricezione, qualsiasi spazio inutilizzato sul richiamo può essere utilizzato dall'uscita per espandere i dati in posizione. In tal caso, il parametro **DataLength** deve essere impostato in modo appropriato dall'uscita.

Nel linguaggio di programmazione C, questo parametro viene passato per indirizzo.

AgentBuffer (MQBYTE x AgentBufferLength) - input/output

Buffer agent.

Il contenuto di questo parametro dipende dal tipo di uscita:

- Per un'uscita di sicurezza del canale, al richiamo dell'uscita contiene un messaggio di sicurezza se *ExitReason* è MQXR_SEC_MSG. Per inviare un messaggio di sicurezza, l'uscita può utilizzare questo buffer o il proprio buffer (*ExitBufferAddr*).
- Per un'uscita del messaggio del canale, al richiamo dell'uscita questo parametro contiene:
  - L'intestazione della coda di trasmissione (MQXQH), che include il descrittore del messaggio (che a sua volta contiene le informazioni di contesto per il messaggio), immediatamente seguita da
  - I dati del messaggio

  Se il messaggio deve continuare, l'uscita può effettuare una delle seguenti operazioni:

  - Lasciare inalterato il contenuto del buffer
  - Modificare il contenuto (restituendo la nuova lunghezza dei dati in *DataLength* ; non deve essere maggiore di *AgentBufferLength*)
  - Copiare il contenuto in *ExitBufferAddr*, apportando le modifiche richieste

  Tutte le modifiche apportate dall'uscita all'intestazione della coda di trasmissione non vengono controllate; tuttavia, modifiche errate potrebbero significare che il messaggio non può essere inserito nella destinazione.
- Per un'uscita di invio o ricezione del canale, al richiamo dell'uscita contiene i dati di trasmissione. L'uscita può effettuare una delle seguenti operazioni:
  - Lasciare inalterato il contenuto del buffer
  - Modificare il contenuto (restituendo la nuova lunghezza dei dati in *DataLength* ; non deve essere maggiore di *AgentBufferLength*)
  - Copiare il contenuto in *ExitBufferAddr*, apportando le modifiche richieste

  I primi 8 byte dei dati non devono essere modificati dall'uscita.

ExitBufferLength (MQLONG) - input/output

Lunghezza del buffer di uscita.

Al primo richiamo dell'uscita, questo parametro è impostato a zero. Successivamente, qualsiasi valore passato dall'uscita, ad ogni richiamo, viene presentato all'uscita la volta successiva che viene richiamato. Il valore non è utilizzato da MCA.

Nota: Questo parametro non deve essere utilizzato da uscite scritte in linguaggi di programmazione che non supportano il tipo di dati puntatore.

ExitBufferAddr (MQPTR) - input/output

Indirizzo del buffer di uscita.

Questo parametro è un puntatore all'indirizzo di un buffer di memoria gestito dall'uscita, dove può scegliere di restituire i dati di trasmissione o di messaggio (a seconda del tipo di uscita) all'agent se il buffer dell'agente è o potrebbe non essere abbastanza grande, o se è più conveniente per l'uscita farlo.

Al primo richiamo dell'uscita, l'indirizzo passato all'uscita è null. In seguito, qualsiasi indirizzo passato dall'uscita, ad ogni chiamata, viene presentato all'uscita la volta successiva che viene richiamato.

Se l'indirizzo ExitBufferè null, i dati utilizzati vengono presi dal parametro AgentBuffer .

Se l'indirizzo ExitBuffernon è null, i dati utilizzati vengono presi dal buffer indicato dal parametro dell'indirizzo ExitBuffer.

Nota: Questo parametro non deve essere utilizzato da uscite scritte in linguaggi di programmazione che non supportano il tipo di dati puntatore.

Richiamo C

```
exitname (&ChannelExitParms, &ChannelDefinition,
&DataLength, &AgentBufferLength, AgentBuffer,
&ExitBufferLength, &ExitBufferAddr);
```

I parametri passati all'uscita vengono dichiarati come segue:

```
MQCXP   ChannelExitParms;   /* Channel exit parameter block */
MQCD    ChannelDefinition;  /* Channel definition */
MQLONG  DataLength;         /* Length of data */
MQLONG  AgentBufferLength;  /* Length of agent buffer */
MQBYTE  AgentBuffer[n];     /* Agent buffer */
MQLONG  ExitBufferLength;   /* Length of exit buffer */
MQPTR   ExitBufferAddr;     /* Address of exit buffer */
```

Richiamo COBOL

```
     CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION,
                           DATALENGTH, AGENTBUFFERLENGTH, AGENTBUFFER,
                           EXITBUFFERLENGTH, EXITBUFFERADDR.
```

I parametri passati all'uscita vengono dichiarati come segue:

```
**   Channel exit parameter block
 01  CHANNELEXITPARMS.
     COPY CMQCXPV.
**   Channel definition
 01  CHANNELDEFINITION.
     COPY CMQCDV.
**   Length of data
 01  DATALENGTH         PIC S9(9) BINARY.
**   Length of agent buffer
 01  AGENTBUFFERLENGTH  PIC S9(9) BINARY.
**   Agent buffer
 01  AGENTBUFFER        PIC X(n).
**   Length of exit buffer
 01  EXITBUFFERLENGTH   PIC S9(9) BINARY.
```

```
**   Address of exit buffer
 01  EXITBUFFERADDR     POINTER.
```

Richiamo RPG (ILE)

```
     C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     C                    CALLP     exitname(MQCXP : MQCD : DATLEN :
     C                                       ABUFL : ABUF : EBUFL :
     C                                       EBUF)
```

La definizione del prototipo per la chiamata è:

```
     D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     Dexitname         PR                EXTPROC('exitname')
     D* Channel exit parameter block
     D MQCXP                          160A
     D* Channel definition
     D MQCD                          1328A
     D* Length of data
     D DATLEN                          10I 0
     D* Length of agent buffer
     D ABUFL                           10I 0
     D* Agent buffer
     D ABUF                              *   VALUE
     D* Length of exit buffer
     D EBUFL                           10I 0
     D* Address of exit buffer
     D EBUF                              *
```

Richiamo assembler System/390

```
         CALL EXITNAME,(CHANNELEXITPARMS,CHANNELDEFINITION,DATALENGTH, X
               AGENTBUFFERLENGTH,AGENTBUFFER,EXITBUFFERLENGTH,         X
               EXITBUFFERADDR)
```

I parametri passati all'uscita vengono dichiarati come segue:

```
CHANNELEXITPARMS   CMQCXPA  ,      Channel exit parameter block
CHANNELDEFINITION  CMQCDA   ,      Channel definition
DATALENGTH         DS       F      Length of data
AGENTBUFFERLENGTH  DS       F      Length of agent buffer
AGENTBUFFER        DS       CL(n)  Agent buffer
EXITBUFFERLENGTH   DS       F      Length of exit buffer
EXITBUFFERADDR     DS       F      Address of exit buffer
```

Note d'utilizzo

1. La funzione eseguita dall'uscita canale è definita dal fornitore dell'uscita. L'uscita, tuttavia, deve essere conforme alle regole definite qui e nel blocco di controllo associato, MQCXP.
2. Il parametro **ChannelDefinition** inoltrato all'uscita del canale potrebbe essere una delle diverse versioni. Per ulteriori informazioni, consultare il campo *Version* nella struttura MQCD.
3. Se l'uscita del canale riceve una struttura MQCD con il campo *Version* impostato su un valore maggiore di MQCD_VERSION_1, l'uscita deve utilizzare il campo *ConnectionName* in MQCD, anziché il campo *ShortConnectionName* .
4. In generale, le uscite canale possono modificare la lunghezza dei dati del messaggio. Ciò può verificarsi come risultato dell'uscita che aggiunge dati al messaggio o che rimuove i dati dal messaggio o che comprime o codifica il messaggio. Tuttavia, si applicano limitazioni speciali se il messaggio è un segmento che contiene solo una parte di un messaggio logico. In particolare, non deve esserci alcuna variazione netta della lunghezza del messaggio a seguito delle azioni delle uscite complementari di invio e ricezione.

Ad esempio, è consentito che un'uscita di invio riduca il messaggio comprimendo il messaggio, ma l'uscita di ricezione complementare deve ripristinare la lunghezza originale del messaggio decomprimendo il messaggio, in modo che non vi siano cambiamenti netti nella lunghezza del messaggio.

Questa limitazione si verifica perché la modifica della lunghezza di un segmento causerebbe l'errore degli offset dei segmenti successivi nel messaggio e ciò impedirebbe la capacità del gestore code di riconoscere che i segmenti formavano un messaggio logico completo.

MQ_CHANNEL_AUTO_DEF_EXIT - Uscita definizione automatica canale

La chiamata MQ_CHANNEL_AUTO_DEF_EXIT descrive i parametri passati all'uscita di definizione automatica del canale richiamata dall'agent del canale dei messaggi.

Nessun punto di ingresso denominato MQ_CHANNEL_AUTO_DEF_EXIT viene fornito dal gestore code; il nome MQ_CHANNEL_AUTO_DEF_EXIT non ha alcun significato particolare poiché i nomi delle uscite di definizione automatica vengono forniti nel gestore code.

Sintassi

MQ_CHANNEL_AUTO_DEF_EXIT *(ChannelExitParms, ChannelDefinition)*

Parametri

La chiamata MQ_CHANNEL_AUTO_DEF_EXIT ha i seguenti parametri.

Parametri ChannelExit(MQCXP) - input/output

Blocco parametro di uscita canale.

Questa struttura contiene ulteriori informazioni relative al richiamo dell'exit.L'uscita imposta le informazioni in questa struttura per indicare come procede l'MCA.

ChannelDefinition (MQCD) - input/output

Definizione di canale.

Questa struttura contiene parametri impostati dall'amministratore per controllare il comportamento dei canali che vengono creati automaticamente.L'uscita imposta le informazioni in questa struttura per modificare il comportamento predefinito impostato dall'amministratore.

I campi MQCD elencati non devono essere modificati dall'uscita:

- *ChannelName*
- *ChannelType*
- *StrucLength*
- *Version*

Se vengono modificati altri campi, il valore impostato dall'uscita deve essere valido. Se il valore non è valido, viene scritto un messaggio di errore nel file di log degli errori o visualizzato sulla console (come appropriato per l'ambiente).

Attenzione: I canali definiti automaticamente creati da un'uscita CHAD (channel automatic definition) non possono impostare l'etichetta del certificato, poiché l'handshake TLS si è verificato al momento della creazione del canale. L'impostazione dell'etichetta del certificato in un'uscita CHAD per i canali in ingresso non ha alcun effetto.

Richiamo C

```
exitname (&ChannelExitParms, &ChannelDefinition);
```

I parametri passati all'uscita vengono dichiarati come segue:

```
MQCXP  ChannelExitParms;   /* Channel exit parameter block */
MQCD   ChannelDefinition;  /* Channel definition */
```

Richiamo COBOL

```
     CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION.
```

I parametri passati all'uscita vengono dichiarati come segue:

```
**   Channel exit parameter block
 01  CHANNELEXITPARMS.
     COPY CMQCXPV.
**   Channel definition
 01  CHANNELDEFINITION.
     COPY CMQCDV.
```

Richiamo RPG (ILE)

```
     C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     C                   CALLP     exitname(MQCXP : MQCD)
```

La definizione del prototipo per la chiamata è:

```
     D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     Dexitname         PR                  EXTPROC('exitname')
     D* Channel exit parameter block
     D MQCXP                       160A
     D* Channel definition
     D MQCD                       1328A
```

Richiamo assembler System/390

```
         CALL EXITNAME,(CHANNELEXITPARMS,CHANNELDEFINITION)
```

I parametri passati all'uscita vengono dichiarati come segue:

```
CHANNELEXITPARMS   CMQCXPA  ,  Channel exit parameter block
CHANNELDEFINITION  CMQCDA   ,  Channel definition
```

Note d'utilizzo

1. La funzione eseguita dall'uscita canale è definita dal fornitore dell'uscita. L'uscita, tuttavia, deve essere conforme alle regole definite qui e nel blocco di controllo associato, MQCXP.
2. Il parametro **ChannelExitParms** inoltrato all'uscita di definizione automatica del canale è una struttura MQCXP. La versione di MQCXP passata dipende dall'ambiente in cui è in esecuzione l'uscita; consultare la descrizione del campo *Version* in “MQCXP - Parametro uscita canale” a pagina 1562 per dettagli.
3. Il parametro **ChannelDefinition** passato all'uscita di definizione automatica del canale è una struttura MQCD. La versione di MQCD passata dipende dall'ambiente in cui l'uscita è in esecuzione; consultare la descrizione del campo *Version* in “MQCD - Definizione canale” a pagina 1521 per i dettagli.

MQXWAIT - Attesa in uscita

La chiamata MQXWAIT attende che si verifichi un evento. Può essere utilizzato solo da un'uscita canale su z/OS.

L'utilizzo di MQXWAIT consente di evitare problemi di prestazioni che potrebbero verificarsi altrimenti se un'uscita del canale esegue un'operazione che causa un'attesa. L'evento MQXWAIT in attesa è segnalato da MVS ECB (event control block). La BCE è descritta nella descrizione del blocco di controllo MQXWD.

z/OS Per ulteriori informazioni sull'utilizzo di MQXWAIT e sulla scrittura dei programmi channel - exit, consultare Writing channel exit programs on z/OS

Sintassi

MQXWAIT ***(Hconn, WaitDesc, CompCode, Reason)***

Parametri

La chiamata MQXWAIT presenta i seguenti parametri.

Hconn (MQHCONN) - input

Handle di connessione.

Questo handle rappresenta la connessione al gestore code. Il valore di *Hconn* è stato restituito da una precedente chiamata MQCONN emessa nella stessa o precedente chiamata dell'uscita.

WaitDesc (MQXWD) - input/output

Descrittore di attesa.

Questo parametro descrive l'evento da attendere. Consultare “MQXWD - Descrittore di attesa uscita” a pagina 1577 per i dettagli dei campi in questa struttura.

CompCode (MQLONG) - output

Codice di completamento.

Si tratta di uno dei seguenti codici:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo (MQLONG) - output

Codice di errore *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

MQRC_ADAPTER_NON_DISPONIBILE
(2204, X'89C') Adattatore non disponibile.

ERRORE MQRC_OPTIONS_
(2046, X'7FE') Opzioni non valide o non congruenti.

MQRC_XWAIT_CANCELED
(2107, X'83B') Chiamata MQXWAIT annullata.

ERRORE MQRC_XWAIT
(2108, X'83C') Richiamo della chiamata MQXWAIT non valido.

Richiamo C

```
MQXWAIT (Hconn, &WaitDesc, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONN  Hconn;     /* Connection handle */
MQXWD    WaitDesc;  /* Wait descriptor */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Richiamo assembler System/390

```
         CALL MQXWAIT,(HCONN,WAITDESC,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
HCONN     DS       F  Connection handle
WAITDESC  CMQXWDA  ,  Wait descriptor
COMPCODE  DS       F  Completion code
REASON    DS       F  Reason code qualifying COMPCODE
```

MQCD - Definizione canale

La struttura MQCD contiene i parametri che controllano l'esecuzione di un canale. Viene passato a ciascuna uscita del canale richiamata da un MCA (Message Channel Agent).

Per ulteriori informazioni sulle uscite del canale, consultare “MQ_CHANNEL_EXIT - Uscita canale” a pagina 1514. La descrizione in questo argomento è relativa sia a canali di messaggi che a canali MQI.

Campi nome uscita

Quando viene richiamata un'uscita, il campo pertinente da *SecurityExit*, *MsgExit*, *SendExit*, *ReceiveExite* *MsgRetryExit* contiene il nome dell'uscita attualmente richiamata. Il significato del nome in questi campi dipende dall'ambiente in cui MCA è in esecuzione. Tranne dove indicato, il nome è allineato a sinistra all'interno del campo, senza spazi intermedi; il nome viene riempito con spazi vuoti fino alla lunghezza del campo. Nelle descrizioni che seguono, le parentesi quadre ([]) indicano informazioni facoltative:

AIX and Linux
Il nome di uscita è il nome di un modulo o di una libreria caricabili dinamicamente, con il suffisso del nome di una funzione che risiede in tale libreria. Il nome della funzione deve essere racchiuso tra parentesi. Il nome della libreria può essere facoltativamente preceduto da un percorso di directory:

```
[ path ] library ( function )
```

Il nome è limitato a un massimo di 128 caratteri.

z/OS
Il nome di uscita è il nome di un modulo di caricamento valido per la specifica sul parametro EP della macro LINK o LOAD. Il nome è limitato a un massimo di otto caratteri.

Windows
Il nome di uscita è il nome di una libreria a collegamento dinamico, con suffisso il nome di una funzione che risiede in tale libreria. Il nome della funzione deve essere racchiuso tra parentesi. Il nome della libreria può essere facoltativamente preceduto da un percorso di directory e da un'unità:

```
[d:][ path ] library ( function )
```

Il nome è limitato a un massimo di 128 caratteri.

IBM i
Il nome di uscita è un nome di programma a 10 byte seguito da un nome di libreria a 10 byte. Se i nomi hanno una lunghezza inferiore a 10 byte, ogni nome viene riempito con spazi vuoti per renderlo di 10 byte. Il nome della libreria può essere *LIBL tranne quando si richiama un'uscita di definizione automatica del canale, nel qual caso è richiesto un nome completo.

Modifica dei campi MQCD in un'uscita canale

Un'uscita canale può cambiare i campi in MQCD. Il valore modificato rimane in MQCD e viene passato a tutte le uscite rimanenti in una catena di uscita e a tutte le conversazioni che condividono l'istanza del canale. Il MQCD modificato viene utilizzato anche dall'MCA per la sua normale elaborazione durante la durata continua del canale.

I seguenti campi MQCD non devono essere alterati dall'uscita:

- ChannelName
- ChannelType
- StrucLength
- Versione

Riferimenti correlati
“Campi” a pagina 1522
Questo argomento elenca tutti i campi nella struttura MQCD e descrive ciascun campo.

“Dichiarazione C” a pagina 1549
Questa dichiarazione è la dichiarazione C per la struttura MQCD.

“Dichiarazione COBOL” a pagina 1551
Questa dichiarazione è la dichiarazione COBOL per la struttura MQCD.

“Dichiarazione RPG (ILE)” a pagina 1554
Questa dichiarazione è la dichiarazione RPG per la struttura MQCD.

“Dichiarazione assembler System/390” a pagina 1556
Questa dichiarazione è la dichiarazione dell'assembler System/390 per la struttura MQCD.

“Dichiarazione Visual Basic” a pagina 1558
Questa dichiarazione è la dichiarazione Visual Basic della struttura MQCD.

“Modifica dei campi MQCD in un'uscita canale” a pagina 1559
Un'uscita canale può cambiare i campi in MQCD. Tuttavia, queste modifiche non vengono generalmente applicate, ad eccezione delle circostanze elencate.

Campi
Questo argomento elenca tutti i campi nella struttura MQCD e descrive ciascun campo.

Limite BatchData(MQLONG)
Questo campo specifica il limite, in kilobyte, della quantità di dati che possono essere inviati attraverso un canale prima di prendere un punto di sincronizzazione.

Un punto di sincronizzazione viene acquisito dopo che il messaggio che ha causato il raggiungimento del limite è stato trasmesso attraverso il canale.

L'esecuzione del batch ha termine quando si verifica una delle seguenti condizioni:

- **BatchSize** messaggi sono stati inviati.
- **BatchDataLimit** byte sono stati inviati.
- La coda di trasmissione è vuota e **BatchInterval** è stata superata.

Il valore deve essere compreso tra 0 e 999999. Il valore predefinito è 5000.

Un valore zero in questo attributo indica che non viene applicato alcun limite di dati ai batch su questo canale.

Questo parametro si applica solo ai canali con un *ChannelType* di MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSRCVR o MQCHT_CLUSSDR.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_11.

BatchHeartbeat (MQLONG)
Questo campo specifica l'intervallo di tempo utilizzato per attivare un heartbeat batch per il canale.

L'heartbeat batch consente ai canali mittente di determinare se l'istanza del canale remoto è ancora attiva prima di essere in dubbio. Un heartbeat batch si verifica se un canale mittente non ha comunicato con l'istanza del canale remoto entro l'intervallo di tempo specificato.

Il valore è compreso tra 0 e 999 999 999; le unità sono millisecondi. Un valore zero indica che l'heartbeat batch non è abilitato.

Questo campo è rilevante solo per i canali che hanno un *ChannelType* di MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_7.

BatchInterval (MQLONG)
Questo campo specifica il tempo approssimativo in millisecondi in cui un canale mantiene aperto un batch, se sono stati trasmessi meno di *BatchSize* messaggi nel batch corrente.

Se *BatchInterval* è maggiore di zero, il batch viene terminato da uno dei seguenti eventi che si verificano per primi:

- *BatchSize* messaggi sono stati inviati o
- *BatchInterval* millisecondi trascorsi dall'inizio del batch.

Se *BatchInterval* è zero, il batch viene terminato da uno dei seguenti eventi che si verificano per primo:

- *BatchSize* messaggi sono stati inviati o
- la coda di trasmissione diventa vuota.

BatchInterval deve essere compreso tra zero e 999 999 999.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Questo è un campo di immissione per l'uscita. Il campo non è presente quando *Version* è minore di MQCD_VERSION_4.

BatchSize (MQLONG)
Questo campo specifica il numero massimo di messaggi che possono essere inviati attraverso un canale prima della sincronizzazione del canale.

Questo campo non è rilevante per i canali con un *ChannelType* di MQCHT_SVRCONN o MQCHT_CLNTCONN.

CertificateLabel (MQCHAR64)
Questo campo fornisce i dettagli dell'etichetta del certificato utilizzata.

IBM MQ inizializza il valore predefinito per il campo *CertificateLabel* come spazi vuoti.

Viene interpretato al runtime come il valore predefinito ed è compatibile con le versioni precedenti.

Ad esempio, specificando una versione MQCD inferiore a 11 o utilizzando il valore predefinito di spazi vuoti per il campo *CertificateLabel* , significa che questo campo viene ignorato.

La lunghezza di questo campo è fornita da MQ_CERT_LABEL_LENGTH.

ChannelMonitoring (MQLONG)
Questo campo specifica il livello corrente di raccolta dati di monitoraggio per il canale.

Questo campo non è rilevante per i canali con ChannelType di MQCHT_CLNTCONN.

È una dei seguenti valori:

- MQMON_DISATTIVO
- MMON_LOW
- MQMON_MEDIO
- MQMON_HIGH

Questo è un campo di immissione per l'uscita. Non è presente se *Version* è minore di MQCD_VERSION_8.

ChannelName (MQCHAR20)
Questo campo specifica il nome della definizione del canale.

Deve essere presente una definizione di canale con lo stesso nome sulla macchina remota per poter comunicare.

Il nome deve utilizzare solo i caratteri:

- Maiuscole A-Z
- Minuscole a-z
- Numeri 0-9
- Punto (.)
- Barra (/)
- Trattino basso (_)
- Segno percentuale (%)

ed essere riempiti a destra con spazi. Non sono consentiti spazi vuoti iniziali o centrali.

La lunghezza di questo campo è fornita da MQ_CHANNEL_NAME_LENGTH.

ChannelStatistics (MQLONG)
Questo campo specifica il livello corrente di raccolta dati statistici per il canale.

Questo campo non è rilevante per i canali con un ChannelType di MQCHT_CLNTCONN o MQCHT_SVRCONN.

È una dei seguenti valori:

- MQMON_DISATTIVO
- MMON_LOW
- MQMON_MEDIO
- MQMON_HIGH

Questo è un campo di immissione per l'uscita. Non è presente se *Version* è minore di MQCD_VERSION_8.

ChannelType (MQLONG)
Questo campo specifica il tipo di canale.

È una dei seguenti valori:

MQCH_SENDER
Mittente.

SERVER MQCHT
Server.

MQCH_DESTINATARIO
Destinatario.

RICHIESTA MQCHT_ER
Richiedente.

CLNTCONN MQCHT
Connessione client.

SVRCONN MQCHT
Connessione server (per l'utilizzo da parte dei client).

MQCHT_CLUSSDR
Mittente cluster.

CLUSRCVR MQCHT
Ricevente cluster.

ClientChannel(MQLONG)
Questo campo specifica una ponderazione per influenzare quale definizione di canale di connessione client viene utilizzata.

L'attributo ClientChannelPeso viene utilizzato in modo che le definizioni di canale client possano essere selezionate in modo casuale in base alla loro ponderazione quando è disponibile più di una definizione adatta. Quando un client emette un MQCONN che richiede la connessione a un gruppo di gestori code, specificando un nome gestore code che inizia con un asterisco e più di una definizione di canale adatta è disponibile nella CCDT (client channel definition table), la definizione da utilizzare viene selezionata casualmente in base al peso, con tutte le definizioni ClientChannelWeight (0) applicabili selezionate per prime in ordine alfabetico.

Specificare un valore compreso nell'intervallo 0 - 99. Il valore predefinito è 0.

Il valore 0 indica che non viene eseguito alcun bilanciamento del carico e che le funzioni applicabili vengono selezionate in ordine alfabetico. Per abilitare il bilanciamento del carico, scegliere un valore compreso fra 1 e 99, dove 1 è il peso minore e 99 quello maggiore. La distribuzione dei messaggi tra due o più canali con pesi diversi da zero è proporzionale al rapporto di tali pesi. Ad esempio, tre canali con valori di peso ClientChanneldi 2, 4 e 14 vengono selezionati approssimativamente il 10%, 20% e 70% del tempo. Questa distribuzione non è garantita.

Questo attributo è valido solo per il tipo di canale di connessione client.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Versione* è inferiore a MQCD_VERSION_9.

ClusterPtr (MQPTR)
Questo campo specifica l'indirizzo di un elenco di nomi cluster.

Se *ClustersDefined* è maggiore di zero, questo indirizzo è l'indirizzo di un elenco di nomi cluster. Il canale appartiene a ciascun cluster elencato.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_5.

ClustersDefined (MQLONG)
Questo campo specifica il numero di cluster a cui appartiene il canale.

Questo campo è il numero di nomi cluster indicati da *ClusterPtr*. È zero o maggiore.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_5.

CLWLChannelPriority (MQLONG)
Questo campo specifica la priorità del canale del carico di lavoro cluster.

L'algoritmo di selezione di Workload Manager seleziona una destinazione con la priorità più alta dalla serie di destinazioni selezionate in base alla classificazione. Se esistono due possibili gestori code di destinazione, questo attributo può essere utilizzato per eseguire il failover di un gestore code sull'altro gestore code. Tutti i messaggi vengono inviati al gestore code con la priorità più alta fino a quando non termina, quindi i messaggi vengono inviati al gestore code con la priorità più alta successiva.

Il valore è compreso tra 0 e 9. Il valore predefinito è 0.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_8.

Per ulteriori informazioni, consultare Configurazione di un cluster di gestori code.

CLWLChannelRank (MQLONG)
Questo campo specifica la classificazione del canale del carico di lavoro del cluster.

L'algoritmo di scelta del gestore del carico di lavoro seleziona una destinazione con la classificazione più alta. Quando la destinazione finale è un gestore code su un cluster differente, è possibile impostare la classificazione dei gestori code del gateway intermedio (all'intersezione dei cluster adiacenti) in modo che l'algoritmo di scelta scelga correttamente un gestore code di destinazione più vicino alla destinazione finale.

Il valore è compreso tra 0 e 9. Il valore predefinito è 0.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_8.

Per ulteriori informazioni, consultare Configurazione di un cluster di gestori code.

CLWLChannelWeight (MQLONG)
Questo campo specifica il peso del canale del carico di lavoro del cluster.

Peso del canale del carico di lavoro del cluster.

L'algoritmo di scelta del gestore del carico di lavoro utilizza l'attributo "weight" del canale per disallineare la scelta di destinazione in modo che più messaggi possano essere inviati a una particolare macchina. Ad esempio, è possibile assegnare a un canale su un server UNIX di grandi dimensioni un "peso" maggiore di un altro canale su un PC desktop di piccole dimensioni e l'algoritmo di scelta sceglie il server UNIX più frequentemente del PC.

Il valore è compreso tra 1 e 99. Il valore predefinito è 50.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_8.

Per ulteriori informazioni, consultare Configurazione di un cluster di gestori code.

ConnectionAffinity (MQLONG)
Questo campo specifica se le applicazioni client che si connettono più volte utilizzando lo stesso nome gestore code utilizzano lo stesso canale client.

Utilizzare questo attributo quando sono disponibili più definizioni canale applicabili.

Il valore è uno dei seguenti:

MQCAFTY_PREFERRED
La prima connessione in un processo che legge una CCDT (client channel definition table) crea un elenco di definizioni applicabili in base alla ponderazione con qualsiasi definizione CLNTWGHT (0) applicabile per prima e in ordine alfabetico. Ciascuna connessione del processo esegue tentativi di connessione utilizzando la prima definizione nell'elenco. Se una connessione non riesce, verrà utilizzata la definizione successiva. Le definizioni non riuscite con valori CLNTWGHT diversi da 0 vengono spostate alla fine dell'elenco. Le definizioni CLNTWGHT(0) restano all'inizio dell'elenco e vengono selezionate prima di ciascuna connessione.

Ogni processo client con lo stesso nome host crea sempre lo stesso elenco.

Per le applicazioni client scritte in C, C + + o il framework di programmazione .NET (incluso il .NETcompletamente gestito), l'elenco viene aggiornato se la CCDT è stata modificata da quando è stato creato l'elenco.

Questo è il valore predefinito.

MQCAFTI_NONE
La prima connessione in un processo che legge una tabella CCDT (client channel definition table) provvede alla creazione di un elenco di definizioni applicabili. Tutte le connessioni in un processo selezionano una definizione applicabile in base all'importanza, con tutte le definizioni CLNTWGHT(0) applicabili selezionate prima in ordine alfabetico.

Per le applicazioni client scritte in C, C + + o il framework di programmazione .NET (incluso il .NETcompletamente gestito), l'elenco viene aggiornato se la CCDT è stata modificata da quando è stato creato l'elenco.

Questo attributo è valido solo per il tipo di canale di connessione client.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Versione* è inferiore a MQCD_VERSION_9.

ConnectionName (MQCHAR264)
Questo campo specifica il nome della connessione per il canale.

Per i canali riceventi del cluster (quando specificato) CONNAME fa riferimento al gestore code locale e per altri canali fa riferimento al gestore code di destinazione. Il valore specificato dipende dal protocollo di trasmissione (*TransportType*) da utilizzare:

- Per MQXPT_LU62, è il nome completo dell'unità logica partner.
- Per MQXPT_NETBIOS, è il nome NetBIOS definito sulla macchina remota.
- Per MQXPT_TCP, è il nome host, l'indirizzo di rete della macchina remota specificato in formato IPv4 decimale con punti o IPv6 esadecimale, oppure la macchina locale per i canali riceventi del cluster.
- Per MQXPT_SPX, è un indirizzo in stile SPX che comprende un indirizzo di rete a 4 byte, un indirizzo nodo a 6 byte e un numero socket a 2 byte.

Quando si definisce un canale, questo campo non è rilevante per i canali con un *ChannelType* di MQCHT_SVRCONN o MQCHT_RECEIVER. Tuttavia, quando la definizione del canale viene passata a un'uscita, questo campo contiene l'indirizzo del partner, indipendentemente dal tipo di canale.

La lunghezza di questo campo è fornita da MQ_CONN_NAME_LENGTH. Questo campo non è presente se *Version* è minore di MQCD_VERSION_2.

DataConversion (MQLONG)
Questo campo specifica se l'agent del canale dei messaggi di invio tenta la conversione dei dati del messaggio dell'applicazione se l'agent del canale dei messaggi di ricezione non è in grado di eseguire questa conversione.

Questo campo si applica solo ai messaggi che non sono segmenti di messaggi logici; l'MCA non tenta mai di convertire i messaggi che sono segmenti.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR. Il valore è uno dei seguenti:

CONVERSIONE MQCDC_SENDER_
Conversione per mittente.

MQCDC_NO_SENDER_CONVERSIONE
Nessuna conversione da parte del mittente.

DefReconnect (MQLONG)
L'attributo del canale `DefReconnect` imposta il valore dell'attributo di riconnessione predefinito per un canale di connessione client.

Opzione predefinita di riconnessione automatica del client. È possibile configurare un IBM MQ MQI client per riconnettere automaticamente un'applicazione client. IBM MQ MQI client prova a riconnettersi a un gestore code dopo un malfunzionamento della connessione. Tenta di riconnettersi senza che il client dell'applicazione emetta una chiamata MQI MQCONN o MQCONNX.

La riconnessione è un'opzione MQCONNX . Utilizzando l'attributo del canale DefReconnect è possibile aggiungere il comportamento di riconnessione alle applicazioni esistenti che utilizzano MQCONN. È anche possibile modificare il comportamento di riconnessione delle applicazioni che utilizzano MQCONNX.

È anche possibile impostare il valore DefRecon dal file mqclient.ini per impostare o modificare il comportamento della riconnessione. Il valore DefRecon dal file mqclient.ini ha la precedenza sull'attributo del canale DefReconnect .

Syntax

DefReconnect (MQRCN_NO (default) |MQRCN_YES|MQRCN_Q_MGR| MQRCN_DISABLED)

Parametri

MQRCN_NO
MQRCN_NO è il valore predefinito.

A meno che non venga sovrascritto da **MQCONNX**, il client non viene riconnesso automaticamente.

MQRCN_YES
A meno che non venga sovrascritta da **MQCONNX**, il client si riconnette automaticamente.

MQRCN_Q_MGR
A meno che non venga sovrascritto da **MQCONNX**, il client si riconnette automaticamente, ma solo allo stesso gestore code. L'opzione QMGR ha lo stesso effetto di MQCNO_RECONNECT_Q_MGR.

MQRCN_DISABLED
La riconnessione è disabilitata, anche se richiesta dal programma client utilizzando la chiamata MQI **MQCONNX** .

La riconnessione automatica del client non è supportata da IBM MQ classes for Java.

Tabella 822. La riconnessione automatica dipende dai valori impostati nell'applicazione e nella definizione del canale

DefReconnect	**Opzioni di riconnessione impostate nell'applicazione**			
	MQCNO_RECONNECT	MQCNO_RECONNECT_Q_MGR	MQCNO_RECONNECT_AS_DEF	MQCNO_RECONNECT_DISABLED
MQRCN_NO	SÌ	QMGR	No	No
MQRCN_YES	SÌ	QMGR	SÌ	No
MQRCN_Q_MGR	SÌ	QMGR	QMGR	No
MQRCN_DISABLED	No	No	No	No

Concetti correlati
Riconnessione automatica del client
Riconnessione canale e client
Stanza CHANNELS del file di configurazione client
Riferimenti correlati
“Opzioni (MQLONG) per MQCNO” a pagina 327
Opzioni che controllano l'azione di MQCONNX.

Descrizione (MQCHAR64)
Questo campo può essere utilizzato per commenti descrittivi.

Il contenuto del campo non è significativo per gli agenti del canale dei messaggi. Tuttavia, deve contenere solo caratteri che possono essere visualizzati. Non può contenere caratteri null; se necessario, viene riempito a destra con spazi. In un'installazione DBCS, il campo può contenere caratteri DBCS (con una lunghezza massima di 64 byte).

Nota: Se questo campo contiene caratteri che non si trovano nella serie di caratteri del gestore code (come definito dall'attributo del gestore code **CodedCharSetId**), tali caratteri potrebbero essere tradotti in modo non corretto se questo campo viene inviato a un altro gestore code.

La lunghezza di questo campo è fornita da MQ_CHANNEL_DESC_LENGTH.

DiscInterval (MQLONG)
Questo campo specifica il tempo massimo in secondi per cui il canale attende l'arrivo di un messaggio sulla coda di trasmissione, prima di terminare il canale.

In altre parole, specifica l'intervallo di disconnessione.

Il valore A pari a zero fa sì che l'MCA attenda indefinitamente.

Per i canali di connessione server che utilizzano il protocollo TCP, l'intervallo rappresenta il valore di disconnessione inattività del client, specificato in secondi. Se una connessione server non ha ricevuto alcuna comunicazione dal client partner per questo periodo di tempo, termina la connessione. L'intervallo di inattività della connessione server si applica solo tra chiamate API IBM MQ da un client, quindi nessun client viene disconnesso durante una chiamata MQGET con attesa di lunga durata.

Questo attributo non è applicabile per i canali di connessione server che utilizzano protocolli diversi da TCP.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, MQCHT_CLUSRCVR o MQCHT_SVRCONN.

Lunghezza ExitData(MQLONG)
Questo campo specifica la lunghezza in byte di ogni elemento dati utente negli elenchi di elementi dati utente di uscita indirizzati dai campi *MsgUserDataPtr*, *SendUserDataPtr*e *ReceiveUserDataPtr* .

Questa lunghezza non è necessariamente uguale a MQ_EXIT_DATA_LENGTH.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_4.

Lunghezza ExitName(MQLONG)
Questo campo specifica la lunghezza in byte di ciascuno dei nomi negli elenchi di nomi di uscita indirizzati dai campi *MsgExitPtr*, *SendExitPtr*e *ReceiveExitPtr* .

Questa lunghezza non è necessariamente la stessa di MQ_EXIT_NAME_LENGTH.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_4.

Elenco HdrComp[2] (MQLONG)
Questo campo specifica l'elenco di tecniche di compressione dei dati di intestazione supportate dal canale.

L'elenco contiene uno o più dei seguenti valori:

MQCOMPRESS_NONE
Nessuna compressione dati di intestazione eseguita.

SISTEMA MQCOMPRESS
Compressione dati di intestazione eseguita correttamente.

MQCOMPRESS_NON_DISPONIBILE
I valori inutilizzati nell'elenco sono impostati su questo valore.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_8.

HeartbeatInterval (MQLONG)
Questo campo specifica il tempo, in secondi, tra i flussi di heartbeat.

L'interpretazione di questo campo dipende dal tipo di canale, come segue:

- Per un tipo di canale di MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER MQCHT_REQUESTER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR, questo campo è il tempo in secondi tra i flussi di heartbeat passati dall'MCA mittente quando non ci sono messaggi nella coda di trasmissione. Ciò fornisce all'MCA ricevente l'opportunità di sospendere il canale. Per essere utile, *HeartbeatInterval* deve essere minore di *DiscInterval*.
- Per un tipo di canale MQCHT_CLNTCONN o MQCHT_SVRCONN con il campo Conversazioni di condivisione MQCD impostato su zero, questo campo è il tempo, in secondi, tra i flussi di heartbeat trasmessi dall'MCA del server quando tale MCA ha emesso una chiamata MQGET con l'opzione MQGMO_WAIT per conto di un'applicazione client. Ciò consente al server MCA di gestire situazioni in cui la connessione client non riesce durante un MQGET con MQGMO_WAIT.
- Per un tipo di canale MQCHT_CLNTCONN o MQCHT_SVRCONN con il campo Conversazioni di condivisione MQCD impostato su un valore diverso da zero, questo campo è il tempo, in secondi, tra il flusso di heartbeat quando non vi sono flussi di dati inviati o ricevuti. Ciò consente al canale di essere disattivato in modo efficiente.

Il valore è compreso tra 0 e 999 999. Il valore utilizzato è il più grande dei valori specificati sul lato di invio e di ricezione a meno che non venga specificato un valore 0 su entrambi i lati, nel qual caso non si verifica alcuno scambio di heartbeat.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_4.

Intervallo KeepAlive(MQLONG)
Questo campo specifica il valore passato allo stack di comunicazioni per il tempo keepalive per il canale.

Il valore è applicabile per i protocolli di comunicazione TCP/IP e SPX, anche se non tutte le implementazioni supportano questo parametro.

Il valore è compreso tra 0 e 99 999; le unità sono i secondi. Un valore zero indica che il keepalive del canale non è abilitato, anche se keepalive potrebbe ancora verificarsi se è abilitato keepalive TCP/IP (piuttosto che keepalive del canale). È valido anche il seguente valore speciale:

AUTO MQKAI
Automatico.

Questo valore indica che l'intervallo keepalive viene calcolato dall'intervallo heartbeat negoziato, come segue:

- Se l'intervallo di heartbeat negoziato è maggiore di zero, l'intervallo keepalive utilizzato è l'intervallo di heartbeat più 60 secondi.
- Se l'intervallo di heartbeat negoziato è zero, l'intervallo keepalive utilizzato è zero.

- Su z/OS, il keepalive TCP/IP si verifica quando viene specificato TCPKEEP (YES) sull'oggetto gestore code.
- In altri ambienti, il keepalive TCP/IP si verifica quando il parametro **KEEPALIVE=YES** viene specificato nella stanza TCP nel file di configurazione dell'accodamento distribuito.

Questo campo è rilevante solo per i canali che hanno un *TransportType* di MQXPT_TCP o MQXPT_SPX.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_7.

LocalAddress (MQCHAR48)
Questo campo specifica l'indirizzo TCP/IP locale definito per il canale per le comunicazioni in uscita.

Questo campo è vuoto se non è definito alcun indirizzo specifico per le comunicazioni in uscita. L'indirizzo può facoltativamente includere un numero di porta o un intervallo di numeri di porta. Il formato di questo indirizzo è:

```
[ip-addr][(low-port[,high-port])]
```

dove le parentesi quadre ([]) indicano informazioni facoltative, `ip-addr` è specificato in IPv4 formato decimale puntato, IPv6 esadecimale o alfanumerico e `low-port` e `high-port` sono numeri di porta racchiusi tra parentesi. Sono tutti facoltativi.

Un indirizzo IP, una porta o un intervallo di porte specifici per le comunicazioni in uscita è utile negli scenari di recupero in cui un canale viene riavviato su uno stack TCP/IP differente.

LocalAddress è simile nel formato a *ConnectionName*, ma non deve essere confuso con esso. *LocalAddress* specifica le caratteristiche delle comunicazioni locali, mentre *ConnectionName* specifica come raggiungere un gestore code remoto.

Da IBM MQ 9.3.0, JMQI (Java Message Queueing Interface) è stato aggiornato per garantire che il campo dell'indirizzo locale sia impostato su un oggetto MQCD dopo che un'istanza del canale è stata creata ed è connessa a un gestore code. Ciò significa che quando un'uscita del canale scritta in Java richiama il metodo `MQCD.getLocalAddress()`, il metodo restituisce l'indirizzo locale utilizzato dall'istanza del canale. Prima di IBM MQ 9.3.0, l'uscita di sicurezza del canale non era in grado di accedere all'indirizzo locale utilizzato dall'istanza del canale e il metodo `MQCD.getLocalAddress()` ha restituito un valore null.

Questo campo è rilevante solo per i canali con un *TransportType* di MQXPT_TCP e un *ChannelType* di MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLNTCONN, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

La lunghezza di questo campo è fornita da MQ_LOCAL_ADDRESS_LENGTH. Questo campo non è presente se *Version* è minore di MQCD_VERSION_7.

LongMCAUserIdLength (MQLONG)
Questo campo specifica la lunghezza, in byte, dell'identificativo utente MCA completo indicato da *LongMCAUserIdPtr*.

Questo campo non è rilevante per i canali con un *ChannelType* MQCHT_CLNTCONN.

Questo è un campo di immissione / emissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_6.

LongMCAUserIdPtr (MQPTR)
Questo campo specifica l'indirizzo dell'identificativo utente MCA lungo.

Se *LongMCAUserIdLength* è maggiore di zero, questo campo è l'indirizzo dell'identificativo utente MCA completo. La lunghezza dell'identificativo completo è fornita da *LongMCAUserIdLength*. I primi 12 byte dell'identificativo utente MCA sono contenuti anche nel campo *MCAUserIdentifier*.

Consultare la descrizione del campo *MCAUserIdentifier* per i dettagli dell'identificativo utente MCA.

Questo campo non è rilevante per i canali con un *ChannelType* di MQCHT_SDR, MQCHT_SVR, MQCHT_CLNTCONN o MQCHT_CLUSSDR.

Questo è un campo di immissione / emissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_6.

LongRemoteUserIdLunghezza (MQLONG)
Questo campo specifica la lunghezza in byte dell'identificativo utente remoto completo indicato da *LongRemoteUserIdPtr*.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_CLNTCONN o MQCHT_SVRCONN.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_6.

LongRemoteUserIdPtr (MQPTR)
Questo campo specifica l'indirizzo dell'identificativo utente remoto lungo.

Se *LongRemoteUserIdLength* è maggiore di zero, questo indicatore è l'indirizzo dell'identificativo utente remoto completo. La lunghezza dell'identificativo completo è fornita da *LongRemoteUserIdLength*. I primi 12 byte dell'identificativo utente remoto sono contenuti anche nel campo *RemoteUserIdentifier*.

Consultare la descrizione del campo *RemoteUserIdentifier* per i dettagli dell'identificativo utente remoto.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_CLNTCONN o MQCHT_SVRCONN.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_6.

Conteggio LongRetry(MQLONG)
Questo campo specifica il conteggio utilizzato dopo che il conteggio specificato da *ShortRetryCount* è stato esaurito.

Specifica il numero massimo di ulteriori tentativi effettuati per collegarsi alla macchina remota, ad intervalli specificati da *LongRetryInterval*, prima di registrare un errore all'operatore.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Intervallo LongRetry(MQLONG)
Questo campo specifica il numero massimo di secondi da attendere prima di ritentare la connessione alla macchina remota.

L'intervallo tra i tentativi può essere esteso se il canale deve attendere per diventare attivo.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

MaxInstances (MQLONG)
Questo campo specifica il numero massimo di istanze simultanee di un singolo canale di connessione server che è possibile avviare.

Questo campo viene utilizzato solo su canali di connessione server.

Il campo può avere un valore compreso tra 0 e 999 999 999. Un valore pari a zero impedisce completamente l'accesso al client.

Il valore predefinito di questo campo è 999 999 999.

Se il valore di questo campo viene ridotto a un numero inferiore al numero di istanze del canale di connessione server attualmente in esecuzione, le istanze in esecuzione non vengono interessate. Tuttavia, le nuove istanze non possono essere avviate fino a quando non cessa l'esecuzione di un numero sufficiente di istanze esistenti in modo che il numero di istanze attualmente in esecuzione sia inferiore al valore del campo.

MaxInstancesPerClient (MQLONG)
Questo campo specifica il numero massimo di istanze simultanee di un singolo canale di connessione server che può essere avviato da un singolo client.

In questo contesto, le connessioni che hanno origine dallo stesso indirizzo di rete remoto sono considerate come provenienti dallo stesso client.

Questo campo viene utilizzato solo su canali di connessione server.

Il campo può avere un valore compreso tra 0 e 999 999 999. Un valore pari a zero impedisce completamente l'accesso al client.

Il valore predefinito di questo campo è 999 999 999.

Se il valore di questo campo viene ridotto a un numero inferiore al numero di istanze del canale di connessione server attualmente in esecuzione da singoli client, tali istanze in esecuzione non vengono interessate. Tuttavia, le nuove istanze da uno qualsiasi di questi client non possono essere avviate fino a quando non cessa l'esecuzione di un numero sufficiente di istanze esistenti, in modo che il numero di istanze attualmente in esecuzione, originate dal client che tenta di avviarne una nuova, sia inferiore al valore del campo.

Lunghezza MaxMsg(MQLONG)
Questo campo specifica la lunghezza massima del messaggio che può essere trasmesso sul canale.

Questo viene confrontato con il valore per il canale remoto e il valore massimo effettivo è il più basso dei due valori.

MCAName (MQCHAR20)
Questo campo è riservato.

Il valore di questo campo è vuoto.

La lunghezza di questo campo è fornita da MQ_MCA_NAME_LENGTH.

MCASecurityId (MQBYTE40)
Questo campo specifica l'identificativo di sicurezza per l'MCA.

Questo campo non è rilevante per i canali con un *ChannelType* MQCHT_CLNTCONN.

Il seguente valore speciale indica che non esiste alcun identificativo di sicurezza:

MQSID_NONE
Nessun identificativo di sicurezza specificato.

Il valore è zero binario per la lunghezza del campo.

Per il linguaggi di programmazione C, viene definita anche la costante MQSID_NONE_ARRAY; questa costante ha lo stesso valore di MQSID_NONE, ma è un array di caratteri invece di una stringa.

Questo è un campo di immissione / emissione per l'uscita. La lunghezza di questo campo è fornita da MQ_SECURITY_ID_LENGTH. Questo campo non è presente se *Version* è inferiore a MQCD_VERSION_6.

MCAType (MQLONG)
Questo campo specifica il tipo di programma agente canale messaggi.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Il valore è uno dei seguenti:

PROCESSO MQMCAT
processo.

L'agente canale messaggi viene eseguito come processo separato.

MQMCAT_THREAD
Thread (Multipiattaforme).

L'agente canale messaggi viene eseguito come sottoprocesso separato.

Questo campo non è presente quando *Versione* è inferiore a MQCD_VERSION_2.

MCAUserIdentifier (MQCHAR12)
Questo campo specifica l'identificativo utente per MCA (message channel agent).

Questo campo utilizza i primi 12 byte dell'identificativo utente MCA e può essere impostato da un agente di sicurezza.

Esistono due campi che contengono l'identificativo utente MCA:

- *MCAUserIdentifier* contiene i primi 12 byte dell'identificativo utente MCA e viene riempito con spazi se l'identificativo è inferiore a 12 byte. *MCAUserIdentifier* può essere vuoto.
- *LongMCAUserIdPtr* punta all'identificativo utente MCA completo, che può essere più lungo di 12 byte. La sua lunghezza è data da *LongMCAUserIdLength*. L'identificativo completo non contiene spazi vuoti finali e non ha terminazione nulla. Se l'identificativo è vuoto, *LongMCAUserIdLength* è zero e il valore di *LongMCAUserIdPtr* non è definito.

  Nota: *LongMCAUserIdPtr* non è presente se *Version* è minore di MQCD_VERSION_6.

Se l'identificativo utente MCA non è vuoto, specifica l'identificativo utente che deve essere utilizzato dall'agent del canale dei messaggi per l'autorizzazione ad accedere alle risorse IBM MQ . Per i tipi di canale MQCHT_REQUESTER, MQCHT_RECEIVER e MQCHT_CLUSRCVR, se PutAuthority è MQPA_DEFAULT questo è l'identificativo utente utilizzato per i controlli di autorizzazione per l'operazione di inserimento nelle code di destinazione.

Se l'identificativo utente MCA è vuoto, l'MCA (message channel agent) utilizza l'identificativo utente predefinito.

L'identificativo utente MCA può essere impostato da un'uscita di sicurezza per indicare l'identificativo utente che deve essere utilizzato dall'agent del canale dei messaggi. L'uscita può modificare *MCAUserIdentifier*o la stringa indicata da *LongMCAUserIdPtr*. Se entrambi vengono modificati ma differiscono l'uno dall'altro, l'MCA utilizza *LongMCAUserIdPtr* invece di *MCAUserIdentifier*. Se l'uscita modifica la lunghezza della stringa indirizzata da *LongMCAUserIdPtr*, *LongMCAUserIdLength* deve essere impostata di conseguenza. Se l'uscita aumenta la lunghezza dell'identificativo, l'uscita deve assegnare la memoria della lunghezza richiesta, impostare tale memoria sull'identificativo richiesto e posizionare l'indirizzo di tale memoria in *LongMCAUserIdPtr*. L'uscita è responsabile della liberazione di tale memoria quando l'uscita viene successivamente richiamata con il motivo MQXR_TERM.

Per i canali con un *ChannelType* di MQCHT_SVRCONN, se *MCAUserIdentifier* nella definizione del canale è vuoto, qualsiasi identificativo utente trasferito dal client viene copiato in esso. Questo identificativo utente (dopo qualsiasi modifica apportata dall'uscita di sicurezza sul server) è quello con cui si presume sia in esecuzione l'applicazione client.

L'identificativo utente MCA non è rilevante per canali con un *ChannelType* di MQCHT_SDR, MQCHT_SVR, MQCHT_CLNTCONN, MQCHT_CLUSSDR.

Questo è un campo di immissione / emissione per l'uscita. La lunghezza di questo campo è fornita da MQ_USER_ID_LENGTH. Questo campo non è presente quando *Version* è minore di MQCD_VERSION_2.

ModeName (MQCHAR8)
Questo campo specifica il nome della modalità LU 6.2 .

Questo campo è rilevante solo se il protocollo di trasmissione (*TransportType*) è MQXPT_LU62e *ChannelType* non è MQCHT_SVRCONN o MQCHT_RECEIVER.

Questo campo è sempre vuoto. Le informazioni sono contenute nell'oggetto laterale delle comunicazioni.

La lunghezza di questo campo è fornita da MQ_MODE_NAME_LENGTH.

Elenco MsgComp[16] (MQLONG)
Questo campo specifica l'elenco delle tecniche di compressione dei dati del messaggio supportate dal canale.

L'elenco contiene uno o più dei seguenti valori:

MQCOMPRESS_NONE
Nessuna compressione dati di messaggi eseguita.

RLE MQCOMPRESS
La compressione dei dati dei messaggi è stata eseguita mediante la codifica run-length.

MQCOMPRESS_ZLIBFAST
La compressione dei dati dei messaggi è stata eseguita mediante la tecnica di compressione zlib. È preferibile che il tempo di compressione sia breve.

MQCOMPRESS_ZLIBHIGH
La compressione dei dati dei messaggi è stata eseguita mediante la tecnica di compressione zlib. È preferibile che il tempo di compressione sia elevato.

V 9.4.0 **LZ4FAST**
La compressione dei dati dei messaggi viene eseguita utilizzando la codifica LZ4 con la velocità con priorità.

V 9.4.0 **LZ4HIGH**
La compressione dei dati dei messaggi viene eseguita utilizzando la codificazione LZ4 con priorità di compressione.

MQCOMPRESS_QUALSIASI
Qualsiasi tecnica di compressione supportata dal gestore code può essere utilizzata per la compressione dei messaggi. MQCOMPRESS_ANY è valido solo per i canali ricevente, richiedente e connessione server.

MQCOMPRESS_NON_DISPONIBILE
I valori inutilizzati nell'elenco sono impostati su questo valore.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_8.

MsgExit (MQCHARn)
Questo campo specifica il nome dell'uscita messaggio del canale.

Se questo nome non è vuoto, l'uscita viene richiamata nei seguenti orari:

- Immediatamente dopo che un messaggio è stato richiamato dalla coda di trasmissione (mittente o server) o immediatamente prima che un messaggio venga inserito in una coda di destinazione (destinatario o richiedente).

  All'uscita viene fornito l'intero messaggio dell'applicazione e l'intera intestazione della coda di trasmissione per la modifica.
- Al momento dell'inizializzazione e della chiusura del canale.

Questo campo non è rilevante per i canali con un *ChannelType* di MQCHT_SVRCONN o MQCHT_CLNTCONN; un'exit dei messaggi non viene mai richiamata per tali canali.

Consultare “MQCD - Definizione canale” a pagina 1521 per una descrizione del contenuto di questo campo in vari ambienti.

La lunghezza di questo campo è fornita da MQ_EXIT_NAME_LENGTH.

Nota: Il valore di questa costante è specifico dell'ambiente.

Ptr MsgExit(MQPTR)
Questo campo specifica l'indirizzo del primo campo *MsgExit* .

Se *MsgExitsDefined* è maggiore di zero, questo indirizzo è l'indirizzo dell'elenco di nomi di ciascuna uscita del messaggio del canale nella catena.

Ogni nome è in un campo di lunghezza *ExitNameLength*, riempito a destra con spazi. Ci sono *MsgExitsDefined* campi adiacenti l'uno all'altro - uno per ogni uscita.

Tutte le modifiche apportate a questi nomi da un'exit vengono conservate, sebbene l'exit del canale dei messaggi non intraprende alcuna azione esplicita - non modifica quali exit vengono richiamate.

Se *MsgExitsDefined* è zero, questo campo è il puntatore null.

Su piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_4.

Definito MsgExits(MQLONG)
Questo campo indica il numero di uscite di messaggi del canale definite nella catena.

È maggiore o uguale a zero.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_4.

Conteggio MsgRetry(MQLONG)
Questo campo specifica il numero di volte in cui MCA tenta di inserire il messaggio, dopo che il primo tentativo ha avuto esito negativo.

Questo campo indica il numero di volte in cui l'MCA tenta l'operazione di apertura o inserimento, se il primo MQOPEN o MQPUT ha esito negativo con codice di completamento MQCC_FAILED. L'effetto di questo attributo dipende dal fatto che *MsgRetryExit* sia vuoto o non vuoto:

- Se *MsgRetryExit* è vuoto, l'attributo **MsgRetryCount** controlla se l'MCA tenta di eseguire nuovi tentativi. Se il valore dell'attributo è zero, non viene effettuato alcun tentativo. Se il valore dell'attributo è maggiore di zero, i tentativi vengono tentati ad intervalli forniti dall'attributo **MsgRetryInterval** .

  I tentativi vengono eseguiti solo per i seguenti codici di errore:

  – MQRC_PAGESET_FULL
  – MQRC_PUT_INIBITO
  – MQRC_Q_FULL

  Per altri codici di errore, l'MCA procede immediatamente alla normale elaborazione dell'errore, senza ripetere il messaggio di errore.
- Se *MsgRetryExit* non è vuoto, l'attributo **MsgRetryCount** non influisce su MCA; è invece l'uscita del nuovo tentativo di messaggio che determina quante volte viene tentato il nuovo tentativo e a quali intervalli; l'uscita viene richiamata anche se l'attributo **MsgRetryCount** è zero.

  L'attributo **MsgRetryCount** viene reso disponibile per l'uscita nella struttura MQCD, ma l'uscita non richiesta per rispettarla - i tentativi continuano indefinitamente fino a quando l'uscita non restituisce MQXCC_SUPPRESS_FUNCTION nel campo *ExitResponse* di MQCXP.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

Questo campo non è presente quando *Version* è inferiore a MQCD_VERSION_3.

Uscita MsgRetry(MQCHARn)
Questo campo specifica il nome dell'uscita dei tentativi del messaggio del canale.

L'uscita di nuovo tentativo del messaggio è un'uscita richiamata dall'MCA quando l'MCA riceve un codice di completamento di MQCC_FAILED da una chiamata MQOPEN o MQPUT. Lo scopo dell'uscita è quello di specificare un intervallo di tempo per cui l'MCA attende prima di ritentare l'operazione MQOPEN o MQPUT. In alternativa, è possibile impostare l'uscita per non tentare di nuovo l'operazione.

L'uscita viene richiamata per tutti i codici motivo che hanno un codice di completamento MQCC_FAILED - le impostazioni dell'uscita determinano quali codici motivo desidera che l'MCA riprovi, per quanti tentativi e a quali intervalli di tempo.

Quando l'operazione non deve più essere tentata, l'MCA esegue la sua normale elaborazione di errore; questa elaborazione include la generazione di un messaggio di report di eccezione (se specificato dal mittente) e l'inserimento del messaggio originale nella coda di messaggi non recapitabili o l'eliminazione del messaggio (a seconda che il mittente abbia specificato MQRO_DEAD_LETTER_Q o MQRO_DISCARD_MSG). Gli errori che riguardano la coda di messaggi non recapitabili (ad esempio, la coda di messaggi non recapitabili piena) non causano il richiamo dell'uscita di nuovi tentativi di messaggi.

Se il nome dell'uscita non è vuoto, l'uscita viene richiamata nei seguenti orari:

- Immediatamente prima di eseguire l'attesa prima di tentare nuovamente la consegna di un messaggio
- All'inizializzazione e alla chiusura del canale

Consultare "MQCD - Definizione canale" a pagina 1521 per una descrizione del contenuto di questo campo in vari ambienti.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

La lunghezza di questo campo è fornita da MQ_EXIT_NAME_LENGTH.

Nota: Il valore di questa costante è specifico dell'ambiente.

Questo campo non è presente quando *Version* è inferiore a MQCD_VERSION_3.

Intervallo MsgRetry(MQLONG)
Questo campo specifica l'intervallo minimo in millisecondi dopo il quale l'operazione di apertura o di inserimento viene ritentata.

L'effetto di questo attributo dipende dal fatto che *MsgRetryExit* sia vuoto o non vuoto:

- Se *MsgRetryExit* è vuoto, l'attributo **MsgRetryInterval** specifica il periodo minimo di attesa di MCA prima di ritentare un messaggio, se il primo MQOPEN o MQPUT ha esito negativo con codice di completamento MQCC_FAILED. Il valore zero indica che il tentativo verrà eseguito il più presto possibile dopo il tentativo precedente. I nuovi tentativi vengono eseguiti solo se *MsgRetryCount* è maggiore di zero.

  Questo attributo viene utilizzato anche come tempo di attesa se l'uscita del nuovo tentativo del messaggio restituisce un valore non valido nel campo *MsgRetryInterval* in MQCXP.
- Se *MsgRetryExit* non è vuoto, l'attributo **MsgRetryInterval** non influisce sull'MCA, ma è l'uscita del nuovo tentativo di messaggio che determina il tempo di attesa dell'MCA. L'attributo **MsgRetryInterval** viene reso disponibile per l'uscita nella struttura MQCD, ma l'uscita non è richiesta per rispettarla.

Il valore è compreso tra 0 e 999 999 999.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

Questo campo non è presente quando *Version* è inferiore a MQCD_VERSION_3.

I seguenti campi in questa struttura non sono presenti se *Version* è inferiore a MQCD_VERSION_4.

MsgRetryUserData (MQCHAR32)
Questo campo specifica i dati utente di uscita dei tentativi dei messaggi del canale.

Questi dati vengono passati all'uscita di nuovo tentativo del canale nel campo *ExitData* del parametro **ChannelExitParms** (vedere MQ_CHANNEL_EXIT).

Questo campo inizialmente contiene i dati impostati nella definizione del canale. Tuttavia, durante il ciclo di vita di questa istanza MCA, le modifiche apportate al contenuto di questo campo da un'uscita di qualsiasi tipo vengono conservate dall'MCA e rese visibili alle successive chiamate di uscite (indipendentemente dal tipo) per questa istanza MCA. Tali modifiche non influenzano la definizione di canale utilizzata dalle altre istanze MCA. È possibile utilizzare qualsiasi carattere (inclusi i dati binari).

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

La lunghezza di questo campo è fornita da MQ_EXIT_DATA_LENGTH. Questo campo non è presente quando *Version* è inferiore a MQCD_VERSION_3.

Questo campo non è rilevante in IBM MQ for IBM i.

Dati MsgUser(MQCHAR32)
Questo campo specifica i dati utente di uscita messaggio canale.

Questi dati vengono trasmessi all'uscita del messaggio del canale nel campo *ExitData* del parametro **ChannelExitParms** (vedere MQ_CHANNEL_EXIT).

Questo campo inizialmente contiene i dati impostati nella definizione del canale. Tuttavia, durante il ciclo di vita di questa istanza MCA, le modifiche apportate al contenuto di questo campo da un'uscita di qualsiasi tipo vengono conservate dall'MCA e rese visibili alle successive chiamate di uscite (indipendentemente dal tipo) per questa istanza MCA. Tali modifiche non influenzano la definizione di canale utilizzata dalle altre istanze MCA. È possibile utilizzare qualsiasi carattere (inclusi i dati binari).

La lunghezza di questo campo è fornita da MQ_EXIT_DATA_LENGTH.

Questo campo non è rilevante in IBM MQ for IBM i.

MsgUserDataPtr (MQPTR)
Questo campo specifica l'indirizzo del primo campo *MsgUserData* .

Se *MsgExitsDefined* è maggiore di zero, questo indirizzo è l'indirizzo dell'elenco di elementi dati utente per ciascuna uscita del messaggio del canale nella concatenazione.

Ogni elemento dati utente è in un campo di lunghezza *ExitDataLength*, riempito a destra con spazi vuoti. Ci sono *MsgExitsDefined* campi adiacenti l'uno all'altro - uno per ogni uscita. Se il numero di elementi dati utente definiti è inferiore al numero di nomi di uscita, gli elementi dati utente non definiti vengono impostati su spazi vuoti. Al contrario, se il numero di elementi dati utente definito è maggiore del numero di nomi di uscita, gli elementi dati utente in eccesso vengono ignorati e non vengono presentati all'uscita.

Tutte le modifiche apportate a questi valori da un'uscita vengono conservate. Ciò consente a un'uscita di trasmettere informazioni a un'altra uscita. Non viene eseguita alcuna convalida su alcuna modifica, quindi, ad esempio, i dati binari possono essere scritti in questi campi, se necessario.

Se *MsgExitsDefined* è zero, questo campo è il puntatore null.

Su piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_4.

NetworkPriority (MQLONG)
Questo campo specifica la priorità della connessione di rete per il canale.

Quando sono disponibili più percorsi per una particolare destinazione, viene scelto il percorso con la priorità più alta. Il valore è compreso nell'intervallo tra 0 e 9; 0 è la priorità più bassa.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_5.

I seguenti campi in questa struttura non sono presenti se *Version* è inferiore a MQCD_VERSION_6.

NonPersistentMsgSpeed (MQLONG)
Questo campo indica la velocità con cui i messaggi non persistenti passano attraverso il canale.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER, MQCHT_REQUESTER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Il valore è uno dei seguenti:

MQNPMS_NORMAL
Velocità normale.

Se un canale è definito come MQNPMS_NORMAL, i messaggi non persistenti viaggiano attraverso il canale alla velocità normale. Ciò ha il vantaggio che questi messaggi non vengono persi se si verifica un errore del canale. Inoltre, i messaggi persistenti e non persistenti sulla stessa coda di trasmissione mantengono l'ordine relativo.

MQNPMS_FAST
Velocità veloce.

Se un canale è definito come MQNPMS_FAST, i messaggi non persistenti attraversano il canale a velocità elevata. Ciò migliora la velocità di trasmissione del canale, ma significa che i messaggi non persistenti vengono persi se si verifica un errore del canale. Inoltre, è possibile che i messaggi non persistenti saltino prima dei messaggi persistenti in attesa sulla stessa coda di trasmissione, vale a dire che l'ordine dei messaggi non persistenti non viene mantenuto relativamente ai messaggi persistenti. Tuttavia, l'ordine dei messaggi non persistenti viene mantenuto. Allo stesso modo, l'ordine dei messaggi persistenti viene mantenuto.

Password (MQCHAR12)
Questo campo specifica la password utilizzata dall'agent del canale messaggi quando si tenta di avviare una sessione SNA sicura con un agent del canale messaggi remoto.

Questo campo non può essere vuoto solo su AIX, Linux, and Windowsed è rilevante solo per i canali con un *ChannelType* di MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER o MQCHT_CLNTCONN. Su z/OS, questo campo non è rilevante.

La lunghezza di questo campo è fornita da MQ_PASSWORD_LENGTH. Tuttavia, vengono utilizzati solo i primi 10 caratteri.

Questo campo non è presente se *Version* è minore di MQCD_VERSION_2.

PropertyControl (MQLONG)
Questo campo specifica cosa succede alle proprietà dei messaggi quando il messaggio sta per essere inviato a un gestore code V6 o precedente (un gestore code che non comprende il concetto di un descrittore di proprietà).

Il valore può essere uno dei seguenti:

COMPATIBILITÀ_MQPROP_
Se il messaggio contiene una proprietà con il prefisso **mcd.**, **jms.**, **usr.**o **mqext.**, tutte le proprietà del messaggio vengono consegnate all'applicazione in un'intestazione MQRFH2 . In caso contrario, tutte le proprietà del messaggio, ad eccezione di quelle contenute nel descrittore del messaggio (o estensione), vengono scartate e non sono più accessibili all'applicazione.

Questo valore è il valore predefinito; consente alle applicazioni, che si aspettano che le proprietà relative a JMSsi trovano in una intestazione MQRFH2 nei dati del messaggio, di continuare a funzionare senza modifiche.

MQPROP_NONE
Tutte le proprietà del messaggio, tranne quelle nel descrittore del messaggio (o estensione), vengono rimosse dal messaggio prima che il messaggio venga inviato al gestore code remoto.

TUTTE le MQPROP
Tutte le proprietà del messaggio sono incluse nel messaggio quando viene inviato al gestore code remoto. Le proprietà, eccetto quelle nel descrittore di messaggi (o estensione) vengono collocate in una o più intestazioni MQRFH2 nei dati del messaggio.

Questo attributo è applicabile ai canali mittente, server, mittente cluster e destinatario cluster.

"MQIA_ * (Selettori di attributi interi)" a pagina 130
"MQPROP_ * (Valori di controllo proprietà coda e canale e lunghezza massima proprietà)" a pagina 170

PutAuthority (MQLONG)
Questo campo specifica se l'identificativo utente nelle informazioni di contesto associate a un messaggio viene utilizzato per stabilire l'autorizzazione per inserire il messaggio nella coda di destinazione.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR. Il valore è uno dei seguenti:

MQPA_PREDEFINITO
Viene utilizzato l'identificativo utente predefinito.

CONTEXT MQPA
Viene utilizzato un identificativo utente di contesto.

z/OS **MQPA_ALTERNATE_OR_MCA**
Viene utilizzato il campo UserIdentifier del descrittore del messaggio. Non viene utilizzato alcun ID utente ricevuto dalla rete. Questo valore è supportato solo su z/OS.

z/OS **MQPA_ONLY_MCA**
Viene utilizzato l'ID utente predefinito. Non viene utilizzato alcun ID utente ricevuto dalla rete. Questo valore è supportato solo su z/OS.

QMgrName (MQCHAR48)
Questo campo specifica il nome del gestore code a cui un'uscita può connettersi.

Per i canali con un *ChannelType* diverso da MQCHT_CLNTCONN, questo campo è il nome del gestore code a cui un'uscita può connettersi, che su AIX, Linux, and Windows, è sempre non vuoto.

La lunghezza di questo campo viene fornita da MQ_Q_MGR_NAME_LENGTH.

ReceiveExit (MQCHARn)
Questo campo specifica il nome dell'uscita di ricezione del canale.

Se questo nome non è vuoto, l'uscita viene richiamata nei seguenti orari:

- Immediatamente prima che i dati di rete ricevuti vengano elaborati.

  All'uscita viene fornito il buffer di trasmissione completo come ricevuto. Il contenuto del buffer può essere modificato come richiesto.
- Al momento dell'inizializzazione e della chiusura del canale.

Consultare "MQCD - Definizione canale" a pagina 1521 per una descrizione del contenuto di questo campo in vari ambienti.

La lunghezza di questo campo è fornita da MQ_EXIT_NAME_LENGTH.

Nota: Il valore di questa costante è specifico dell'ambiente.

Ptr ReceiveExit(MQPTR)
Questo campo specifica l'indirizzo del primo campo *ReceiveExit* .

Se *ReceiveExitsDefined* è maggiore di zero, questo indirizzo è l'indirizzo dell'elenco di nomi di ciascuna uscita di ricezione del canale nella concatenazione.

Ogni nome è in un campo di lunghezza *ExitNameLength*, riempito a destra con spazi. Ci sono *ReceiveExitsDefined* campi adiacenti l'uno all'altro - uno per ogni uscita.

Tutte le modifiche apportate a questi nomi da un'exit vengono conservate, sebbene l'exit del canale dei messaggi non intraprende alcuna azione esplicita - non modifica quali exit vengono richiamate.

Se *ReceiveExitsDefined* è zero, questo campo è il puntatore null.

Su piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_4.

ReceiveExitsdefinito (MQLONG)
Questo campo specifica il numero di uscite di ricezione del canale definito nella catena.

È maggiore o uguale a zero.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_4.

Dati ReceiveUser(MQCHAR32)
Questo canale specifica i dati utente di uscita ricezione canale.

Questi dati vengono passati all'uscita di ricezione del canale nel campo *ExitData* del parametro **ChannelExitParms** (vedere MQ_CHANNEL_EXIT).

Questo campo inizialmente contiene i dati impostati nella definizione del canale. Tuttavia, durante il ciclo di vita di questa istanza MCA, le modifiche apportate al contenuto di questo campo da un'uscita di qualsiasi tipo vengono conservate dall'MCA e rese visibili alle successive chiamate di uscite (indipendentemente dal tipo) per questa istanza MCA. Ciò si applica alle uscite su conversazioni diverse. Tali modifiche non influenzano la definizione di canale utilizzata dalle altre istanze MCA. È possibile utilizzare qualsiasi carattere (inclusi i dati binari).

La lunghezza di questo campo è fornita da MQ_EXIT_DATA_LENGTH.

Questo campo non è rilevante in IBM MQ for IBM i.

I seguenti campi in questa struttura non sono presenti se *Version* è minore di MQCD_VERSION_2.

ReceiveUserDataPtr (MQPTR)
Questo campo specifica l'indirizzo del primo campo *ReceiveUserData* .

Se *ReceiveExitsDefined* è maggiore di zero, questo indirizzo è l'indirizzo dell'elenco di voci di dati utente per ogni uscita di ricezione del canale nella concatenazione.

Ogni elemento dati utente è in un campo di lunghezza *ExitDataLength*, riempito a destra con spazi vuoti. Ci sono *ReceiveExitsDefined* campi adiacenti l'uno all'altro - uno per ogni uscita. Se il numero di elementi dati utente definiti è inferiore al numero di nomi di uscita, gli elementi dati utente non definiti vengono impostati su spazi vuoti. Al contrario, se il numero di elementi dati utente definito è maggiore del numero di nomi di uscita, gli elementi dati utente in eccesso vengono ignorati e non vengono presentati all'uscita.

Tutte le modifiche apportate a questi valori da un'uscita vengono conservate. Ciò consente a un'uscita di trasmettere informazioni a un'altra uscita. Non viene eseguita alcuna convalida su alcuna modifica, quindi, ad esempio, i dati binari possono essere scritti in questi campi, se necessario.

Se *ReceiveExitsDefined* è zero, questo campo è il puntatore null.

Su piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_4.

I seguenti campi in questa struttura non sono presenti se *Version* è minore di MQCD_VERSION_5.

RemotePassword (MQCHAR12)
Questo campo specifica la password da un partner.

Questo campo contiene informazioni valide solo se *ChannelType* è MQCHT_CLNTCONN o MQCHT_SVRCONN.

- Per un'uscita di sicurezza su un canale MQCHT_CLNTCONN, questa password è una password che è stata ottenuta dall'ambiente. L'uscita può scegliere di inviarlo all'uscita di sicurezza sul server.
- Per un'uscita di sicurezza su un canale MQCHT_SVRCONN, questo campo potrebbe contenere una password ottenuta dall'ambiente sul client, se non è presente alcuna uscita di sicurezza client. L'uscita può utilizzare questa parola d'ordine per confermare l'identificativo utente in *RemoteUserIdentifier*.

  Se è presente un'uscita di sicurezza sul client, queste informazioni possono essere ottenute in un flusso di sicurezza dal client.

La lunghezza di questo campo è fornita da MQ_PASSWORD_LENGTH. Questo campo non è presente se *Version* è minore di MQCD_VERSION_2.

ID RemoteSecurity(MQBYTE40)
Questo campo specifica l'identificativo di sicurezza per l'utente remoto.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_CLNTCONN o MQCHT_SVRCONN.

Il seguente valore speciale indica che non esiste alcun identificativo di sicurezza:

MQSID_NONE
Nessun identificativo di sicurezza specificato.

Il valore è zero binario per la lunghezza del campo.

Per il linguaggi di programmazione C, viene definita anche la costante MQSID_NONE_ARRAY; questa costante ha lo stesso valore di MQSID_NONE, ma è un array di caratteri invece di una stringa.

Questo è un campo di immissione per l'uscita. La lunghezza di questo campo è fornita da MQ_SECURITY_ID_LENGTH. Questo campo non è presente se *Version* è inferiore a MQCD_VERSION_6.

I seguenti campi in questa struttura non sono presenti se *Version* è inferiore a MQCD_VERSION_7.

Identificativo RemoteUser(MQCHAR12)
Questo campo specifica i primi 12 byte di un identificativo utente da un partner.

Ci sono due campi che contengono l'identificativo utente remoto:

- *RemoteUserIdentifier* contiene i primi 12 byte dell'identificativo utente remoto e viene riempito con spazi vuoti se l'identificativo è inferiore a 12 byte. *RemoteUserIdentifier* può essere vuoto.
- *LongRemoteUserIdPtr* punta all'identificativo utente remoto completo, che può essere più lungo di 12 byte. La sua lunghezza è data da *LongRemoteUserIdLength*. L'identificativo completo non contiene spazi vuoti finali e non ha terminazione nulla. Se l'identificativo è vuoto, *LongRemoteUserIdLength* è zero e il valore di *LongRemoteUserIdPtr* non è definito.

  LongRemoteUserIdPtr non è presente se *Version* è minore di MQCD_VERSION_6.

L'identificativo utente remoto è rilevante solo per i canali con un *ChannelType* di MQCHT_CLNTCONN o MQCHT_SVRCONN.

- Per un'uscita di sicurezza su un canale MQCHT_CLNTCONN, questo valore è un identificativo utente che è stato ottenuto dall'ambiente. L'uscita può scegliere di inviarlo all'uscita di sicurezza sul server.
- Per un'uscita di sicurezza su un canali MQCHT_SVRCONN, questo campo potrebbe contenere un identificativo utente che è stato ottenuto dall'ambiente sul client, se non esiste alcuna uscita di sicurezza client. L'uscita potrebbe convalidare questo ID utente (possibilmente con la password in *RemotePassword*) e aggiornare il valore in *MCAUserIdentifier*.

  Se è presente un'uscita di sicurezza sul client, queste informazioni possono essere ottenute in un flusso di sicurezza dal client.

La lunghezza di questo campo è fornita da MQ_USER_ID_LENGTH. Questo campo non è presente se *Version* è minore di MQCD_VERSION_2.

SecurityExit (MQCHARn)
Questo campo specifica il nome dell'uscita di sicurezza del canale.

Se questo nome non è vuoto, l'uscita viene richiamata nei seguenti orari:

- Immediatamente dopo aver stabilito un canale.

  Prima che un messaggio venga trasferito, viene fornita all'uscita la possibilità di avviare flussi di sicurezza per convalidare l'autorizzazione di connessione.
- Al ricevimento di una risposta ad un flusso di messaggi di sicurezza.

  Tutti i flussi di messaggi di sicurezza ricevuti dal processore remoto sulla macchina remota vengono forniti all'uscita.
- Al momento dell'inizializzazione e della chiusura del canale.

Consultare “MQCD - Definizione canale” a pagina 1521 per una descrizione del contenuto di questo campo in vari ambienti.

La lunghezza di questo campo è fornita da MQ_EXIT_NAME_LENGTH.

Nota: Il valore di questa costante è specifico dell'ambiente.

SecurityUserDati (MQCHAR32)
Questo canale specifica i dati utente dell'uscita di sicurezza del canale.

Questi dati vengono passati all'uscita di sicurezza nel campo *ExitData* del parametro **ChannelExitParms** (vedere MQ_CHANNEL_EXIT).

Questo campo inizialmente contiene i dati impostati nella definizione del canale. Tuttavia, durante il ciclo di vita di questa istanza MCA, le modifiche apportate al contenuto di questo campo da un'uscita di qualsiasi tipo vengono conservate dall'MCA e rese visibili alle successive chiamate di uscite (indipendentemente dal tipo) per questa istanza MCA. Ciò si applica alle uscite su conversazioni diverse. Tali modifiche non hanno effetto sulla definizione di canale utilizzata da altre istanze MCA. È possibile utilizzare qualsiasi carattere (inclusi i dati binari).

La lunghezza di questo campo è fornita da MQ_EXIT_DATA_LENGTH.

Questo campo non è rilevante in IBM MQ for IBM i.

SendExit (MQCHARn)
Questo campo specifica il nome dell'uscita di invio del canale.

Se questo nome non è vuoto, l'uscita viene richiamata nei seguenti orari:

- Immediatamente prima che i dati vengano inviati sulla rete.

  All'uscita viene fornito il buffer di trasmissione completo prima che venga trasmesso. Il contenuto del buffer può essere modificato come richiesto.
- Al momento dell'inizializzazione e della chiusura del canale.

Consultare “MQCD - Definizione canale” a pagina 1521 per una descrizione del contenuto di questo campo in vari ambienti.

La lunghezza di questo campo è fornita da MQ_EXIT_NAME_LENGTH.

Nota: Il valore di questa costante è specifico dell'ambiente.

Ptr SendExit(MQPTR)
Questo campo specifica l'indirizzo del primo campo *SendExit* .

Se *SendExitsDefined* è maggiore di zero, questo indirizzo è l'indirizzo dell'elenco di nomi di ciascuna uscita di invio del canale nella catena.

Ogni nome è in un campo di lunghezza *ExitNameLength*, riempito a destra con spazi. Ci sono *SendExitsDefined* campi adiacenti l'uno all'altro - uno per ogni uscita.

Tutte le modifiche apportate a questi nomi da un'uscita vengono conservate, sebbene l'uscita di invio del messaggio non intraprende alcuna azione esplicita - non modifica quali uscite vengono richiamate.

Se *SendExitsDefined* è zero, questo campo è il puntatore null.

Su piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_4.

SendExitsdefinito (MQLONG)
Questo campo specifica il numero di uscite di invio del canale definite nella catena.

È maggiore o uguale a zero.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_4.

Dati SendUser(MQCHAR32)
Questo campo specifica i dati utente di uscita di invio del canale.

Questi dati vengono passati all'uscita di invio del canale nel campo *ExitData* del parametro **ChannelExitParms** (vedere MQ_CHANNEL_EXIT).

Questo campo inizialmente contiene i dati impostati nella definizione del canale. Tuttavia, durante il ciclo di vita di questa istanza MCA, le modifiche apportate al contenuto di questo campo da un'uscita di qualsiasi tipo vengono conservate dall'MCA e rese visibili alle successive chiamate di uscite (indipendentemente dal tipo) per questa istanza MCA. Ciò si applica alle uscite su conversazioni diverse. Tali modifiche non influenzano la definizione di canale utilizzata dalle altre istanze MCA. È possibile utilizzare qualsiasi carattere (inclusi i dati binari).

La lunghezza di questo campo è fornita da MQ_EXIT_DATA_LENGTH.

Questo campo non è rilevante in IBM MQ for IBM i.

SendUserDataPtr (MQPTR)
Questo campo specifica l'indirizzo del campo *SendUserData* .

Se *SendExitsDefined* è maggiore di zero, questo indirizzo è l'indirizzo dell'elenco di elementi dati utente per ciascuna uscita del messaggio del canale nella concatenazione.

Ogni elemento dati utente è in un campo di lunghezza *ExitDataLength*, riempito a destra con spazi vuoti. Ci sono *MsgExitsDefined* campi adiacenti l'uno all'altro - uno per ogni uscita. Se il numero di elementi dati utente definiti è inferiore al numero di nomi di uscita, gli elementi dati utente non definiti vengono impostati su spazi vuoti. Al contrario, se il numero di elementi dati utente definito è maggiore del numero di nomi di uscita, gli elementi dati utente in eccesso vengono ignorati e non vengono presentati all'uscita.

Tutte le modifiche apportate a questi valori da un'uscita vengono conservate. Ciò consente a un'uscita di trasmettere informazioni a un'altra uscita. Non viene eseguita alcuna convalida su alcuna modifica, quindi, ad esempio, i dati binari possono essere scritti in questi campi, se necessario.

Se *SendExitsDefined* è zero, questo campo è il puntatore null.

Su piattaforme in cui il linguaggio di programmazione non supporta il tipo di dati puntatore, questo campo viene dichiarato come una stringa di byte della lunghezza appropriata.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è inferiore a MQCD_VERSION_4.

Wrapper SeqNumber(MQLONG)
Questo campo specifica il numero di sequenza messaggi massimo consentito.

Quando questo valore viene raggiunto, i numeri di sequenza vengono riportati a capo per iniziare di nuovo da 1.

Questo valore non è negoziabile e deve corrispondere in entrambe le definizioni di canale locale e remoto.

Questo campo non è rilevante per i canali con un *ChannelType* di MQCHT_SVRCONN o MQCHT_CLNTCONN.

SharingConversations (MQLONG)
Questo campo specifica il numero massimo di conversazioni che possono condividere un'istanza del canale associata a questo canale.

Questo campo viene utilizzato sui canali di connessione client e server.

Il valore 0 indica che il canale funziona come nelle versioni precedenti a IBM WebSphere MQ 7.0 rispetto ai seguenti attributi:

- Condivisione della conversazione
- Lettura anticipata
- STOP CHANNEL(*channelname*) MODE(QUIESCE)
- Heartbeat in corso
- Utilizzo asincrono client

Un valore di 1 è il valore minimo per il funzionamento di IBM MQ . Anche se è consentita solo una conversazione sull'istanza del canale, sono disponibili la lettura anticipata, il consumo asincrono e il comportamento dell'arresto del canale quiescente e dell'heartbeat CLNTCONN-SVRCONN .

Questo è un campo di immissione per l'uscita. Non è presente se *Version* è minore di MQCD_VERSION_9.

Il valore predefinito di questo campo è 10.

Nota: I limiti *MaxInstances* e *MaxInstancesPerClient* applicati a un canale limitano il numero di istanze del canale, non il numero di conversazioni che potrebbero condividere tali istanze.

ShortConnectionNome (MQCHAR20)
Questo campo specifica i primi 20 byte di un nome connessione.

Se il campo *Version* è MQCD_VERSION_1, *ShortConnectionName* contiene il nome completo della connessione.

Se il campo *Version* è MQCD_VERSION_2 o superiore, *ShortConnectionName* contiene i primi 20 caratteri del nome della connessione. Il nome completo della connessione viene fornito dal campo *ConnectionName* ; *ShortConnectionName* e i primi 20 caratteri di *ConnectionName* sono identici.

Consultare *ConnectionName* per i dettagli del contenuto di questo campo.

Nota: Il nome di questo campo è stato modificato per MQCD_VERSION_2 e le versioni successive di MQCD; il campo era precedentemente denominato *ConnectionName*.

La lunghezza di questo campo è fornita da MQ_SHORT_CONN_NAME_LENGTH.

Conteggio ShortRetry(MQLONG)
Questo campo specifica il numero massimo di tentativi effettuati per collegarsi a una macchina remota.

Questo campo è il numero massimo di tentativi effettuati per connettersi alla macchina remota, ad intervalli specificati da *ShortRetryInterval*, prima che vengano utilizzati *LongRetryCount* e *LongRetryInterval* (normalmente più lunghi).

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Intervallo ShortRetry(MQLONG)
Questo campo specifica il numero massimo di secondi da attendere prima di ritentare la connessione alla macchina remota.

L'intervallo tra i tentativi potrebbe essere esteso se il canale deve attendere per diventare attivo.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

z/OS *SPLProtection (MQLONG)*
This field specifies the value of the AMS security policy protection.

The value is one of the following:

MQSPL_PASSTHRU
Pass through, unchanged, any messages sent or received by the MCA for this channel.

This value is relevant only for channels with a *ChannelType* of MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER, or MQCHT_REQUESTER, and is the default value.

MQSPL_REMOVE
Remove any AMS protection from messages retrieved from the transmission queue by the MCA, and send the messages to the partner.

This value is relevant only for channels with a *ChannelType* of MQCHT_SENDER or MQCHT_SERVER.

MQSPL_ASPOLICY
Based on the policy defined for the target queue, apply AMS protection to inbound messages prior to putting them on to the target queue.

This value is relevant only for channels with a *ChannelType* of MQCHT_RECEIVER or MQCHT_REQUESTER.

This is an input field to the exit. This field is not present if *Version* is less than MQCD_VERSION_12.

SSLCipherSpec (MQCHAR32)
Questo campo specifica la specifica di cifratura in uso quando si utilizza TLS.

Se SSLCipherSpec è vuoto, il canale non utilizza TLS. Se non è vuoto, questo campo contiene una stringa che specifica la CipherSpec in uso.

Questo parametro è valido per tutti i tipi di canale. È supportato sulle piattaforme seguenti:

- AIX
- IBM i
- Linux
- Windows
- z/OS

È valido solo per i tipi di canale di un tipo di trasporto (TRPTYPE) di TCP.

Questo è un campo di immissione per l'uscita. La lunghezza di questo campo è fornita da MQ_SSL_CIPHER_SPEC_LENGTH. Il campo non è presente se *Version* è minore di MQCD_VERSION_7.

SSLClientAuth)
Questo campo specifica se è richiesta l'autenticazione client TLS.

Questo campo è rilevante solo per le definizioni di canale SVRCONN.

È una dei seguenti valori:

MQSCA_XX_ENCODE_CASE_ONE obbligatorio
Autenticazione client richiesta.

MQSCA_XX_ENCODE_CASE_ONE facoltativo
Autenticazione client facoltativa.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_7.

SSLPeerNameLunghezza (MQLONG)
Questo campo specifica la lunghezza, in byte, del nome peer TLS indicato da *SSLPeerNamePtr*.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_7.

Ptr SSLPeerName(MQPTR)
Questo campo specifica l'indirizzo del nome peer TLS.

Quando un certificato viene ricevuto durante un handshake TLS riuscito, il DN (Distinguished Name) dell'oggetto del certificato viene copiato nel campo MQCD a cui accede SSLPeerNamePtr alla fine del canale che riceve il certificato. Sovrascrive il valore SSLPeerName per il canale se questo valore è

presente nella definizione del canale dell'utente locale. Se un'uscita di sicurezza viene specificata a questa estremità del canale, riceve il DN (Distinguished Name) dal certificato peer in MQCD.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCD_VERSION_7.

Nota: Le applicazioni di uscita di sicurezza create prima del rilascio di IBM WebSphere MQ 7.1 potrebbero richiedere un aggiornamento. Per ulteriori informazioni, consultare Programmi di uscita di sicurezza del canale.

StrucLength (MQLONG)
Questo campo specifica la lunghezza in byte della struttura MQCD.

La lunghezza non include le stringhe indirizzate dai campi puntatore contenuti nella struttura. Il valore è uno dei seguenti:

MQCD_LENGTH_4
Lunghezza della struttura di definizioni di canali version-4 .

MQCD_LENGTH_5
Lunghezza della struttura di definizione del canale version-5 .

MQCD_LENGTH_6
Lunghezza della struttura di definizione del canale version-6 .

MQCD_LENGTH_7
Lunghezza della struttura di definizione del canale version-7 .

MQCD_LENGTH_8
Lunghezza della struttura di definizione del canale version-8 .

MQCD_LENGTH_9
Lunghezza della struttura di definizione del canale version-9 .

MQCD_LENGTH_10
Lunghezza della struttura di definizione del canale version-10 .

MQCD_LENGTH_11
Lunghezza della struttura di definizioni di canali version-11 .

z/OS **MQCD_LENGTH_12**
Lunghezza della struttura di definizione del canale version-12 .

La seguente costante specifica la lunghezza della versione corrente:

LENGTH MQCD_XX_ENCODE_CASE_ONE corrente
Lunghezza della versione corrente della struttura di definizione del canale.

Nota: Queste costanti hanno valori specifici dell'ambiente.

Il campo non è presente se *Version* è inferiore a MQCD_VERSION_4.

TpName (MQCHAR64)
Questo campo specifica il nome del programma di transazione LU 6.2 .

Questo campo è rilevante solo se il protocollo di trasmissione (*TransportType*) è MQXPT_LU62e *ChannelType* non è MQCHT_SVRCONN o MQCHT_RECEIVER.

Questo campo è sempre vuoto sulle piattaforme su cui le informazioni sono contenute nell'Oggetto laterale delle comunicazioni.

La lunghezza di questo campo è fornita da MQ_TP_NAME_LENGTH.

TransportType (MQLONG)
Questo campo specifica il protocollo di trasmissione da utilizzare.

Il valore non viene controllato se il canale è stato avviato dall'altra estremità.

È una dei seguenti valori:

MQXPT_LU62
Protocollo di trasporto LU 6.2 .

TCP MQXPT
Protocollo di trasporto TCP/IP.

NETBIOS MQXPT
Protocollo di trasporto NetBIOS .

Questo valore è supportato nei seguenti ambienti Windows.

SPX MQXPT
Protocollo di trasporto SPX.

Questo valore è supportato nei seguenti ambienti: Windows, più IBM MQ client connessi a questi sistemi.

UseDLQ (MQLONG)
Questo campo specifica se la coda di messaggi non recapitabili (o la coda di messaggi non recapitati) viene utilizzata quando i messaggi non possono essere consegnati dai canali.

Può contenere uno dei seguenti valori:

MQUSEDLQ_NO
I messaggi che non possono essere consegnati da un canale vengono considerati un errore. Il canale elimina il messaggio o il canale termina in base all'impostazione NPMSPEED.

MQUSEDLQ_SÌ
Quando l'attributo gestore code DEADQ fornisce il nome di una coda di messaggi non recapitabili, viene utilizzato, altrimenti il comportamento è NO. YES è il valore predefinito.

UserIdentifier (MQCHAR12)
Questo campo specifica l'identificativo utente utilizzato dall'agente canale dei messaggi quando si tenta di avviare una sessione SNA sicura con un agente canale dei messaggi remoto.

Questo campo non può essere vuoto solo su AIX, Linux, and Windowsed è rilevante solo per i canali con un *ChannelType* di MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER o MQCHT_CLNTCONN. Su z/OS, questo campo non è rilevante.

La lunghezza di questo campo è fornita da MQ_USER_ID_LENGTH. Tuttavia, vengono utilizzati solo i primi 10 caratteri.

Questo campo non è presente quando *Version* è minore di MQCD_VERSION_2.

Versione (MQLONG)
Il campo `Version` specifica il numero di versione più alto che è possibile impostare per la struttura.

Il valore dipende dall'ambiente:

MQCD _VERSION_1
Struttura di definizione del canale versione 1.

MQCD _VERSION_2
Struttura di definizione canale versione 2.

MQCD _VERSION_3
Struttura di definizione del canale versione 3.

MQCD _VERSION_4
Struttura di definizione del canale versione 4.

MQCD _VERSION_5
Struttura di definizione del canale versione 5.

MQCD _VERSION_6
Struttura di definizione del canale versione 6.

MQCD _VERSION_7
Struttura di definizione del canale versione 7.

MQCD _VERSION_8
Struttura di definizione canale versione 8.

MQCD _VERSION_9
Struttura di definizione del canale versione 9.

MQCD _VERSION_10
Struttura di definizione del canale versione 10.

MQCD _VERSION_11
Struttura di definizione del canale versione 11.

La versione 11 è la più alta su cui è possibile impostare il campo su IBM MQ 8.0 su tutte le piattaforme.

z/OS **MQCD _VERSION_12**
Struttura di definizione del canale versione 12.

La versione 12 è la più alta su cui è possibile impostare il campo su IBM MQ 9.1.3.

I campi che esistono solo nelle versioni più recenti della struttura vengono identificati come tali nelle descrizioni dei campi. La seguente costante specifica il numero di versione della versione corrente:

MQCD_CURRENT_VERSION
Il valore impostato in MQCD_CURRENT_VERSION è la versione corrente della struttura di definizioni di canale utilizzata.

Il valore di MQCD_CURRENT_VERSION dipende dall'ambiente. Contiene il valore più alto supportato dalla piattaforma.

MQCD_CURRENT_VERSION non viene utilizzato per inizializzare le strutture predefinite fornite nell'intestazione, copiare e includere i file forniti per i diversi linguaggi di programmazione. L'inizializzazione predefinita di Version dipende dalla piattaforma e dalla release.

Le dichiarazioni MQCD nei file di intestazione, copia e inclusione vengono inizializzate in MQCD_VERSION_6. Per utilizzare ulteriori campi MQCD , le applicazioni devono impostare il numero di versione su MQCD_CURRENT_VERSION. Se si sta scrivendo un'applicazione che è portabile tra diversi ambienti, è necessario scegliere una versione supportata in tutti gli ambienti.

Suggerimento: Quando viene introdotta una nuova versione della struttura MQCD , il layout della parte esistente non viene modificato. L'uscita deve controllare il numero di versione. Deve essere uguale o superiore alla versione più bassa che contiene i campi che l'uscita deve utilizzare.

XmitQName (MQCHAR48)
Questo campo specifica il nome della coda di trasmissione da cui vengono richiamati i messaggi.

Questo campo è rilevante solo per i canali con un *ChannelType* di MQCHT_SENDER o MQCHT_SERVER.

La lunghezza di questo campo è fornita da MQ_Q_NAME_LENGTH.

Dichiarazione C
Questa dichiarazione è la dichiarazione C per la struttura MQCD.

```
typedef struct tagMQCD MQCD;
typedef MQCD MQPOINTER PMQCD;
typedef PMQCD MQPOINTER PPMQCD;

 struct tagMQCD {
   MQCHAR    ChannelName[20];          /* Channel definition name */
   MQLONG    Version;                  /* Structure version number */
   MQLONG    ChannelType;              /* Channel type */
   MQLONG    TransportType;            /* Transport type */
   MQCHAR    Desc[64];                 /* Channel description */
   MQCHAR    QMgrName[48];             /* Queue manager name */
   MQCHAR    XmitQName[48];            /* Transmission queue name */
   MQCHAR    ShortConnectionName[20];  /* First 20 bytes of */
```

```
                                    /* connection name */
  MQCHAR    MCAName[20];            /* Reserved */
  MQCHAR    ModeName[8];            /* LU 6.2 Mode name */
  MQCHAR    TpName[64];             /* LU 6.2 transaction program */
                                    /* name */
  MQLONG    BatchSize;              /* Batch size */
  MQLONG    DiscInterval;           /* Disconnect interval */
  MQLONG    ShortRetryCount;        /* Short retry count */
  MQLONG    ShortRetryInterval;     /* Short retry wait interval */
  MQLONG    LongRetryCount;         /* Long retry count */
  MQLONG    LongRetryInterval;      /* Long retry wait interval */
  MQCHAR    SecurityExit[128];      /* Channel security exit name */
  MQCHAR    MsgExit[128];           /* Channel message exit name */
  MQCHAR    SendExit[128];          /* Channel send exit name */
  MQCHAR    ReceiveExit[128];       /* Channel receive exit name */
  MQLONG    SeqNumberWrap;          /* Highest allowable message */
                                    /* sequence number */
  MQLONG    MaxMsgLength;           /* Maximum message length */
  MQLONG    PutAuthority;           /* Put authority */
  MQLONG    DataConversion;         /* Data conversion */
  MQCHAR    SecurityUserData[32];   /* Channel security exit user */
                                    /* data */
  MQCHAR    MsgUserData[32];        /* Channel message exit user */
                                    /* data */
  MQCHAR    SendUserData[32];       /* Channel send exit user */
                                    /* data */
  MQCHAR    ReceiveUserData[32];    /* Channel receive exit user */
                                    /* data */
  /* Ver:1 */
  MQCHAR    UserIdentifier[12];     /* User identifier */
  MQCHAR    Password[12];           /* Password */
  MQCHAR    MCAUserIdentifier[12];  /* First 12 bytes of MCA user */
                                    /* identifier */
  MQLONG    MCAType;                /* Message channel agent type */
  MQCHAR    ConnectionName[264];    /* Connection name */
  MQCHAR    RemoteUserIdentifier[12]; /* First 12 bytes of user */
                                    /* identifier from partner */
  MQCHAR    RemotePassword[12];     /* Password from partner */
  /* Ver:2 */
  MQCHAR    MsgRetryExit[128];      /* Channel message retry exit */
                                    /* name */
  MQCHAR    MsgRetryUserData[32];   /* Channel message retry exit */
                                    /* user data */
  MQLONG    MsgRetryCount;          /* Number of times MCA will */
                                    /* try to put the message, */
                                    /* after first attempt has */
                                    /* failed */
  MQLONG    MsgRetryInterval;       /* Minimum interval in */
                                    /* milliseconds after which */
                                    /* the open or put operation */
                                    /* will be retried */
  /* Ver:3 */
  MQLONG    HeartbeatInterval;      /* Time in seconds between */
                                    /* heartbeat flows */
  MQLONG    BatchInterval;          /* Batch duration */
  MQLONG    NonPersistentMsgSpeed;  /* Speed at which */
                                    /* nonpersistent messages are */
                                    /* sent */
  MQLONG    StrucLength;            /* Length of MQCD structure */
  MQLONG    ExitNameLength;         /* Length of exit name */
  MQLONG    ExitDataLength;         /* Length of exit user data */
  MQLONG    MsgExitsDefined;        /* Number of message exits */
                                    /* defined */
  MQLONG    SendExitsDefined;       /* Number of send exits */
                                    /* defined */
  MQLONG    ReceiveExitsDefined;    /* Number of receive exits */
                                    /* defined */
  MQPTR     MsgExitPtr;             /* Address of first MsgExit */
                                    /* field */
  MQPTR     MsgUserDataPtr;         /* Address of first */
                                    /* MsgUserData field */
  MQPTR     SendExitPtr;            /* Address of first SendExit */
                                    /* field */
  MQPTR     SendUserDataPtr;        /* Address of first */
                                    /* SendUserData field */
  MQPTR     ReceiveExitPtr;         /* Address of first */
                                    /* ReceiveExit field */
  MQPTR     ReceiveUserDataPtr;     /* Address of first */
                                    /* ReceiveUserData field */
  /* Ver:4 */
  MQPTR     ClusterPtr;             /* Address of a list of */
                                    /* cluster names */
```

```
   MQLONG    ClustersDefined;          /* Number of clusters to */
                                       /* which the channel belongs */
   MQLONG    NetworkPriority;          /* Network priority */
   /* Ver:5 */
   MQLONG    LongMCAUserIdLength;      /* Length of long MCA user */
                                       /* identifier */
   MQLONG    LongRemoteUserIdLength;   /* Length of long remote user */
                                       /* identifier */
   MQPTR     LongMCAUserIdPtr;         /* Address of long MCA user */
                                       /* identifier */
   MQPTR     LongRemoteUserIdPtr;      /* Address of long remote */
                                       /* user identifier */
   MQBYTE40  MCASecurityId;            /* MCA security identifier */
   MQBYTE40  RemoteSecurityId;         /* Remote security identifier */
   /* Ver:6 */
   MQCHAR    SSLCipherSpec[32];        /* TLS CipherSpec */
   MQPTR     SSLPeerNamePtr;           /* Address of TLS peer name */
   MQLONG    SSLPeerNameLength;        /* Length of TLS peer name */
   MQLONG    SSLClientAuth;            /* Whether TLS client */
                                       /* authentication is required */
   MQLONG    KeepAliveInterval;        /* Keepalive interval */
   MQCHAR    LocalAddress[48];         /* Local communications */
                                       /* address */
   MQLONG    BatchHeartbeat;           /* Batch heartbeat interval */
   /* Ver:7 */
   MQLONG    HdrCompList[2];           /* Header data compression */
                                       /* list */
   MQLONG    MsgCompList[16];          /* Message data compression */
                                       /* list */
   MQLONG    CLWLChannelRank;          /* Channel rank */
   MQLONG    CLWLChannelPriority;      /* Channel priority */
   MQLONG    CLWLChannelWeight;        /* Channel weight */
   MQLONG    ChannelMonitoring;        /* Channel monitoring */
   MQLONG    ChannelStatistics;        /* Channel statistics */
   /* Ver:8 */
   MQLONG    SharingConversations;     /* Limit on sharing */
                                       /* conversations */
   MQLONG    PropertyControl;          /* Message property control */
   MQLONG    MaxInstances;             /* Limit on SVRCONN channel */
                                       /* instances */
   MQLONG    MaxInstancesPerClient;    /* Limit on SVRCONN channel */
                                       /* instances per client */
   MQLONG    ClientChannelWeight;      /* Client channel weight */
   MQLONG    ConnectionAffinity;       /* Connection affinity */
   /* Ver:9 */
   MQLONG    BatchDataLimit;           /* Batch data limit */
   MQLONG    UseDLQ;                   /* Use Dead Letter Queue */
   MQLONG    DefReconnect;             /* Default client reconnect */
                                       /* option */
   /* Ver:10 */
   MQCHAR64  CertificateLabel;         /* Certificate label */
 /* Ver:11 */
   MQLONG    SPLProtection             /* AMS Security policy protection */
 /* Ver:12 */
};
```

Dichiarazione COBOL

Questa dichiarazione è la dichiarazione COBOL per la struttura MQCD.

```
** MQCD structure
       10  MQCD.
      ** Channel definition name
       15  MQCD-CHANNELNAME PIC X(20).
      ** Structure version number
       15  MQCD-VERSION PIC S9(9) BINARY.
      ** Channel type
       15  MQCD-CHANNELTYPE PIC S9(9) BINARY.
      ** Transport type
       15  MQCD-TRANSPORTTYPE PIC S9(9) BINARY.
      ** Channel description
       15  MQCD-DESC PIC X(64).
      ** Queue manager name
       15  MQCD-QMGRNAME PIC X(48).
      ** Transmission queue name
       15  MQCD-XMITQNAME PIC X(48).
      ** First 20 bytes of connection name
       15  MQCD-SHORTCONNECTIONNAME PIC X(20).
      ** Reserved
```

```
        15  MQCD-MCANAME PIC X(20).
       ** LU 6.2 Mode name
        15  MQCD-MODENAME PIC X(8).
       ** LU 6.2 transaction program name
        15  MQCD-TPNAME PIC X(64).
       ** Batch size
        15  MQCD-BATCHSIZE PIC S9(9) BINARY.
       ** Disconnect interval
        15  MQCD-DISCINTERVAL PIC S9(9) BINARY.
       ** Short retry count
        15  MQCD-SHORTRETRYCOUNT PIC S9(9) BINARY.
       ** Short retry wait interval
        15  MQCD-SHORTRETRYINTERVAL PIC S9(9) BINARY.
       ** Long retry count
        15  MQCD-LONGRETRYCOUNT PIC S9(9) BINARY.
       ** Long retry wait interval
        15  MQCD-LONGRETRYINTERVAL PIC S9(9) BINARY.
       ** Channel security exit name
        15  MQCD-SECURITYEXIT PIC X(20).
       ** Channel message exit name
        15  MQCD-MSGEXIT PIC X(20).
       ** Channel send exit name
        15  MQCD-SENDEXIT PIC X(20).
       ** Channel receive exit name
        15  MQCD-RECEIVEEXIT PIC X(20).
       ** Highest allowable message sequence number
        15  MQCD-SEQNUMBERWRAP PIC S9(9) BINARY.
       ** Maximum message length
        15  MQCD-MAXMSGLENGTH PIC S9(9) BINARY.
       ** Put authority
        15  MQCD-PUTAUTHORITY PIC S9(9) BINARY.
       ** Data conversion
        15  MQCD-DATACONVERSION PIC S9(9) BINARY.
       ** Channel security exit user data
        15  MQCD-SECURITYUSERDATA PIC X(32).
       ** Channel message exit user data
        15  MQCD-MSGUSERDATA PIC X(32).
       ** Channel send exit user data
        15  MQCD-SENDUSERDATA PIC X(32).
       ** Channel receive exit user data
        15  MQCD-RECEIVEUSERDATA PIC X(32).
       ** Ver:1 **
       ** User identifier
        15  MQCD-USERIDENTIFIER PIC X(12).
       ** Password
        15  MQCD-PASSWORD PIC X(12).
       ** First 12 bytes of MCA user identifier
        15  MQCD-MCAUSERIDENTIFIER PIC X(12).
       ** Message channel agent type
        15  MQCD-MCATYPE PIC S9(9) BINARY.
       ** Connection name
        15  MQCD-CONNECTIONNAME PIC X(264).
       ** First 12 bytes of user identifier from partner
        15  MQCD-REMOTEUSERIDENTIFIER PIC X(12).
       ** Password from partner
        15  MQCD-REMOTEPASSWORD PIC X(12).
       ** Ver:2 **
       ** Channel message retry exit name
        15  MQCD-MSGRETRYEXIT PIC X(20).
       ** Channel message retry exit user data
        15  MQCD-MSGRETRYUSERDATA PIC X(32).
       ** Number of times MCA will try to put the message, after first
       ** attempt has failed
        15  MQCD-MSGRETRYCOUNT PIC S9(9) BINARY.
       ** Minimum interval in milliseconds after which the open or put
       ** operation will be retried
        15  MQCD-MSGRETRYINTERVAL PIC S9(9) BINARY.
       ** Ver:3 **
       ** Time in seconds between heartbeat flows
        15  MQCD-HEARTBEATINTERVAL PIC S9(9) BINARY.
       ** Batch duration
        15  MQCD-BATCHINTERVAL PIC S9(9) BINARY.
       ** Speed at which nonpersistent messages are sent
        15  MQCD-NONPERSISTENTMSGSPEED PIC S9(9) BINARY.
       ** Length of MQCD structure
        15  MQCD-STRUCLENGTH PIC S9(9) BINARY.
       ** Length of exit name
        15  MQCD-EXITNAMELENGTH PIC S9(9) BINARY.
       ** Length of exit user data
        15  MQCD-EXITDATALENGTH PIC S9(9) BINARY.
       ** Number of message exits defined
        15  MQCD-MSGEXITSDEFINED PIC S9(9) BINARY.
```

```
      ** Number of send exits defined
       15  MQCD-SENDEXITSDEFINED PIC S9(9) BINARY.
      ** Number of receive exits defined
       15  MQCD-RECEIVEEXITSDEFINED PIC S9(9) BINARY.
      ** Address of first MsgExit field
       15  MQCD-MSGEXITPTR POINTER.
      ** Address of first MsgUserData field
       15  MQCD-MSGUSERDATAPTR POINTER.
      ** Address of first SendExit field
       15  MQCD-SENDEXITPTR POINTER.
      ** Address of first SendUserData field
       15  MQCD-SENDUSERDATAPTR POINTER.
      ** Address of first ReceiveExit field
       15  MQCD-RECEIVEEXITPTR POINTER.
      ** Address of first ReceiveUserData field
       15  MQCD-RECEIVEUSERDATAPTR POINTER.
      ** Ver:4 **
      ** Address of a list of cluster names
       15  MQCD-CLUSTERPTR POINTER.
      ** Number of clusters to which the channel belongs
       15  MQCD-CLUSTERSDEFINED PIC S9(9) BINARY.
      ** Network priority
       15  MQCD-NETWORKPRIORITY PIC S9(9) BINARY.
      ** Ver:5 **
      ** Length of long MCA user identifier
       15  MQCD-LONGMCAUSERIDLENGTH PIC S9(9) BINARY.
      ** Length of long remote user identifier
       15  MQCD-LONGREMOTEUSERIDLENGTH PIC S9(9) BINARY.
      ** Address of long MCA user identifier
       15  MQCD-LONGMCAUSERIDPTR POINTER.
      ** Address of long remote user identifier
       15  MQCD-LONGREMOTEUSERIDPTR POINTER.
      ** MCA security identifier
       15  MQCD-MCASECURITYID PIC X(40).
      ** Remote security identifier
       15  MQCD-REMOTESECURITYID PIC X(40).
      ** Ver:6 **
      ** TLS CipherSpec
       15  MQCD-SSLCIPHERSPEC PIC X(32).
      ** Address of TLS peer name
       15  MQCD-SSLPEERNAMEPTR POINTER.
      ** Length of TLS peer name
       15  MQCD-SSLPEERNAMELENGTH PIC S9(9) BINARY.
      ** Whether TLS client authentication is required
       15  MQCD-SSLCLIENTAUTH PIC S9(9) BINARY.
      ** Keepalive interval
       15  MQCD-KEEPALIVEINTERVAL PIC S9(9) BINARY.
      ** Local communications address
       15  MQCD-LOCALADDRESS PIC X(48).
      ** Batch heartbeat interval
       15  MQCD-BATCHHEARTBEAT PIC S9(9) BINARY.
      ** Ver:7 **
      ** Header data compression list
       15  MQCD-HDRCOMPLIST PIC S9(9) BINARY.
      ** Message data compression list
       15  MQCD-MSGCOMPLIST PIC S9(9) BINARY.
      ** Channel rank
       15  MQCD-CLWLCHANNELRANK PIC S9(9) BINARY.
      ** Channel priority
       15  MQCD-CLWLCHANNELPRIORITY PIC S9(9) BINARY.
      ** Channel weight
       15  MQCD-CLWLCHANNELWEIGHT PIC S9(9) BINARY.
      ** Channel monitoring
       15  MQCD-CHANNELMONITORING PIC S9(9) BINARY.
      ** Channel statistics
       15  MQCD-CHANNELSTATISTICS PIC S9(9) BINARY.
      ** Ver:8 **
      ** Limit on sharing conversations
       15  MQCD-SHARINGCONVERSATIONS PIC S9(9) BINARY.
      ** Message property control
       15  MQCD-PROPERTYCONTROL PIC S9(9) BINARY.
      ** Limit on SVRCONN channel instances
       15  MQCD-MAXINSTANCES PIC S9(9) BINARY.
      ** Limit on SVRCONN channel instances per client
       15  MQCD-MAXINSTANCESPERCLIENT PIC S9(9) BINARY.
      ** Client channel weight
       15  MQCD-CLIENTCHANNELWEIGHT PIC S9(9) BINARY.
      ** Connection affinity
       15  MQCD-CONNECTIONAFFINITY PIC S9(9) BINARY.
      ** Ver:9 **
      ** Batch data limit
       15  MQCD-BATCHDATALIMIT PIC S9(9) BINARY.
```

```
      ** Use Dead Letter Queue
       15  MQCD-USEDLQ PIC S9(9) BINARY.
      ** Default client reconnect option
       15  MQCD-DEFRECONNECT PIC S9(9) BINARY.
      ** Ver:10 **
      ** Certificate Label
       15 MQCD-CERTLABL PIC X (64)
      ** Ver:11 **
      ** AMS Security policy protection
       15 MQCD-SPLPROTECTION PIC S9(9) BINARY
      ** Ver:12 **
```

Dichiarazione RPG (ILE)

Questa dichiarazione è la dichiarazione RPG per la struttura MQCD.

```
D* MQCD Structure
     D*
     D* Channel definition name
     D  CDCHN                  1     20
     D* Structure version number
     D  CDVER                 21     24I 0
     D* Channel type
     D  CDCHT                 25     28I 0
     D* Transport type
     D  CDTRT                 29     32I 0
     D* Channel description
     D  CDDES                 33     96
     D* Queue manager name
     D  CDQM                  97    144
     D* Transmission queue name
     D  CDXQ                 145    192
     D* First 20 bytes of connection name
     D  CDSCN                193    212
     D* Reserved
     D  CDMCA                213    232
     D* LU 6.2 Mode name
     D  CDMOD                233    240
     D* LU 6.2 transaction program name
     D  CDTP                 241    304
     D* Batch size
     D  CDBS                 305    308I 0
     D* Disconnect interval
     D  CDDI                 309    312I 0
     D* Short retry count
     D  CDSRC                313    316I 0
     D* Short retry wait interval
     D  CDSRI                317    320I 0
     D* Long retry count
     D  CDLRC                321    324I 0
     D* Long retry wait interval
     D  CDLRI                325    328I 0
     D* Channel security exit name
     D  CDSCX                329    348
     D* Channel message exit name
     D  CDMSX                349    368
     D* Channel send exit name
     D  CDSNX                369    388
     D* Channel receive exit name
     D  CDRCX                389    408
     D* Highest allowable message sequence number
     D  CDSNW                409    412I 0
     D* Maximum message length
     D  CDMML                413    416I 0
     D* Put authority
     D  CDPA                 417    420I 0
     D* Data conversion
     D  CDDC                 421    424I 0
     D* Channel security exit user data
     D  CDSCD                425    456
     D* Channel message exit user data
     D  CDMSD                457    488
     D* Channel send exit user data
     D  CDSND                489    520
     D* Channel receive exit user data
     D  CDRCD                521    552
     D* Ver:1 **
     D* User identifier
     D  CDUID                553    564
```

```
     D* Password
     D  CDPW                 565    576
     D* First 12 bytes of MCA user identifier
     D  CDAUI                577    588
     D* Message channel agent type
     D  CDCAT                589    592I 0
     D* Connection name
     D  CDCON                593    848
     D  CDCN2                849    856
     D* First 12 bytes of user identifier from partner
     D  CDRUI                857    868
     D* Password from partner
     D  CDRPW                869    880
     D* Ver:2 **
     D* Channel message retry exit name
     D  CDMRX                881    900
     D* Channel message retry exit user data
     D  CDMRD                901    932
     D* Number of times MCA will try to put the message, after first
     D* attempt has failed
     D  CDMRC                933    936I 0
     D* Minimum interval in milliseconds after which the open or put
     D* operation will be retried
     D  CDMRI                937    940I 0
     D* Ver:3 **
     D* Time in seconds between heartbeat flows
     D  CDHBI                941    944I 0
     D* Batch duration
     D  CDBI                 945    948I 0
     D* Speed at which nonpersistent messages are sent
     D  CDNPM                949    952I 0
     D* Length of MQCD structure
     D  CDLEN                953    956I 0
     D* Length of exit name
     D  CDXNL                957    960I 0
     D* Length of exit user data
     D  CDXDL                961    964I 0
     D* Number of message exits defined
     D  CDMXD                965    968I 0
     D* Number of send exits defined
     D  CDSXD                969    972I 0
     D* Number of receive exits defined
     D  CDRXD                973    976I 0
     D* Address of first MsgExit field
     D  CDMXP                977    992*
     D* Address of first MsgUserData field
     D  CDMUP                993   1008*
     D* Address of first SendExit field
     D  CDSXP               1009   1024*
     D* Address of first SendUserData field
     D  CDSUP               1025   1040*
     D* Address of first ReceiveExit field
     D  CDRXP               1041   1056*
     D* Address of first ReceiveUserData field
     D  CDRUP               1057   1072*
     D* Ver:4 **
     D* Address of a list of cluster names
     D  CDCLP               1073   1088*
     D* Number of clusters to which the channel belongs
     D  CDCLD               1089   1092I 0
     D* Network priority
     D  CDNP                1093   1096I 0
     D* Ver:5 **
     D* Length of long MCA user identifier
     D  CDLML               1097   1100I 0
     D* Length of long remote user identifier
     D  CDLRL               1101   1104I 0
     D* Address of long MCA user identifier
     D  CDLMP               1105   1120*
     D* Address of long remote user identifier
     D  CDLRP               1121   1136*
     D* MCA security identifier
     D  CDMSI               1137   1176
     D* Remote security identifier
     D  CDRSI               1177   1216
     D* Ver:6 **
     D* TLS CipherSpec
     D  CDSCS               1217   1248
     D* Address of TLS peer name
     D  CDSPN               1249   1264*
     D* Length of TLS peer name
     D  CDSPL               1265   1268I 0
```

```
     D* Whether TLS client authentication is required
     D  CDSCA               1269   1272I 0
     D* Keepalive interval
     D  CDKAI               1273   1276I 0
     D* Local communications address
     D  CDLOA               1277   1324
     D* Batch heartbeat interval
     D  CDBHB               1325   1328I 0
     D* Ver:7 **
     D* Header data compression list
     D  CDHCL0
     D  CDHCL1              1329   1332I 0
     D  CDHCL2              1333   1336I 0
     D  CDHCL                        10I 0 DIM(2) OVERLAY(CDHCL0)
     D* Message data compression list
     D  CDMCL0
     D  CDMCL1              1337   1340I 0
     D  CDMCL2              1341   1344I 0
     D  CDMCL3              1345   1348I 0
     D  CDMCL4              1349   1352I 0
     D  CDMCL5              1353   1356I 0
     D  CDMCL6              1357   1360I 0
     D  CDMCL7              1361   1364I 0
     D  CDMCL8              1365   1368I 0
     D  CDMCL9              1369   1372I 0
     D  CDMCL10             1373   1376I 0
     D  CDMCL11             1377   1380I 0
     D  CDMCL12             1381   1384I 0
     D  CDMCL13             1385   1388I 0
     D  CDMCL14             1389   1392I 0
     D  CDMCL15             1393   1396I 0
     D  CDMCL16             1397   1400I 0
     D  CDMCL                        10I 0 DIM(16) OVERLAY(CDMCL0)
     D* Channel rank
     D  CDCWCR              1401   1404I 0
     D* Channel priority
     D  CDCWCP              1405   1408I 0
     D* Channel weight
     D  CDCWCW              1409   1412I 0
     D* Channel monitoring
     D  CDCHLMON            1413   1416I 0
     D* Channel statistics
     D  CDCHLST             1417   1420I 0
     D* Ver:8 **
     D* Limit on sharing conversations
     D  CDSHC               1421   1424I 0
     D* Message property control
     D  CDPRC               1425   1428I 0
     D* Limit on SVRCONN channel instances
     D  CDMXIN              1429   1432I 0
     D* Limit on SVRCONN channel instances per client
     D  CDMXIC              1433   1436I 0
     D* Client channel weight
     D  CDCLNCHLW           1437   1440I 0
     D* Connection affinity
     D  CDCONNAFF           1441   1444I 0
     D* Ver:9 **
     D* Batch data limit
     D  CDBDL               1445   1448I 0
     D* Use Dead Letter Queue
     D  CDUDLQ              1449   1452I 0
     D* Default client reconnect option
     D  CDDRCN              1453   1456I 0
     D* Ver:10 **
```

Dichiarazione assembler System/390

Questa dichiarazione è la dichiarazione dell'assembler System/390 per la struttura MQCD.

```
MQCD                          DSECT
MQCD_CHANNELNAME              DS   CL20   Channel definition name
MQCD_VERSION                  DS   F      Structure version number
MQCD_CHANNELTYPE              DS   F      Channel type
MQCD_TRANSPORTTYPE            DS   F      Transport type
MQCD_DESC                     DS   CL64   Channel description
MQCD_QMGRNAME                 DS   CL48   Queue manager name
MQCD_XMITQNAME                DS   CL48   Transmission queue name
MQCD_SHORTCONNECTIONNAME      DS   CL20   First 20 bytes of connection
*                                         name
```

```
MQCD_MCANAME                  DS   CL20   Reserved
MQCD_MODENAME                 DS   CL8    LU 6.2 Mode name
MQCD_TPNAME                   DS   CL64   LU 6.2 transaction program name
MQCD_BATCHSIZE                DS   F      Batch size
MQCD_DISCINTERVAL             DS   F      Disconnect interval
MQCD_SHORTRETRYCOUNT          DS   F      Short retry count
MQCD_SHORTRETRYINTERVAL       DS   F      Short retry wait interval
MQCD_LONGRETRYCOUNT           DS   F      Long retry count
MQCD_LONGRETRYINTERVAL        DS   F      Long retry wait interval
MQCD_SECURITYEXIT             DS   CLn    Channel security exit name
MQCD_MSGEXIT                  DS   CLn    Channel message exit name
MQCD_SENDEXIT                 DS   CLn    Channel send exit name
MQCD_RECEIVEEXIT              DS   CLn    Channel receive exit name
MQCD_SEQNUMBERWRAP            DS   F      Highest allowable message
*                                         sequence number
MQCD_MAXMSGLENGTH             DS   F      Maximum message length
MQCD_PUTAUTHORITY             DS   F      Put authority
MQCD_DATACONVERSION           DS   F      Data conversion
MQCD_SECURITYUSERDATA         DS   CL32   Channel security exit user data
MQCD_MSGUSERDATA              DS   CL32   Channel message exit user data
MQCD_SENDUSERDATA             DS   CL32   Channel send exit user data
MQCD_RECEIVEUSERDATA          DS   CL32   Channel receive exit user data
MQCD_USERIDENTIFIER           DS   CL12   User identifier
MQCD_PASSWORD                 DS   CL12   Password
MQCD_MCAUSERIDENTIFIER        DS   CL12   First 12 bytes of MCA user
*                                         identifier
MQCD_MCATYPE                  DS   F      Message channel agent type
MQCD_CONNECTIONNAME           DS   CL264  Connection name
MQCD_REMOTEUSERIDENTIFIER     DS   CL12   First 12 bytes of user
*                                         identifier from partner
MQCD_REMOTEPASSWORD           DS   CL12   Password from partner
MQCD_MSGRETRYEXIT             DS   CLn    Channel message retry exit name
MQCD_MSGRETRYUSERDATA         DS   CL32   Channel message retry exit user
*                                         data
MQCD_MSGRETRYCOUNT            DS   F      Number of times MCA will try to
*                                         put the message, after the
*                                         first attempt has failed
MQCD_MSGRETRYINTERVAL         DS   F      Minimum interval in
*                                         milliseconds after which the
*                                         open or put operation will be
*                                         retried
MQCD_HEARTBEATINTERVAL        DS   F      Time in seconds between
*                                         heartbeat flows
MQCD_BATCHINTERVAL            DS   F      Batch duration
MQCD_NONPERSISTENTMSGSPEED    DS   F      Speed at which nonpersistent
*                                         messages are sent
MQCD_STRUCLENGTH              DS   F      Length of MQCD structure
MQCD_EXITNAMELENGTH           DS   F      Length of exit name
MQCD_EXITDATALENGTH           DS   F      Length of exit user data
MQCD_MSGEXITSDEFINED          DS   F      Number of message exits defined
MQCD_SENDEXITSDEFINED         DS   F      Number of send exits defined
MQCD_RECEIVEEXITSDEFINED      DS   F      Number of receive exits defined
MQCD_MSGEXITPTR               DS   F      Address of first MSGEXIT field
MQCD_MSGUSERDATAPTR           DS   F      Address of first MSGUSERDATA
*                                         field
MQCD_SENDEXITPTR              DS   F      Address of first SENDEXIT field
MQCD_SENDUSERDATAPTR          DS   F      Address of first SENDUSERDATA
*                                         field
MQCD_RECEIVEEXITPTR           DS   F      Address of first RECEIVEEXIT
*                                         field
MQCD_RECEIVEUSERDATAPTR       DS   F      Address of first
*                                         RECEIVEUSERDATA field
MQCD_CLUSTERPTR               DS   F      Address of a list of cluster
*                                         names
MQCD_CLUSTERSDEFINED          DS   F      Number of clusters to which the
*                                         channel belongs
MQCD_NETWORKPRIORITY          DS   F      Network priority
MQCD_LONGMCAUSERIDLENGTH      DS   F      Length of long MCA user
*                                         identifier
MQCD_LONGREMOTEUSERIDLENGTH   DS   F      Length of long remote user
*                                         identifier
MQCD_LONGMCAUSERIDPTR         DS   F      Address of long MCA user
*                                         identifier
MQCD_LONGREMOTEUSERIDPTR      DS   F      Address of long remote user
*                                         identifier
MQCD_MCASECURITYID            DS   XL40   MCA security identifier
MQCD_REMOTESECURITYID         DS   XL40   Remote security identifier
MQCD_SSLCIPHERSPEC            DS   CL32   TLS CipherSpec
MQCD_SSLPEERNAMEPTR           DS   F      Address of TLS peer name
MQCD_SSLPEERNAMELENGTH        DS   F      Length of TLS peer name
MQCD_SSLCLIENTAUTH            DS   F      Whether TLS client
*                                         authentication is required
```

```
MQCD_KEEPALIVEINTERVAL       DS   F      Keepalive interval
MQCD_LOCALADDRESS            DS   CL48   Local communications address
MQCD_BATCHHEARTBEAT          DS   F      Batch heartbeat interval
MQCD_HDRCOMPLIST             DS   CL2    Header data compression list
MQCD_MSGCOMPLIST             DS   CL16   Message data compression list
MQCD_CLWLCHANNELRANK         DS   F      Channel rank
MQCD_CLWLCHANNELPRIORITY     DS   F      Channel priority
MQCD_CLWLCHANNELWEIGHT       DS   F      Channel weight
MQCD_CHANNELMONITORING       DS   F      Channel monitoring
MQCD_CHANNELSTATISTICS       DS   F      Channel statistics
MQCD_SHARINGCONVERSATIONS    DS   F      Limit on sharing
*                                        conversations
MQCD_PROPERTYCONTROL         DS   F      Message property
*                                        control
MQCD_SHARINGCONVERSATIONS    DS   F      Limit on sharing conversations
MQCD_PROPERTYCONTROL         DS   F      Message property control
MQCD_MAXINSTANCES            DS   F      Limit on SVRCONN chl instances
MQCD_MAXINSTANCESPERCLIENT   DS   F      Limit on SVRCONN chl instances
                                         per client
MQCD_CLIENTCHANNELWEIGHT     DS   F      Channel weight
MQCD_CONNECTIONAFFINITY      DS   F      Connection Affinty
MQCD_BATCHDATALIMIT          DS   F      Batch data limit
MQCD_USEDLQ                  DS   F      Use dead-letter queue
MQCD_DEFRECONNECT            DS   F      Default client reconnect option
MQCD_CERTLABL                DS   F      Certificate label
MQCD_SPLPROTECTION           DS   F      AMS Security policy protection
MQCD_LENGTH                  EQU  *-MQCD
                             ORG  MQCD
MQCD_AREA                    DS   CL(MQCD_LENGTH)
```

Dichiarazione Visual Basic

Questa dichiarazione è la dichiarazione Visual Basic della struttura MQCD.

In Visual Basic, la struttura MQCD può essere utilizzata con la struttura MQCNO sulla chiamata MQCONNX.

```
Type MQCD
  ChannelName            As String*20  'Channel definition name'
  Version                As Long       'Structure version number'
  ChannelType            As Long       'Channel type'
  TransportType          As Long       'Transport type'
  Desc                   As String*64  'Channel description'
  QMgrName               As String*48  'Queue manager name'
  XmitQName              As String*48  'Transmission queue name'
  ShortConnectionName    As String*20  'First 20 bytes of connection'
                                       'name'
  MCAName                As String*20  'Reserved'
  ModeName               As String*8   'LU 6.2 Mode name'
  TpName                 As String*64  'LU 6.2 transaction program name'
  BatchSize              As Long       'Batch size'
  DiscInterval           As Long       'Disconnect interval'
  ShortRetryCount        As Long       'Short retry count'
  ShortRetryInterval     As Long       'Short retry wait interval'
  LongRetryCount         As Long       'Long retry count'
  LongRetryInterval      As Long       'Long retry wait interval'
  SecurityExit           As String*128 'Channel security exit name'
  MsgExit                As String*128 'Channel message exit name'
  SendExit               As String*128 'Channel send exit name'
  ReceiveExit            As String*128 'Channel receive exit name'
  SeqNumberWrap          As Long       'Highest allowable message'
                                       'sequence number'
  MaxMsgLength           As Long       'Maximum message length'
  PutAuthority           As Long       'Put authority'
  DataConversion         As Long       'Data conversion'
  SecurityUserData       As String*32  'Channel security exit user data'
  MsgUserData            As String*32  'Channel message exit user data'
  SendUserData           As String*32  'Channel send exit user data'
  ReceiveUserData        As String*32  'Channel receive exit user data'
  UserIdentifier         As String*12  'User identifier'
  Password               As String*12  'Password'
  MCAUserIdentifier      As String*12  'First 12 bytes of MCA user'
                                       'identifier'
  MCAType                As Long       'Message channel agent type'
  ConnectionName         As String*264 'Connection name'
  RemoteUserIdentifier   As String*12  'First 12 bytes of user'
                                       'identifier from partner'
  RemotePassword         As String*12  'Password from partner'
  MsgRetryExit           As String*128 'Channel message retry exit name'
```

```
  MsgRetryUserData       As String*32  'Channel message retry exit user'
                                       'data'
  MsgRetryCount          As Long       'Number of times MCA will try to'
                                       'put the message, after the'
                                       'first attempt has failed'
  MsgRetryInterval       As Long       'Minimum interval in'
                                       'milliseconds after which the'
                                       'open or put operation will be'
                                       'retried'
  HeartbeatInterval      As Long       'Time in seconds between'
                                       'heartbeat flows'
  BatchInterval          As Long       'Batch duration'
  NonPersistentMsgSpeed  As Long       'Speed at which nonpersistent'
                                       'messages are sent'
  StrucLength            As Long       'Length of MQCD structure'
  ExitNameLength         As Long       'Length of exit name'
  ExitDataLength         As Long       'Length of exit user data'
  MsgExitsDefined        As Long       'Number of message exits defined'
  SendExitsDefined       As Long       'Number of send exits defined'
  ReceiveExitsDefined    As Long       'Number of receive exits defined'
  MsgExitPtr             As MQPTR      'Address of first MsgExit field'
  MsgUserDataPtr         As MQPTR      'Address of first MsgUserData'
                                       'field'
  SendExitPtr            As MQPTR      'Address of first SendExit field'
  SendUserDataPtr        As MQPTR      'Address of first SendUserData'
                                       'field'
  ReceiveExitPtr         As MQPTR      'Address of first ReceiveExit'
                                       'field'
  ReceiveUserDataPtr     As MQPTR      'Address of first'
                                       'ReceiveUserData field'
  ClusterPtr             As MQPTR      'Address of a list of cluster'
                                       'names'
  ClustersDefined        As Long       'Number of clusters to which the'
                                       'channel belongs'
  NetworkPriority        As Long       'Network priority'
  LongMCAUserIdLength    As Long       'Length of long MCA user'
                                       'identifier'
  LongRemoteUserIdLength As Long       'Length of long remote user'
                                       'identifier'
  LongMCAUserIdPtr       As MQPTR      'Address of long MCA user'
                                       'identifier'
  LongRemoteUserIdPtr    As MQPTR      'Address of long remote user'
                                       'identifier'
  MCASecurityId          As MQBYTE40   'MCA security identifier'
  RemoteSecurityId       As MQBYTE40   'Remote security identifier'
  SSLCipherSpec          As String*32  'TLS CipherSpec'
  SSLPeerNamePtr         As MQPTR      'Address of TLS peer name'
  SSLPeerNameLength      As Long       'Length of TLS peer name'
  SSLClientAuth          As Long       'Whether TLS client'
                                       'authentication is required'
  KeepAliveInterval      As Long       'Keepalive interval'
  LocalAddress           As String*48  'Local communications address'
  BatchHeartbeat         As Long       'Batch heartbeat interval'
  HdrCompList(0 to 1)    As Long2      'Header data compression list'
  MsgCompList(0 To 15)   As Long16     'Message data compression list'
  CLWLChannelRank        As Long       'Channel Rank'
  CLWLChannelPriority    As Long       'Channel priority'
  CLWLChannelWeight      As Long       'Channel Weight'
  ChannelMonitoring      As Long       'Channel Monitoring control'
  ChannelStatistics      As Long       'Channel Statistics'
End Type
```

Modifica dei campi MQCD in un'uscita canale

Un'uscita canale può cambiare i campi in MQCD. Tuttavia, queste modifiche non vengono generalmente applicate, ad eccezione delle circostanze elencate.

Se un programma di uscita del canale modifica un campo nella struttura dati MQCD, il nuovo valore viene generalmente ignorato dal processo del canale IBM MQ . Tuttavia, il nuovo valore rimane nel MQCD e viene passato a tutte le uscite rimanenti in una catena di uscita e a tutte le conversazioni che condividono l'istanza del canale.

Se SharingConversations è impostato su FALSE nella struttura MQCXP, le modifiche a determinati campi possono essere eseguite, a seconda del tipo di programma di uscita, del tipo di canale e del codice di errore di uscita. La seguente tabella mostra i campi che possono essere modificati e influenzano il comportamento del canale e in quali circostanze. Se un programma di uscita modifica uno di questi campi in qualsiasi altra circostanza o qualsiasi campo non elencato, il nuovo valore viene ignorato dal processo

del canale. Il nuovo valore rimane nel MQCD e viene passato a tutte le uscite rimanenti in una catena di uscita e a tutte le conversazioni che condividono l'istanza del canale.

Qualsiasi tipo di programma di uscita quando richiamato per l'inizializzazione (MQXR_INIT) può modificare il campo ChannelName di qualsiasi tipo di canale, purché MQCXP SharingConverstions sia impostato su FALSE. Solo un'uscita di sicurezza può modificare il campo MCAUserIdentifier , indipendentemente dal valore di MQCXP SharingConverstions.

Tabella 823. Campi che possono essere modificati e influenzare il comportamento del canale

Campo	**Codice di errore di uscita**	**Tipo di uscita**	**Tipo di canale**
ChannelName	INIT MQXR	Tutto	Tutto
TransportType	INIT MQXR	Tutto	Tutto
XmitQName	INIT MQXR	Tutto	SDR, RCVR
ModeName	INIT MQXR	Tutto	Tutto
TpName	INIT MQXR	Tutto	Tutto
BatchSize	INIT MQXR	Tutto	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
DiscInterval	INIT MQXR	Tutto	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Conteggio ShortRetry	INIT MQXR	Tutto	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
shortRetryInterval	INIT MQXR	Tutto	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Conteggio LongRetry	INIT MQXR	Tutto	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
longRetryInterval	INIT MQXR	Tutto	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR

Tabella 823. Campi che possono essere modificati e influenzare il comportamento del canale (Continua)

Campo	Codice di errore di uscita	Tipo di uscita	Tipo di canale
Wrapper SeqNumber	INIT MQXR	Tutto	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
MaxMsgLength	INIT MQXR	Tutto	Tutto
PutAuthority	INIT MQXR	Tutto	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
DataConversion	INIT MQXR	Tutto	Tutto
MCAUserIdentifier	MQXR_INIT, MQXR_INIT_SEC, MQXR_SEC_MSG, MQXR_SEC_PARMS	Sicurezza	RCVR, RQSTR, SVRCONN, CLUSRCVR
ConnectionName	INIT MQXR	Tutto	SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, CLUSRCVR
MsgRetryUserData	INIT MQXR	Tutto	RCVR, RQSTR, CLUSRCVR
Conteggio MsgRetry	INIT MQXR	Tutto	RCVR, RQSTR, CLUSRCVR
Intervallo MsgRetry	INIT MQXR	Tutto	RCVR, RQSTR, CLUSRCVR
HeartbeatInterval	INIT MQXR	Tutto	Tutto
BatchInterval	INIT MQXR	Tutto	SDR, SVR, CLUSSDR, CLUSRCVR
NonPersistentMsgSpeed	INIT MQXR	Tutto	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
MCASecurityId	MQXR_INIT, MQXR_INIT_SEC, MQXR_SEC_MSG, MQXR_SEC_PARMS	Sicurezza	SDR, SVR, RCVR, RQSTR, SVRCONN, CLUSSDR, CLUSRCVR

Tabella 823. Campi che possono essere modificati e influenzare il comportamento del canale (Continua)

Campo	Codice di errore di uscita	Tipo di uscita	Tipo di canale
SSLCipherSpec	INIT MQXR	Tutto	Tutto
Ptr SSLPeerName	INIT MQXR	Tutto	Tutto
SSLPeerNameLunghezza	INIT MQXR	Tutto	Tutto
SSLClientAuth	INIT MQXR	Tutto	SVR, RCVR, RQSTR, SVRCONN, CLUSRCVR
KeepAliveInterval	INIT MQXR	Tutto	Tutto
LocalAddress	INIT MQXR	Tutto	SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, CLUSRCVR
BatchHeartbeat	INIT MQXR	Tutto	SDR, SVR, CLUSSDR, CLUSRCVR
Elenco HdrComp	INIT MQXR	Tutto	Tutto
Elenco MsgComp	INIT MQXR	Tutto	Tutto
ChannelMonitoring	INIT MQXR	Tutto	SDR, SVR, RCVR, RQSTR, SVRCONN, CLUSSDR, CLUSRCVR
ChannelStatistics	INIT MQXR	Tutto	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
SharingConversations	INIT MQXR	Tutto	SVRCONN, CLNTCONN
PropertyControl	INIT MQXR	Tutto	SDR, SVR, CLUSSDR, CLUSRCVR

MQCXP - Parametro uscita canale

La struttura MQCXP viene passata a ciascun tipo di exit richiamato da un MCA (Message Channel Agent), da un canale di connessione client o da un canale di connessione server.

Vedere MQ_CHANNEL_EXIT.

I campi descritti come "input all'uscita" nelle descrizioni che seguono vengono ignorati dal canale quando l'uscita restituisce il controllo al canale. I campi di input che l'uscita modifica nel blocco del parametro di uscita del canale non verranno conservati per il successivo richiamo. Le modifiche apportate ai campi di input / output (ad esempio, il campo *ExitUserArea*), vengono conservate solo per i richiami di tale

istanza dell'exit. Tali modifiche non possono essere utilizzate per passare i dati tra diverse uscite definite sullo stesso canale o tra la stessa uscita definita su canali differenti.

Riferimenti correlati

“Campi” a pagina 1563
Questo argomento elenca tutti i campi nella struttura MQCXP e descrive ciascun campo.

“Dichiarazione C” a pagina 1574
Questa dichiarazione è la dichiarazione C per la struttura MQCXP.

“Dichiarazione COBOL” a pagina 1575
Questa dichiarazione è la dichiarazione COBOL per la struttura MQCXP.

“Dichiarazione RPG (ILE)” a pagina 1576
Questa dichiarazione è la dichiarazione RPG per la struttura MQCXP.

“Dichiarazione assembler System/390” a pagina 1577
Questa dichiarazione è la dichiarazione dell'assembler System/390 per la struttura MQCXP.

Campi

Questo argomento elenca tutti i campi nella struttura MQCXP e descrive ciascun campo.

StrucId (MQCHAR4)
Questo campo specifica l'identificativo della struttura.

Il valore deve essere:

ID_STRUC_MQCXP
Identificativo per la struttura del parametro di uscita canale.

Per il linguaggio di programmazione C, viene definita anche la costante MQCXP_STRUC_ID_ARRAY; questa costante ha lo stesso valore di MQCXP_STRUC_ID, ma è un array di caratteri anziché una stringa.

Questo è un campo di immissione per l'uscita.

Version (MQLONG)
Questo campo specifica il numero di versione della struttura.

Il valore dipende dall'ambiente:

MQCXP_VERSION_1
Struttura del parametro di uscita canale Version-1 .

MQCXP_VERSION_2
Struttura del parametro di uscita canale Version-2 .

MQCXP_VERSION_3
Struttura del parametro di uscita del canale Version-3 .

Linux AIX Il campo ha questo valore nei sistemi AIX and Linux non elencati altrove.

MQCXP_VERSION_4
Struttura del parametro di uscita canale Version-4 .

MQCXP_VERSION_5
Struttura del parametro di uscita canale Version-5 .

MQCXP_VERSION_6
Struttura del parametro di uscita del canale Version-6 .

MQCXP_VERSION_8
Struttura del parametro di uscita del canale Version-8 .

z/OS Il campo ha questo valore in z/OS.

MQCXP_VERSION_9
Struttura del parametro di uscita del canale Version-9 .

Il campo ha questo valore nei seguenti ambienti:

- 

AIX
- 

IBM i
- 

Linux
- 

Windows
- 

z/OS

I campi che esistono solo nelle versioni più recenti della struttura vengono identificati come tali nelle descrizioni dei campi. La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQCXP_CURRENT_
Versione corrente della struttura del parametro di uscita del canale.

Il valore dipende dall'ambiente.

Nota: Quando viene introdotta una nuova versione della struttura di MQCXP, il layout della parte esistente non viene modificato. L'uscita deve quindi controllare che il numero di versione sia uguale o superiore alla versione più bassa che contiene i campi che l'uscita deve utilizzare.

Questo è un campo di immissione per l'uscita.

ExitId (MQLONG)
Questo campo specifica il tipo di uscita che si sta richiamando e viene impostato all'entrata della routine di uscita.

Sono possibili i seguenti valori:

MQXT_CHANNEL_SEC_EXIT
Uscita di sicurezza del canale.

MQXT_CHANNEL_MSG_EXIT
Uscita messaggio canale.

MQXT_CHANNEL_SEND_EXIT
Uscita di invio canale.

MQXT_CHANNEL_RCV_EXIT
Uscita ricezione canale.

MQXT_CHANNEL_MSG_RETRY_EXIT
Uscita nuovo tentativo messaggio canale.

MQXT_CHANNEL_AUTO_DEF_EXIT
Uscita di definizione automatica del canale.

Su z/OS, questo tipo di exit è supportato solo per canali di tipo MQCHT_CLUSSDR e MQCHT_CLUSRCVR.

Questo è un campo di immissione per l'uscita.

ExitReason (MQLONG)
Questo campo specifica il motivo per cui l'uscita viene chiamata ed è impostata all'entrata della routine di uscita.

Non viene utilizzato dall'uscita di definizione automatica. Sono possibili i seguenti valori:

INIT MQXR
Uscire dall'inizializzazione.

Questo valore indica che l'uscita viene richiamata per la prima volta. Consente all'uscita di acquisire e inizializzare tutte le risorse di cui ha bisogno (ad esempio: memoria).

MQXR_TERM
Chiusura uscita.

Questo valore indica che l'uscita sta per essere terminata. L'uscita deve liberare tutte le risorse che ha acquisito da quando è stata inizializzata (ad esempio: memoria).

MQXR_MSG
Elaborare un messaggio.

Questo valore indica che l'uscita viene richiamata per l'elaborazione di un messaggio. Questo valore si verifica solo per le uscite dei messaggi del canale.

XMIT MQXR
Elaborare una trasmissione.

Questo valore si verifica solo per le uscite di invio e ricezione del canale.

MQXR_SEC_MSG
Messaggio di sicurezza ricevuto.

Questo valore si verifica solo per le uscite di sicurezza del canale.

MQXR_INIT_SEC
Avviare lo scambio di sicurezza.

Questo valore si verifica solo per le uscite di sicurezza del canale.

L'uscita di sicurezza del destinatario viene sempre richiamata con questo motivo immediatamente dopo essere stata richiamata con MQXR_INIT, per dargli la possibilità di avviare uno scambio di sicurezza. Se declina l'opportunità (restituendo MQXCC_OK invece di MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG), l'uscita di sicurezza del mittente viene richiamata con MQXR_INIT_SEC.

Se l'uscita di sicurezza del destinatario non avvia uno scambio di sicurezza (restituendo MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG), l'uscita di sicurezza del mittente non viene mai richiamata con MQXR_INIT_SEC; viene invece richiamata con MQXR_SEC_MSG per elaborare il messaggio del ricevente. (In entrambi i casi viene richiamato per la prima volta con MQXR_INIT.)

A meno che una delle uscite di sicurezza non richieda la terminazione del canale (impostando *ExitResponse* su MQXCC_SUPPRESS_FUNCTION o MQXCC_CLOSE_CHANNEL), lo scambio di sicurezza deve essere completato sul lato che ha avviato lo scambio. Pertanto, se un'uscita di sicurezza viene richiamata con MQXR_INIT_SEC e avvia uno scambio, la volta successiva che l'uscita viene richiamata sarà con MQXR_SEC_MSG. Ciò si verifica se è presente un messaggio di sicurezza per l'uscita da elaborare o meno. Esiste un messaggio di sicurezza se il partner restituisce MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG, ma non se il partner restituisce MQXCC_OK o se non esiste alcuna uscita di sicurezza sul partner. Se non esiste alcun messaggio di sicurezza da elaborare, l'uscita di sicurezza all'estremità di avvio viene richiamata con un *DataLength* uguale a zero.

MQXR_RETRY
Riprovare un messaggio.

Questo valore si verifica solo per le uscite di nuovi tentativi di messaggio.

MQXR_AUTO_CLUSSDR
Definizione automatica di un canale mittente del cluster.

Questo valore si verifica solo per le uscite di definizione automatica del canale.

MQXR_AUTO_RECEIVER
Definizione automatica di un canale ricevente.

Questo valore si verifica solo per le uscite di definizione automatica del canale.

SVRCONN MQXR_AUTO_
Definizione automatica di un canale di connessione server.

Questo valore si verifica solo per le uscite di definizione automatica del canale.

MQXR_AUTO_CLUSRCVR
Definizione automatica di un canale ricevente del cluster.

Questo valore si verifica solo per le uscite di definizione automatica del canale.

PARM_SEC_MQXR
Parametri di sicurezza

Questo valore si applica solo alle uscite di protezione e indica che una struttura MQCSP viene passata all'uscita. Per ulteriori informazioni, vedi "MQCSP - Parametri di sicurezza" a pagina 340

Nota:

1. Se si dispone di più di un'uscita definita per un canale, vengono richiamate con MQXR_INIT quando l'MCA viene inizializzato. Inoltre, vengono richiamati con MQXR_TERM quando l'MCA viene terminato.
2. Per l'uscita di definizione automatica del canale, *ExitReason* non è impostato se *Version* è inferiore a MQCXP_VERSION_4. Il valore MQXR_AUTO_SVRCONN è implicito in questo caso.

Questo è un campo di immissione per l'uscita.

ExitResponse (MQLONG)
Questo campo specifica la risposta dall'uscita.

Questo campo viene impostato dall'uscita per comunicare con l'MCA. Deve essere uno dei seguenti valori:

MQXCC_OK
Uscita completata correttamente.

- Per l'uscita di sicurezza del canale, questo valore indica che il trasferimento del messaggio può ora procedere normalmente.
- Per l'uscita del nuovo tentativo di messaggio del canale, questo valore indica che l'MCA deve attendere l'intervallo di tempo restituito dall'uscita nel campo *MsgRetryInterval* in MQCXP, quindi ripetere il messaggio.

Il campo *ExitResponse2* potrebbe contenere ulteriori informazioni.

MQXCC_SUPPRESS_FUNZIONE
Sopprimere la funzione.

- Per l'uscita di sicurezza del canale, questo valore indica che il canale deve essere terminato.
- Per l'uscita del messaggio del canale, questo valore indica che il messaggio non deve procedere ulteriormente verso la sua destinazione. Invece, l'MCA genera un messaggio di report di eccezione (se richiesto dal mittente del messaggio originale) e colloca il messaggio contenuto nel buffer originale nella coda di messaggi non instradabili (se il mittente ha specificato MQRO_DEAD_LETTER_Q) o lo elimina (se il mittente ha specificato MQRO_DISCARD_MSG).

  Per i messaggi persistenti, se il mittente ha specificato MQRO_DEAD_LETTER_Q, ma l'operazione di inserimento nella coda di messaggi non recapitabili ha esito negativo o non è presente alcuna coda di messaggi non recapitabili, il messaggio originale viene lasciato nella coda di trasmissione e il messaggio di report non viene generato. Il messaggio originale viene lasciato anche nella coda di trasmissione se non è possibile generare correttamente il messaggio di report.

  Il campo *Feedback* nella struttura MQDLH all'inizio del messaggio sulla coda di messaggi non instradabili indica il motivo per cui il messaggio è stato inserito nella coda di messaggi non instradabili; questo codice di feedback viene utilizzato anche nel descrittore del messaggio di report dell'errore (se richiesto dal mittente).
- Per l'uscita dei tentativi dei messaggi del canale, questo valore indica che l'MCA non attende e ritenta il messaggio; invece, l'MCA continua immediatamente con la normale elaborazione degli errori (il messaggio viene inserito nella coda dei messaggi non recapitabili o eliminato, come specificato dal mittente del messaggio).
- Per l'uscita di definizione automatica del canale, è necessario specificare MQXCC_OK o MQXCC_SUPPRESS_FUNCTION. Se non viene specificato nessuno di questi valori, per impostazione

predefinita viene utilizzato MQXCC_SUPPRESS_FUNCTION e la definizione automatica viene abbandonata.

Questa risposta non è supportata per le uscite di invio e ricezione del canale.

MQXCC_SEND_SEC_MSG
Inviare un messaggio di sicurezza.

Questo valore può essere impostato solo da un'uscita di sicurezza del canale. Indica che l'uscita ha fornito un messaggio di sicurezza che deve essere trasmesso al partner.

MQXCC_SEND_AND_REQUEST_SEC_MSG
Inviare un messaggio di sicurezza che richiede una risposta.

Questo valore può essere impostato solo da un'uscita di sicurezza del canale. Indica

- che l ' uscita ha fornito un messaggio di sicurezza che può essere trasmesso al partner e
- che l'exit richiede una risposta dal partner. Se non viene ricevuta alcuna risposta, il canale deve essere terminato, poiché l'uscita non ha ancora deciso se le comunicazioni possono continuare.

MQXCC_SUPPRESS_EXIT
Elimina uscita.

- Questo valore può essere impostato da tutti i tipi di uscita di canale diversi da un'uscita di sicurezza o da un'uscita di definizione automatica. Elimina qualsiasi ulteriore richiamo di tale uscita (come se il suo nome fosse stato vuoto nella definizione del canale), fino alla chiusura del canale, quando l'uscita viene nuovamente richiamata con un *ExitReason* di MQXR_TERM.
- Se un'uscita di nuovo tentativo del messaggio restituisce questo valore, i nuovi tentativi del messaggio per i messaggi successivi vengono controllati dagli attributi del canale *MsgRetryCount* e *MsgRetryInterval* come di consueto. Per il messaggio corrente, l'MCA esegue il numero di tentativi in sospeso, ad intervalli forniti dall'attributo del canale *MsgRetryInterval* , ma solo se il codice di errore è uno di quelli che l'MCA ritenterebbe normalmente (consultare il campo *MsgRetryCount* descritto in “MQCD - Definizione canale” a pagina 1521). Il numero di tentativi in sospeso è il valore dell'attributo **MsgRetryCount** , meno il numero di volte in cui l'uscita ha restituito MQXCC_OK per il messaggio corrente; se questo numero è negativo, non vengono eseguiti ulteriori tentativi dall'MCA per il messaggio corrente.

MQXCC_CLOSE_CHANNEL
Chiudere il canale.

Questo valore può essere impostato da qualsiasi tipo di uscita del canale ad eccezione di un'uscita di definizione automatica.

Se la condivisione delle conversazioni non è abilitata, questo valore chiude il canale.

Se la condivisione delle conversazioni è abilitata, questo valore termina la conversazione. Se questa conversazione è l'unica conversazione sul canale, anche il canale si chiude.

Questo campo è un campo di immissione / emissione dall'uscita.

ExitResponse2 (MQLONG)
Questo campo specifica la risposta secondaria dall'uscita.

Questo campo è impostato su zero all'entrata della routine di uscita. Può essere impostato dall'uscita per fornire ulteriori informazioni alle funzioni del canale IBM MQ . Non viene utilizzato dall'uscita di definizione automatica.

L'uscita può impostare uno o più dei seguenti valori. Se ne è richiesto più di uno, vengono aggiunti i valori. Vengono annotate le combinazioni non valide; sono consentite altre combinazioni.

MQXR2_PUT_WITH_DEF_ACTION
Inserisci con azione predefinita.

Questo valore viene impostato dall'uscita del messaggio del canale del ricevitore. Indica che il messaggio deve essere inserito con l'azione predefinita dell'MCA, ovvero l'ID utente predefinito dell'MCA o il contesto *UserIdentifier* nell'MQMD (descrittore del messaggio) del messaggio.

Il valore è zero, che corrisponde al valore iniziale impostato quando viene richiamata l'uscita. La costante viene fornita a scopo di documentazione.

MQXR2_PUT_WITH_DEF_USERID
Inserire con l'identificativo utente predefinito.

Questo valore può essere impostato solo dall'uscita del messaggio del canale del ricevitore. Indica che il messaggio deve essere inserito con l'identificativo utente predefinito dell'MCA.

MQXR2_PUT_WITH_MSG_USERID
Inserire con l'identificativo utente del messaggio.

Questo valore può essere impostato solo dall'uscita del messaggio del canale del ricevitore. Indica che il messaggio deve essere inserito con il contesto *UserIdentifier* nel MQMD (message descriptor) del messaggio (potrebbe essere stato modificato dall'uscita).

Deve essere impostato solo uno tra MQXR2_PUT_WITH_DEF_ACTION, MQXR2_PUT_WITH_DEF_USERIDe MQXR2_PUT_WITH_MSG_USERID .

MQXR2_USE_AGENT_BUFFER
Utilizzare il buffer dell'agent.

Questo valore indica che i dati da trasmettere si trovano in *AgentBuffer*, non in *ExitBufferAddr*.

Il valore è zero, che corrisponde al valore iniziale impostato quando viene richiamata l'uscita. La costante viene fornita a scopo di documentazione.

MQXR2_USE_EXIT_BUFFER
Utilizzare buffer di uscita.

Questo valore indica che i dati da trasmettere si trovano in *ExitBufferAddr*, non in *AgentBuffer*.

Solo uno tra MQXR2_USE_AGENT_BUFFER e MQXR2_USE_EXIT_BUFFER deve essere impostato.

MQXR2_DEFAULT_CONTINUATION
Continuazione predefinita.

La continuazione con la successiva uscita nella catena dipende dalla risposta dall'ultima uscita richiamata:

- Se vengono restituiti MQXCC_SUPPRESS_FUNCTION o MQXCC_CLOSE_CHANNEL, non vengono richiamate ulteriori uscite nella catena.
- Altrimenti, viene richiamata l'uscita successiva nella catena.

MQXR2_CONTINUE_CHAIN
Continuare con l'uscita successiva.

MQXR2_SUPPRESS_CHAIN
Ignora le uscite rimanenti nella catena.

Questo è un campo di immissione / emissione per l'uscita.

Feedback (MQLONG)
Questo campo specifica il codice di feedback.

Questo campo è impostato su MQFB_NONE all'entrata della routine di uscita.

Se un'uscita del messaggio del canale imposta il campo *ExitResponse* su MQXCC_SUPPRESS_FUNCTION, il campo *Feedback* specifica il codice di feedback che identifica il motivo per cui il messaggio è stato inserito nella coda dei messaggi non recapitabili (messaggio non recapitato) e viene utilizzato anche per inviare un report di eccezioni se ne è stato richiesto uno. In questo caso, se il campo *Feedback* è MQFB_NONE, viene utilizzato il seguente codice di feedback:

MQFB_STOPPED_BY_MSG_EXIT
Il messaggio è stato arrestato dall'uscita del messaggio di canale.

Il valore restituito in questo campo dalle uscite di sicurezza del canale, invio, ricezione e nuovo tentativo di messaggio non è utilizzato da MCA.

Il valore restituito in questo campo dalle uscite di definizione automatica non viene utilizzato se *ExitResponse* è MQXCC_OK, ma altrimenti viene utilizzato per il parametro *AuxErrorDataInt1* nel messaggio di evento.

Questo è un campo di immissione / emissione dall'uscita.

Lunghezza MaxSegment(MQLONG)
Questo campo specifica la lunghezza massima in byte che può essere inviata in una singola trasmissione.

Non viene utilizzato dall'uscita di definizione automatica. È di interesse per un'uscita di invio del canale, perché questa uscita deve garantire che non aumenti la dimensione di un segmento di trasmissione a un valore maggiore di *MaxSegmentLength*. La lunghezza include gli 8 byte iniziali che l'exit non deve modificare. Il valore viene negoziato tra le funzioni del canale IBM MQ quando il canale viene avviato. Consultare Scrittura dei programmi di uscita del canale per ulteriori informazioni sulle lunghezze dei segmenti.

Il valore in questo campo non è significativo se *ExitReason* è MQXR_INIT.

Questo è un campo di immissione per l'uscita.

Area ExitUser(MQBYTE16)
Questo campo specifica l'area utente di uscita - un campo disponibile per l'uscita da utilizzare.

Viene inizializzato su zero binario prima del primo richiamo dell'exit (che ha un *ExitReason* impostato su MQXR_INIT), e successivamente tutte le modifiche apportate a questo campo dall'exit vengono conservate tra i richiami dell'exit.

Viene definito il seguente valore:

MQXUA_NONE
Nessuna informazione utente.

Il valore è zero binario per la lunghezza del campo.

Per il linguaggio di programmazione C, è definita anche la costante MQXUA_NONE_ARRAY; questa costante ha lo stesso valore di MQXUA_NONE, ma è un array di caratteri invece di una stringa.

La lunghezza di questo campo è fornita da MQ_EXIT_USER_AREA_LENGTH. Questo è un campo di immissione / emissione per l'uscita.

ExitData (MQCHAR32)
Questo campo specifica i dati di uscita.

Questo campo è impostato all'entrata della routine di uscita per le informazioni che le funzioni del canale IBM MQ hanno preso dalla definizione del canale. Se tali informazioni non sono disponibili, questo campo è vuoto.

La lunghezza di questo campo è fornita da MQ_EXIT_DATA_LENGTH.

Questo è un campo di immissione per l'uscita.

I seguenti campi in questa struttura non sono presenti se *Version* è inferiore a MQCXP_VERSION_2.

Conteggio MsgRetry(MQLONG)
Questo campo specifica il numero di volte in cui il messaggio è stato ritentato.

La prima volta che l'uscita viene richiamata per un messaggio particolare, questo campo ha il valore zero (nessun tentativo ancora eseguito). Ad ogni richiamo successivo dell'exit per quel messaggio, il valore viene incrementato di uno dall'MCA.

Questo è un campo di immissione per l'uscita. Il valore in questo campo non è significativo se *ExitReason* è MQXR_INIT. Il campo non è presente se *Version* è minore di MQCXP_VERSION_2.

Intervallo MsgRetry(MQLONG)
Questo campo specifica l'intervallo minimo in millisecondi dopo il quale l'operazione di inserimento viene ritentata.

La prima volta che l'uscita viene richiamata per un messaggio particolare, questo campo contiene il valore dell'attributo del canale *MsgRetryInterval* . L'uscita può lasciare il valore invariato o modificarlo per specificare un intervallo di tempo diverso in millisecondi. Se l'uscita restituisce MQXCC_OK in *ExitResponse*, l'MCA attende almeno questo intervallo di tempo prima di ritentare l'operazione MQOPEN o MQPUT. L'intervallo di tempo specificato deve essere zero o maggiore.

La seconda e le successive volte in cui viene richiamata l'exit per tale messaggio, questo campo contiene il valore restituito dal richiamo precedente dell'exit.

Se il valore restituito nel campo *MsgRetryInterval* è inferiore a zero o superiore a 999 999 999 e *ExitResponse* è MQXCC_OK, l'MCA ignora il campo *MsgRetryInterval* in MQCXP e attende invece l'intervallo specificato dall'attributo del canale *MsgRetryInterval* .

Questo è un campo di immissione / emissione per l'uscita. Il valore in questo campo non è significativo se *ExitReason* è MQXR_INIT. Il campo non è presente se *Version* è minore di MQCXP_VERSION_2.

Motivo MsgRetry(MQLONG)
Questo campo specifica il codice di errore del precedente tentativo di inserire il messaggio.

Questo campo è il codice motivo del precedente tentativo di inserire il messaggio; è uno dei valori MQRC_ *.

Questo è un campo di immissione per l'uscita. Il valore in questo campo non è significativo se *ExitReason* è MQXR_INIT. Il campo non è presente se *Version* è minore di MQCXP_VERSION_2.

I seguenti campi in questa struttura non sono presenti se *Version* è inferiore a MQCXP_VERSION_3.

HeaderLength (MQLONG)
Questo campo specifica la lunghezza delle informazioni di intestazione.

Questo campo è rilevante solo per un'uscita messaggio e un'uscita nuovo tentativo messaggio. Il valore è la lunghezza delle strutture dell'intestazione di instradamento all'inizio dei dati del messaggio; si tratta della struttura MQXQH, di MQMDE (intestazione di estensione della descrizione del messaggio) e (per un messaggio di elenco di distribuzione) della struttura MQDH e degli array di record MQOR e MQPMR che seguono la struttura MQXQH.

L'uscita messaggio può esaminare queste informazioni di intestazione e modificarle se necessario, ma i dati restituiti dall'uscita devono essere ancora nel formato corretto. L'uscita non deve, ad esempio, codificare o comprimere i dati di intestazione all'estremità di invio, anche se l'uscita del messaggio all'estremità di ricezione effettua modifiche di compensazione.

Se l'uscita del messaggio modifica le informazioni dell'intestazione in modo da modificarne la lunghezza (ad esempio, aggiungendo un'altra destinazione a un messaggio dell'elenco di distribuzione), deve modificare il valore di *HeaderLength* in modo corrispondente prima di ritornare.

Questo è un campo di immissione / emissione per l'uscita. Il valore in questo campo non è significativo se *ExitReason* è MQXR_INIT. Il campo non è presente se *Version* è minore di MQCXP_VERSION_3.

PartnerName (MQCHAR48)
Questo campo specifica il nome del partner.

Il nome del partner, come segue:

- Per i canali SVRCONN, è l'ID utente collegato al client.
- Per tutti gli altri tipi di canale, è il nome del gestore code del partner.

Quando l'uscita viene inizializzata, questo campo è vuoto perché il gestore code non conosce il nome del partner fino a quando non ha avuto luogo la negoziazione iniziale.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCXP_VERSION_3.

Livello API (MQLONG)
Livello di protocolli e formati negoziati.

Questo è un campo di immissione per l'uscita. Le modifiche a questo campo devono essere apportate solo sotto la direzione del servizio IBM . Il campo non è presente se *Version* è minore di MQCXP_VERSION_3.

CapabilityFlags (MQLONG)
È possibile impostare l'indicatore della funzionalità su MQCF_NONE o MQCF_DIST_LISTS.

È possibile impostare uno dei seguenti indicatori di funzionalità:

MQCF_NONE
Nessun indicatore.

MQCF_DIST_LISTS
Elenchi di distribuzione supportati.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCXP_VERSION_3.

ExitNumber (MQLONG)
Questo campo specifica il numero ordinale dell'uscita.

Il numero ordinale dell'uscita, all'interno del tipo definito in *ExitId*. Ad esempio, se l'uscita richiamata è la terza uscita messaggio definita, questo campo contiene il valore 3. Se il tipo di uscita è uno per cui non è possibile definire un elenco di uscite (ad esempio, un'uscita di sicurezza), questo campo ha il valore 1.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCXP_VERSION_3.

I seguenti campi in questa struttura non sono presenti se *Version* è inferiore a MQCXP_VERSION_5.

ExitSpace (MQLONG)
Questo campo specifica il numero di byte nel buffer di trasmissione riservato per l'uscita da utilizzare.

Questo campo è rilevante solo per un'uscita di invio. Specifica la quantità di spazio in byte che le funzioni del canale IBM MQ riservano nel buffer di trasmissione per l'uscita da utilizzare. Questo campo consente all'uscita di aggiungere al buffer di trasmissione una piccola quantità di dati (in genere non superiore a poche centinaia di byte) per l'utilizzo da parte di un'altra uscita di ricezione complementare. I dati aggiunti dall'uscita di invio devono essere rimossi dall'uscita di ricezione.

Il valore è sempre zero su z/OS.

Nota: Questa funzione non deve essere utilizzata per inviare grandi quantità di dati, poiché potrebbe degradare le prestazioni o addirittura inibire il funzionamento del canale.

Impostando *ExitSpace* l'exit è garantito che ci sia sempre almeno quel numero di byte disponibili nel buffer di trasmissione per l'exit da utilizzare. Tuttavia, l'uscita può utilizzare meno della quantità riservata o più della quantità riservata se c'è spazio disponibile nel buffer di trasmissione. Lo spazio di uscita nel buffer viene fornito seguendo i dati esistenti.

ExitSpace può essere impostato dall'uscita solo quando *ExitReason* ha valore MQXR_INIT; in tutti gli altri casi il valore restituito dall'uscita viene ignorato. All'input dell'uscita, *ExitSpace* è zero per la chiamata MQXR_INIT ed è il valore restituito dalla chiamata MQXR_INIT in altri casi.

Se il valore restituito dalla chiamata MQXR_INIT è negativo o sono disponibili meno di 1024 byte nel buffer di trasmissione per i dati del messaggio dopo aver riservato lo spazio di uscita richiesto per tutte le uscite di invio nella catena, l'MCA emette un messaggio di errore e chiude il canale. Allo stesso modo, se durante il trasferimento dei dati le uscite nella catena di uscita di invio assegnano più spazio utente di quello riservato in modo che meno di 1024 byte rimangano nel buffer di trasmissione per i dati del messaggio, l'MCA emette un messaggio di errore e chiude il canale. Il limite di 1024 consente ai flussi di controllo e amministrativi del canale di essere elaborati dalla catena di uscite di invio, senza la necessità di segmentare i flussi.

Si tratta di un campo di input / output per l'uscita se *ExitReason* è MQXR_INIT e di un campo di input in tutti gli altri casi. Il campo non è presente se *Version* è minore di MQCXP_VERSION_5.

ID SSLCertUser(MQCHAR12)
Questo campo specifica l' UserId associato al certificato remoto.

È vuoto su tutte le piattaforme tranne z/OS

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCXP_VERSION_6.

SSLRemCertIssNameLunghezza (MQLONG)
Questo campo specifica la lunghezza in byte del DN (Distinguished Name) completo dell'emittente del certificato remoto a cui punta SSLCertRemoteIssuerNamePtr.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCXP_VERSION_6. Il valore è zero se non è un canale TLS.

SSLRemCertIssNamePtr (PMQVOID)
Questo campo specifica l'indirizzo del DN (Distinguished Name) completo dell'emittente del certificato remoto.

Il suo valore è il puntatore null se non è un canale TLS.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCXP_VERSION_6.

Nota: Il comportamento delle uscite di sicurezza del canale nella determinazione del DN (distinguished name) dell'oggetto e del DN (distinguished name) dell'emittente viene modificato da IBM WebSphere MQ 7.1. Per ulteriori informazioni, consultare Programmi di uscita di sicurezza del canale.

SecurityParms (PMQCSP)
Questo campo specifica l'indirizzo della struttura MQCSP utilizzata per specificare le credenziali di autenticazione.

Il valore iniziale di questo campo è il puntatore null.

Questo è un campo di immissione / emissione per l'uscita. Il campo non è presente se *Version* è minore di MQCXP_VERSION_6.

Il valore restituito dall'uscita in questo campo deve essere utilizzabile da IBM MQ fino a MQXR_TERM.

Compressione CurHdr(MQLONG)
Questo campo specifica quale tecnica viene attualmente utilizzata per comprimere i dati di intestazione.

È impostato su uno dei seguenti valori:

MQCOMPRESS_NONE
Nessuna compressione dati di intestazione eseguita.

SISTEMA MQCOMPRESS
Compressione dati di intestazione eseguita correttamente.

Il valore può essere modificato da un'uscita del messaggio del canale di invio a uno dei valori supportati negoziati a cui si accede dal campo Elenco HdrCompdi MQCD. Ciò abilita la tecnica utilizzata per comprimere i dati di intestazione da scegliere per ogni messaggio in base al contenuto del messaggio. Il valore modificato viene utilizzato solo per il messaggio corrente. Il canale termina se l'attributo viene modificato in un valore non supportato. Il valore viene ignorato se viene modificato al di fuori dell'uscita del messaggio del canale di invio.

Questo è un campo di immissione / emissione per l'uscita. Il campo non è presente se *Version* è minore di MQCXP_VERSION_6.

Compressione CurMsg(MQLONG)
Questo campo specifica la tecnica attualmente utilizzata per comprimere i dati del messaggio.

È impostato su uno dei seguenti valori:

MQCOMPRESS_NONE
Nessuna compressione dati di intestazione eseguita.

RLE MQCOMPRESS
La compressione dei dati dei messaggi è stata eseguita mediante la codifica run-length.

MQCOMPRESS_ZLIBFAST
La compressione dei dati dei messaggi è stata eseguita mediante la tecnica di compressione zlib. È preferibile che il tempo di compressione sia breve.

MQCOMPRESS_ZLIBHIGH
La compressione dei dati dei messaggi è stata eseguita mediante la tecnica di compressione zlib. È preferibile che il tempo di compressione sia elevato.

Il valore può essere modificato da un'uscita messaggio del canale di invio in uno dei valori supportati negoziati a cui si accede dal campo Elenco MsgCompdi MQCD. Ciò abilita la tecnica utilizzata per comprimere i dati del messaggio da stabilire per ciascun messaggio in base al contenuto del messaggio. Il valore modificato viene utilizzato solo per il messaggio corrente. Il canale termina se l'attributo viene modificato in un valore non supportato. Il valore viene ignorato se viene modificato al di fuori dell'uscita del messaggio del canale di invio.

Questo è un campo di immissione / emissione per l'uscita. Il campo non è presente se *Version* è minore di MQCXP_VERSION_6.

Hconn (MQHCONN)
Questo campo specifica l'handle di connessione che l'uscita utilizza se deve effettuare chiamate MQI all'interno dell'uscita.

Questo campo non è rilevante per le uscite in esecuzione sui canali di connessione client, dove contiene il valore MQHC_UNUSABLE_HCONN (-1).

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCXP_VERSION_7.

SharingConversations (MQBOOL)
Questo campo specifica se la conversazione è l'unica attualmente in esecuzione su questa istanza del canale o se più di una conversazione può essere attualmente in esecuzione su questa istanza del canale.

Indica inoltre se il programma di uscita è soggetto al rischio che l'MQCD venga modificato da un altro programma di uscita in esecuzione contemporaneamente.

Questo campo è rilevante solo per i programmi di uscita in esecuzione sui canali di connessione client o server.

È impostato su uno dei seguenti valori:

FALSO
L'istanza di uscita è l'unica istanza di uscita attualmente in esecuzione su questa istanza del canale. Ciò consente all'uscita di aggiornare in modo sicuro i campi MQCD senza conflitti da altre uscite in esecuzione su altre istanze del canale. Se le modifiche ai campi MQCD vengono eseguite dal canale è definito dalla tabella dei campi MQCD in “Modifica dei campi MQCD in un'uscita canale” a pagina 1559.

VERO
L'istanza di uscita non è l'unica istanza di uscita attualmente in esecuzione su questa istanza del canale. Le modifiche apportate a MQCD non vengono applicate dal canale, ad eccezione delle modifiche elencate nella tabella dei campi MQCD in “Modifica dei campi MQCD in un'uscita canale” a pagina 1559 per motivi di uscita diversi da MQXR_INIT. Se questa uscita aggiorna i campi MQCD, assicurarsi che non vi sia alcun conflitto tra altre uscite in esecuzione su altre conversazioni contemporaneamente, fornendo la serializzazione tra le uscite eseguite su questa istanza del canale.

Questo è un campo di immissione per l'uscita. Il campo non è presente se *Version* è minore di MQCXP_VERSION_7.

MCAUserSource (MQLONG)
Questo campo specifica l'origine dell'ID utente MCA fornito.

Può contenere uno dei seguenti valori:

MAP MQUSRC
L'ID utente viene specificato nell'attributo MCAUSER.

MQUSRC_CHALLEGATO
L'ID utente viene trasmesso dal partner in entrata o specificato nel campo MCAUSER definito nell'oggetto canale.

Questo è un campo di immissione per l'uscita. Il campo non è presente se la versione è inferiore a MQCXP_VERSION_8.

Punti pEntry(PMQIEP)
Questo campo specifica l'indirizzo del punto di ingresso dell'interfaccia per la chiamata MQI o DCI.

Il campo non è presente se la *Versione* è inferiore a MQCXP_VERSION_8.

RemoteProduct (MQCHAR4)
Questo campo specifica il nome del prodotto remoto.

Questo campo identifica il prodotto remoto del client, ad esempio, C o Java, come visualizzato nel campo **RPRODUCT** di DISPLAY CHSATUS.

Il campo non è presente se la *Versione* è inferiore a MQCXP_VERSION_9.

RemoteVersion (MQCHAR8)
Questo campo specifica il nome della versione remota.

Questo campo identifica la versione delle librerie client, come visualizzato nel campo **RVERSION** di DISPLAY CHSTATUS.

Il campo non è presente se la *Versione* è inferiore a MQCXP_VERSION_9.

Dichiarazione C
Questa dichiarazione è la dichiarazione C per la struttura MQCXP.

```
typedef struct tagMQCXP MQCXP;
struct tagMQCXP {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    ExitId;            /* Type of exit */
  MQLONG    ExitReason;        /* Reason for invoking exit */
  MQLONG    ExitResponse;      /* Response from exit */
  MQLONG    ExitResponse2;     /* Secondary response from exit */
  MQLONG    Feedback;          /* Feedback code */
  MQLONG    MaxSegmentLength;  /* Maximum segment length */
  MQBYTE16  ExitUserArea;      /* Exit user area */
  MQCHAR32  ExitData;          /* Exit data */
  MQLONG    MsgRetryCount;     /* Number of times the message has been
                                  retried */
  MQLONG    MsgRetryInterval;  /* Minimum interval in milliseconds after
                                  which the put operation should be
                                  retried */
  MQLONG    MsgRetryReason;    /* Reason code from previous attempt to
                                  put the message */
  MQLONG    HeaderLength;      /* Length of header information */
  MQCHAR48  PartnerName;       /* Partner Name */
  MQLONG    FAPLevel;          /* Negotiated Formats and Protocols
                                  level */
  MQLONG    CapabilityFlags;   /* Capability flags */
  MQLONG    ExitNumber;        /* Exit number */
  /* Ver:3 */
  /* Ver:4 */
  MQLONG    ExitSpace;         /* Number of bytes in transmission buffer
                                  reserved for exit to use */
  /* Ver:5 */
  MQCHAR12  SSLCertUserid;     /* User identifier associated
                                  with remote TLS certificate */
  MQLONG    SSLRemCertIssNameLength;   /* Length of
```

```
                                  distinguished name of issuer
                                  of remote TLS certificate */
  MQPTR     SSLRemCertIssNamePtr;      /* Address of
                                  distinguished name of issuer
                                  of remote TLS certificate */
  PMQVOID   SecurityParms;     /* Security parameters */
  MQLONG    CurHdrCompression; /* Header data compression
                                  used for current message */
  MQLONG    CurMsgCompression; /* Message data compression
                                  used for current message */
  /* Ver:6 */
  MQHCONN   Hconn;             /* Connection handle */
  MQBOOL    SharingConversations;      /* Multiple conversations
                                  possible on channel inst?  */
  /* Ver:7 */
  MQLONG    MCAUserSource;     /* Source of the provided MCA user ID */
  PMQIEP    pEntryPoints;      /* Address of the MQIEP structure */
  /* Ver:8 */
  MQCHAR4   RemoteProduct;     /* The identifier for the remote product */
  MQCHAR8   RemoteVersion;     /* The version of the remote product */
  /* Ver:9 */
};
```

Dichiarazione COBOL

Questa dichiarazione è la dichiarazione COBOL per la struttura MQCXP.

```
**   MQCXP structure
  10 MQCXP.
**    Structure identifier
   15 MQCXP-STRUCID          PIC X(4).
**    Structure version number
   15 MQCXP-VERSION          PIC S9(9) BINARY.
**    Type of exit
   15 MQCXP-EXITID           PIC S9(9) BINARY.
**    Reason for invoking exit
   15 MQCXP-EXITREASON       PIC S9(9) BINARY.
**    Response from exit
   15 MQCXP-EXITRESPONSE     PIC S9(9) BINARY.
**    Secondary response from exit
   15 MQCXP-EXITRESPONSE2    PIC S9(9) BINARY.
**    Feedback code
   15 MQCXP-FEEDBACK         PIC S9(9) BINARY.
**    Maximum segment length
   15 MQCXP-MAXSEGMENTLENGTH PIC S9(9) BINARY.
**    Exit user area
   15 MQCXP-EXITUSERAREA     PIC X(16).
**    Exit data
   15 MQCXP-EXITDATA         PIC X(32).
**    Number of times the message has been retried
   15 MQCXP-MSGRETRYCOUNT    PIC S9(9) BINARY.
**    Minimum interval in milliseconds after which the put operation
**    should be retried
   15 MQCXP-MSGRETRYINTERVAL PIC S9(9) BINARY.
**    Reason code from previous attempt to put the message
   15 MQCXP-MSGRETRYREASON   PIC S9(9) BINARY.
**    Length of header information
   15 MQCXP-HEADERLENGTH     PIC S9(9) BINARY.
**    Partner Name
   15 MQCXP-PARTNERNAME      PIC X(48).
**    Negotiated Formats and Protocols level
   15 MQCXP-FAPLEVEL         PIC S9(9) BINARY.
**    Capability flags
   15 MQCXP-CAPABILITYFLAGS  PIC S9(9) BINARY.
**    Exit number
   15 MQCXP-EXITNUMBER       PIC S9(9) BINARY.
**    Number of bytes in transmission buffer reserved for exit to use
   15 MQCXP-EXITSPACE        PIC S9(9) BINARY.
**    User Id associated with remote certificate
   15  MQCXP-SSLCERTUSERID   PIC X(12).
** Length of distinguished name of issuer of remote TLS
** certificate
   15  MQCXP-SSLREMCERTISSNAMELENGTH PIC S9(9) BINARY.
** Address of distinguished name of issuer of remote TLS
** certificate
   15  MQCXP-SSLREMCERTISSNAMEPTR    POINTER.
** Security parameters
   15  MQCXP-SECURITYPARMS           PIC S9(18) BINARY.
** Header data compression used for current message
```

```
   15  MQCXP-CURHDRCOMPRESSION       PIC S9(9) BINARY.
** Message data compression used for current message
   15  MQCXP-CURMSGCOMPRESSION       PIC S9(9) BINARY.
** Connection handle
   15 MQCXP-HCONN                    PIC S9(9) BINARY.
** Multiple conversations possible on channel instance?
   15 MQCXP-SHARINGCONVERSATIONS     PIC S9(9) BINARY.
** Source of the provided MCA user ID
   15 MQCXP-MCAUSERSOURCE            PIC S9(9) BINARY.
** Identifier of the remote product
   15 MQCXP-RPRODUCT                 PIC X(4).
** Identifier of the remote version
   15 MQCXP-RVERSION                 PIC X(8).
```

Dichiarazione RPG (ILE)

Questa dichiarazione è la dichiarazione RPG per la struttura MQCXP.

```
     D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     D* MQCXP Structure
     D*
     D* Structure identifier
     D  CXSID                  1      4
     D* Structure version number
     D  CXVER                  5      8I 0
     D* Type of exit
     D  CXXID                  9     12I 0
     D* Reason for invoking exit
     D  CXREA                 13     16I 0
     D* Response from exit
     D  CXRES                 17     20I 0
     D* Secondary response from exit
     D  CXRE2                 21     24I 0
     D* Feedback code
     D  CXFB                  25     28I 0
     D* Maximum segment length
     D  CXMSL                 29     32I 0
     D* Exit user area
     D  CXUA                  33     48
     D* Exit data
     D  CXDAT                 49     80
     D* Number of times the message has been retried
     D  CXMRC                 81     84I 0
     D* Minimum interval in milliseconds after which the put operation
     D* should be retried
     D  CXMRI                 85     88I 0
     D* Reason code from previous attempt to put the message
     D  CXMRR                 89     92I 0
     D* Length of header information
     D  CXHDL                 93     96I 0
     D* Partner Name
     D  CXPNM                 97    144
     D* Negotiated Formats and Protocols level
     D  CXFAP                145    148I 0
     D* Capability flags
     D  CXCAP                149    152I 0
     D* Exit number
     D  CXEXN                153    156I 0
     D* Number of bytes in transmission buffer reserved for exit to use
     D  CXHDL                157    160I 0
     D* User identifier associated with remote TLS certificate
     D  CXSSLCU              161    172
     D* Length of distinguished name of issuer of remote TLS certificate
     D  CXSRCINL             173    176I 0
     D* Address of distinguished name of issuer of remote TLS certificate
     D  CXSRCINP             177    192*
     D* Security parameters
     D  CXSECP               193    208*
     D* Header data compression used for current message
     D  CXCHC                209    212I 0
     D* Message data compression used for current message
     D  CXCMC                213    216I 0
     D* Connection handle
     D  CXHCONN               217    220I   0
     D* Multiple conversations possible on channel instance?
     D  CXSHARECONV           221    224I   0
     D* Source of the provided MCA user ID
     D  MCAUSERSOURCE         225    228I   0
     D* Identifer of the remote product
```

```
     D  CXRPRO               229    232I   0
     D* Identifier of the remote version
     D  CXRVER               233    240I   0
```

Dichiarazione assembler System/390

Questa dichiarazione è la dichiarazione dell'assembler System/390 per la struttura MQCXP.

```
MQCXP                   DSECT
MQCXP_STRUCID           DS   CL4   Structure identifier
MQCXP_VERSION           DS   F     Structure version number
MQCXP_EXITID            DS   F     Type of exit
MQCXP_EXITREASON        DS   F     Reason for invoking exit
MQCXP_EXITRESPONSE      DS   F     Response from exit
MQCXP_EXITRESPONSE2     DS   F     Secondary response from exit
MQCXP_FEEDBACK          DS   F     Feedback code
MQCXP_MAXSEGMENTLENGTH  DS   F     Maximum segment length
MQCXP_EXITUSERAREA      DS   XL16  Exit user area
MQCXP_EXITDATA          DS   CL32  Exit data
MQCXP_MSGRETRYCOUNT     DS   F     Number of times the message has been
*                                  retried
MQCXP_MSGRETRYINTERVAL  DS   F     Minimum interval in milliseconds
*                                  after which the put operation should
*                                  be retried
MQCXP_MSGRETRYREASON    DS   F     Reason code from previous attempt to
*                                  put the message
MQCXP_HEADERLENGTH      DS   F     Length of header information
MQCXP_PARTNERNAME       DS   CL48  Partner Name
MQCXP_FAPLEVEL          DS   F     Negotiated Formats and Protocols
*                                  level
MQCXP_CAPABILITYFLAGS   DS   F     Capability flags
MQCXP_EXITNUMBER        DS   F     Exit number
MQCXP_EXITSPACE         DS   F     Number of bytes in transmission
*                                  buffer reserved for exit to use
MQCXP_SSLCERTUSERID     DS   CL12  User identifier associated with
*                                  remote TLS certificate
MQCXP_SSLREMCERTISSNAMELENGTH   DS   F     Length of distinguished name
*                                  of issuer of remote TLS certificate
MQCXP_SSLREMCERTISSNAMEPTR      DS   F     Address of distinguished name
*                                  of issuer of remote TLS certificate
MQCXP_SECURITYPARMS     DS   F    Address of security parameters
MQCXP_CURHDRCOMPRESSION         DS   F     Header data compression used for
*                                  current message
MQCXP_CURMSGCOMPRESSION         DS   F      Message data compression used for
*                                  current message
MQCXP_HCONN             DS   F Connection handle
MQCXP_SHARINGCONVERSATIONS      DS   F Multiple conversations possible on
*                                  channel inst?
MQCXP_MCAUSERSOURCE     DS   F    Source of the provided MCA user ID
MQCXP_RPRODUCT          DS   CL4  Identifer of the remote product
MQCXP_RVERSION          DS   CL8  Identifer of the remote version

MQCXP_LENGTH            EQU  *-MQCXP
                        ORG  MQCXP
MQCXP_AREA              DS   CL(MQCXP_LENGTH)
```

MQXWD - Descrittore di attesa uscita

La struttura MQXWD è un parametro di input / output nella chiamata MQXWAIT.

Questa struttura è supportata solo su z/OS.

Riferimenti correlati

“Campi” a pagina 1578
Questo argomento elenca tutti i campi nella struttura MQXWD e descrive ciascun campo.

“Dichiarazione C” a pagina 1578
Questa dichiarazione è la dichiarazione C per la struttura MQXWD.

“Dichiarazione assembler System/390” a pagina 1578
Questa dichiarazione è la dichiarazione del programma di assemblaggio System/390 per la struttura MQXWD.

Campi

Questo argomento elenca tutti i campi nella struttura MQXWD e descrive ciascun campo.

StrucId (MQCHAR4)
Questo campo specifica l'identificativo della struttura.

Il valore deve essere:

ID_STRUC_MQXWD_
Identificativo per la struttura del descrittore di attesa uscita.

Per il linguaggio di programmazione C, viene definita anche la costante MQXWD_STRUC_ID_ARRAY, che ha lo stesso valore di MQXWD_STRUC_ID, ma è un array di caratteri anziché una stringa.

Il valore iniziale di questo campo è MQXWD_STRUC_ID.

Version (MQLONG)
Questo campo specifica il numero di versione della struttura.

Il valore deve essere:

MQXWD_VERSION_1
Numero di versione per la struttura del descrittore di attesa uscita.

Il valore iniziale di questo campo è MQXWD_VERSION_1.

Reserved1 (MQLONG)
Questo campo è riservato. Il valore deve essere zero.

Questo è un campo di immissione.

Reserved2 (MQLONG)
Questo campo è riservato. Il valore deve essere zero.

Questo è un campo di immissione.

Reserved3 (MQLONG)
Questo campo è riservato. Il valore deve essere zero.

Questo è un campo di immissione.

BCE (MQLONG)
Questo campo specifica il blocco di controllo eventi da attendere.

Questo campo è il blocco di controllo eventi (ECB) su cui attendere. Deve essere impostato su zero prima che venga emessa la chiamata MQXWAIT; una volta completato correttamente, contiene il codice postale.

Questo campo è un campo di immissione / emissione.

Dichiarazione C

Questa dichiarazione è la dichiarazione C per la struttura MQXWD.

```
typedef struct tagMQXWD MQXWD;
struct tagMQXWD {
  MQCHAR4  StrucId;    /* Structure identifier */
  MQLONG   Version;    /* Structure version number */
  MQLONG   Reserved1;  /* Reserved */
  MQLONG   Reserved2;  /* Reserved */
  MQLONG   Reserved3;  /* Reserved */
  MQLONG   ECB;        /* Event control block to wait on */
};
```

Dichiarazione assembler System/390

Questa dichiarazione è la dichiarazione del programma di assemblaggio System/390 per la struttura MQXWD.

```
MQXWD            DSECT
MQXWD_STRUCID    DS   CL4  Structure identifier
MQXWD_VERSION    DS   F    Structure version number
MQXWD_RESERVED1  DS   F    Reserved
MQXWD_RESERVED2  DS   F    Reserved
MQXWD_RESERVED3  DS   F    Reserved
MQXWD_ECB        DS   F    Event control block to wait on
*
MQXWD_LENGTH     EQU  *-MQXWD
                 ORG  MQXWD
MQXWD_AREA       DS   CL(MQXWD_LENGTH)
```

Chiamate di uscita del carico di lavoro cluster e strutture dati

Questa sezione fornisce informazioni di riferimento per l'uscita del carico di lavoro del cluster e le strutture dati. Si tratta di informazioni sull'interfaccia di programmazione di uso generale.

È possibile scrivere uscite del carico di lavoro del cluster nei seguenti linguaggi di programmazione:

- C
- Assembler System/390 (IBM MQ for z/OS)

La chiamata è descritta in:

- "MQ_CLUSTER_WORKLOAD_EXIT - Descrizione chiamata" a pagina 1580

I tipi di dati della struttura utilizzati dall'uscita sono descritti in:

- "MQXCLWLN - Esplora record del carico di lavoro del cluster" a pagina 1581
- "MQWXP - Struttura dei parametri di uscita del carico di lavoro cluster" a pagina 1585
- "MQWDR - Struttura record di destinazione del carico di lavoro del cluster" a pagina 1593
- "MQWQR - Struttura record coda carico di lavoro cluster" a pagina 1598
- "MQWCR - Struttura di record del cluster di workload del cluster" a pagina 1603
- z/OS Funzionamento asincrono dei comandi CLUSTER su z/OS

In questa sezione, gli attributi del gestore code e gli attributi della coda sono visualizzati in modo completo. I nomi equivalenti utilizzati nei comandi MQSC sono riportati di seguito. Per i dettagli dei comandi MQSC, vedere Comandi MQSC.

Tabella 824. Attributi gestore code

Nome completo	**Nome utilizzato in MQSC**
ClusterWorkloadData	CLWLDATA
ClusterWorkloadExit	CLWLEXIT
ClusterWorkloadLength	CLWLLEN

Tabella 825. Attributi Coda

Nome completo	**Nome utilizzato in MQSC**
DefBind	DEFBIND
DefPersistence	DEFPSIST
DefPriority	DEFPRTY
InhibitPut	PUT
QDesc	DESCR

Attività correlate

Scrittura e compilazione delle uscite del carico di lavoro del cluster

MQ_CLUSTER_WORKLOAD_EXIT - Descrizione chiamata

L'uscita del carico di lavoro cluster viene richiamata dal gestore code per instradare un messaggio a un gestore code disponibile.

Nota: Il gestore code non fornisce alcun punto di ingresso denominato MQ_CLUSTER_WORKLOAD_EXIT . Invece, il nome dell'uscita del carico di lavoro del cluster viene definito dall'attributo gestore code ClusterWorkloadExit .

L'uscita MQ_CLUSTER_WORKLOAD_EXIT è supportata su tutte le piattaforme.

Sintassi

```
MQ_CLUSTER_WORKLOAD_EXIT (ExitParms)
```

Riferimenti correlati

MQXCLWLN - Esplora record del carico di lavoro del cluster
La chiamata MQXCLWLN è utilizzata per esplorare le catene di record MQWDR, MQWQRe MQWCR memorizzati nella cache del cluster.

MQWXP - Struttura dei parametri di uscita del carico di lavoro cluster
La seguente tabella riepiloga i campi nella struttura del parametro di uscita del carico di lavoro del cluster MQWXP .

MQWDR - Struttura record di destinazione del carico di lavoro del cluster
La seguente tabella riepiloga i campi in MQWDR - Struttura di record di destinazione del carico di lavoro del cluster.

MQWQR - Struttura record coda carico di lavoro cluster
La seguente tabella riepiloga i campi in MQWQR - Struttura record della coda del workload del cluster.

MQWCR - Struttura di record del cluster di workload del cluster
La seguente tabella riepiloga i campi nella struttura di record del carico di lavoro del cluster MQWCR .

Parametri per MQ_CLUSTER_WORKLOAD_EXIT

Descrizione dei parametri nella chiamata MQ_CLUSTER_WORKLOAD_EXIT .

***ExitParms* (MQWXP) - input/output**
Blocco parametro di uscita.

- L'uscita imposta le informazioni in MQWXP per indicare come gestire il workload.

Riferimenti correlati

Note d'utilizzo
La funzione eseguita dall'uscita del carico di lavoro del cluster è definita dal fornitore dell'uscita. L'uscita, tuttavia, deve essere conforme alle regole definite nel blocco di controllo associato MQWXP.

Richiami della lingua per MQ_CLUSTER_WORKLOAD_EXIT
MQ_CLUSTER_WORKLOAD_EXIT supporta due linguaggi, C e High Level Assembler.

Note d'utilizzo

La funzione eseguita dall'uscita del carico di lavoro del cluster è definita dal fornitore dell'uscita. L'uscita, tuttavia, deve essere conforme alle regole definite nel blocco di controllo associato MQWXP.

Il gestore code non fornisce alcun punto di ingresso denominato MQ_CLUSTER_WORKLOAD_EXIT . Tuttavia, un typedef viene fornito per il nome MQ_CLUSTER_WORKLOAD_EXIT nel linguaggio di programmazione C. Utilizzare typedef per dichiarare l'uscita scritta dall'utente, per verificare che i parametri siano corretti.

Riferimenti correlati

Parametri per MQ_CLUSTER_WORKLOAD_EXIT
Descrizione dei parametri nella chiamata MQ_CLUSTER_WORKLOAD_EXIT .

Richiami della lingua per MQ_CLUSTER_WORKLOAD_EXIT
MQ_CLUSTER_WORKLOAD_EXIT supporta due linguaggi, C e High Level Assembler.

Richiami della lingua per MQ_CLUSTER_WORKLOAD_EXIT

MQ_CLUSTER_WORKLOAD_EXIT supporta due linguaggi, C e High Level Assembler.

Richiamo C

```
MQ_CLUSTER_WORKLOAD_EXIT (&ExitParms);
```

Sostituire *MQ_CLUSTER_WORKLOAD_EXIT* con il nome della funzione di uscita del carico di lavoro del cluster.

Dichiarare i parametri **MQ_CLUSTER_WORKLOAD_EXIT** come segue:

```
MQWXP ExitParms; /* Exit parameter block */
```

Chiamata High Level Assembler

```
CALL EXITNAME,(EXITPARMS)
```

Dichiarare i parametri come segue:

```
EXITPARMS      CMQWXPA       Exit parameter block
```

Riferimenti correlati

Parametri per MQ_CLUSTER_WORKLOAD_EXIT
Descrizione dei parametri nella chiamata MQ_CLUSTER_WORKLOAD_EXIT .

Note d'utilizzo
La funzione eseguita dall'uscita del carico di lavoro del cluster è definita dal fornitore dell'uscita. L'uscita, tuttavia, deve essere conforme alle regole definite nel blocco di controllo associato MQWXP.

MQXCLWLN - Esplora record del carico di lavoro del cluster

La chiamata MQXCLWLN è utilizzata per esplorare le catene di record MQWDR, MQWQRe MQWCR memorizzati nella cache del cluster.

La cache del cluster è un'area della memoria principale utilizzata per memorizzare le informazioni relative al cluster.

Se la cache del cluster è statica, ha una dimensione fissa. Se la si imposta su dinamica, la cache del cluster può espandersi come richiesto.

Impostare il tipo di cache del cluster su STATIC o DYNAMIC utilizzando un parametro di sistema o una macro.

- Multi Utilizzare il parametro di sistema ClusterCacheTipo su Multipiattaforme.
- z/OS Utilizzare il parametro CLCACHE nella macro CSQ6SYSP su z/OS.

Sintassi

```
MQXCLWLN (ExitParms, CurrentRecord, NextOffset, NextRecord, Compcode, Reason)
```

Riferimenti correlati

MQ_CLUSTER_WORKLOAD_EXIT - Descrizione chiamata
L'uscita del carico di lavoro cluster viene richiamata dal gestore code per instradare un messaggio a un gestore code disponibile.

MQWXP - Struttura dei parametri di uscita del carico di lavoro cluster
La seguente tabella riepiloga i campi nella struttura del parametro di uscita del carico di lavoro del cluster MQWXP .

MQWDR - Struttura record di destinazione del carico di lavoro del cluster
La seguente tabella riepiloga i campi in MQWDR - Struttura di record di destinazione del carico di lavoro del cluster.

MQWQR - Struttura record coda carico di lavoro cluster
La seguente tabella riepiloga i campi in MQWQR - Struttura record della coda del workload del cluster.

MQWCR - Struttura di record del cluster di workload del cluster
La seguente tabella riepiloga i campi nella struttura di record del carico di lavoro del cluster MQWCR .

Parametri per MQXCLWLN - Esplora record del carico di lavoro del cluster

Descrizione dei parametri nella chiamata MQXCLWLN .

ExitParms (MQWXP) - input/output
Blocco parametro di uscita.

Questa struttura contiene informazioni relative al richiamo dell'exit. L'uscita imposta le informazioni in questa struttura per indicare come gestire il workload.

CurrentRecord (MQPTR) - immissione
Indirizzo del record corrente.

Questa struttura contiene informazioni relative all'indirizzo del record attualmente esaminato dall'exit. Il record deve essere uno dei seguenti tipi:

- Record di destinazione del carico di lavoro del cluster (MQWDR)
- Record coda carico di lavoro cluster (MQWQR)
- Record cluster di workload cluster (MQWCR)

NextOffset (MQLONG) - immissione
Offset del record successivo.

Questa struttura contiene informazioni relative allo scostamento del record o della struttura successivi. *NextOffset* è il valore del campo offset appropriato nel record corrente e deve essere uno dei campi seguenti:

- Campo ChannelDefOffset in MQWDR
- Campo ClusterRecOffset in MQWDR
- Campo ClusterRecClusterRec in MQWQR
- Campo ClusterRecClusterRec in MQWCR

NextRecord (MQPTR) - output
Indirizzo del record o della struttura successiva.

Questa struttura contiene informazioni relative all'indirizzo del record o della struttura successiva. Se *CurrentRecord* è l'indirizzo di un MQWDR, e *NextOffset* è il valore del campo ChannelDefOffset , *NextRecord* è l'indirizzo della struttura di definizione del canale (MQCD).

Se non ci sono record o strutture successivi, il gestore code imposta *NextRecord* sul puntatore null e la chiamata restituisce il codice di completamento MQCC_WARNING e il codice motivo MQRC_NO_RECORD_AVAILABLE.

CompCode (MQLONG) - output
Codice di completamento.

Il codice di completamento ha uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_WARNING
Avvertenza (completamento parziale).

MQCC_FAILED
Chiamata fallita.

Motivo (MQLONG) - output
Codice di errore CompCode

Se CompCode è MQCC_OK:

MQRC_NONE
(0, X'0000')
Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

MQRC_NO_RECORD_AVAILABLE
(2359, X'0937')
Nessun record disponibile. È stata immessa una chiamata MQXCLWLN da un'uscita del carico di lavoro del cluster per ottenere l'indirizzo del record successivo nella catena. Il record corrente è l'ultimo record nella catena. Azione correttiva: nessuna.

Se *CompCode* è MQCC_FAILED:

MQRC_CURRENT_RECORD_ERROR
(2357, X'0935')
Parametro **CurrentRecord** non valido. È stata immessa una chiamata MQXCLWLN da un'uscita del carico di lavoro del cluster per ottenere l'indirizzo del record successivo nella catena. L'indirizzo specificato dal parametro **CurrentRecord** non è l'indirizzo di un record valido.

CurrentRecord deve essere l'indirizzo di un record di destinazione, MQWDR, di un record coda (MQWQR) o di un record cluster (MQWCR) che si trovano nella cache cluster. Azione correttiva: assicurarsi che l'uscita del carico di lavoro del cluster passi l'indirizzo di un record valido che risiede nella cache del cluster.

MQRC_ENVIRONMENT_ERROR
(2012, X'07DC')
Chiamata non valida nell'ambiente. È stata emessa una chiamata MQXCLWLN , ma non da un'uscita del carico di lavoro del cluster.

MQRC_NEXT_OFFSET_ERROR
(2358, X'0936')
Parametro **NextOffset** non valido. È stata immessa una chiamata MQXCLWLN da un'uscita del carico di lavoro del cluster per ottenere l'indirizzo del record successivo nella catena. Lo scostamento specificato dal parametro **NextOffset** non è valido. **NextOffset** deve essere il valore di uno dei seguenti campi:

- Campo ChannelDefOffset in MQWDR
- Campo ClusterRecOffset in MQWDR
- Campo ClusterRecClusterRec in MQWQR
- Campo ClusterRecClusterRec in MQWCR

Azione correttiva: assicurarsi che il valore specificato per il parametro **NextOffset** sia il valore di uno dei campi elencati precedentemente.

MQRC_NEXT_RECORD_ERROR
(2361, X'0939')
Parametro **NextRecord** non valido.

MQRC_WXP_ERROR
(2356, X'0934')
Struttura del parametro di uscita del carico di lavoro non valida. È stata immessa una chiamata MQXCLWLN da un'uscita del carico di lavoro del cluster per ottenere l'indirizzo del record successivo nella catena. La struttura del parametro di uscita del carico di lavoro **ExitParms** non è valida, per uno dei seguenti motivi:

- Il puntatore del parametro non è valido. Non è sempre possibile rilevare puntatori di parametri non validi; se non rilevati, si verificano risultati imprevedibili.
- Il campo StrucId non è MQWXP_STRUC_ID.
- Il campo Versione non è MQWXP_VERSION_2.
- Il campo Contesto non contiene il valore passato all'uscita dal gestore code.

Azione correttiva: assicurarsi che il parametro specificato per **ExitParms** sia la struttura MQWXP che è stata passata all'exit quando è stata richiamata l'exit.

Riferimenti correlati
Note sull'utilizzo per MQXCLWLN - Esplora record del carico di lavoro del cluster
Utilizzare MQXCLWLN per navigare tra i record del cluster, anche se la cache è statica.

Richiami del linguaggio di MQXCLWLN
MQXCLWLN supporta due lingue, C e High Level Assembler.

Note sull'utilizzo per MQXCLWLN - Esplora record del carico di lavoro del cluster
Utilizzare MQXCLWLN per navigare tra i record del cluster, anche se la cache è statica.

Se la cache del cluster è dinamica, la chiamata MQXCLWLN deve essere utilizzata per navigare tra i record. L'uscita termina in modo anomalo se si utilizza l'aritmetica semplice di puntatore e scostamento per navigare tra i record.

Se la cache del cluster è statica, non è necessario utilizzare MQXCLWLN per navigare tra i record. Generalmente, utilizzare MQXCLWLN anche quando la cache è statica. È quindi possibile modificare la cache del cluster in dinamica senza modificare l'uscita del carico di lavoro.

Riferimenti correlati
Parametri per MQXCLWLN - Esplora record del carico di lavoro del cluster
Descrizione dei parametri nella chiamata MQXCLWLN .

Richiami del linguaggio di MQXCLWLN
MQXCLWLN supporta due lingue, C e High Level Assembler.

Richiami del linguaggio di MQXCLWLN
MQXCLWLN supporta due lingue, C e High Level Assembler.

Richiamo C

```
MQXCLWLN (&ExitParms, CurrentRecord, NextOffset, &NextRecord, &CompCode, &Reason) ;
```

Dichiarare i parametri come segue:

```
Typedef struct tagMQXCLWLN {
MQWXP   ExitParms;      /* Exit parameter block */
MQPTR   CurrentRecord;  /* Address of current record*/
MQLONG  NextOffset;     /* Offset of next record */
MQPTR   NextRecord;     /* Address of next record or structure */
```

```
MQLONG  CompCode;       /* Completion code */
MQLONG  Reason;         /* Reason code qualifying CompCode */
```

Chiamata High Level Assembler

```
CALL MQXCLWLN,(CLWLEXITPARMS,CURRENTRECORD,NEXTOFFSET,NEXTRECORD,COMPCODE,REASON)
```

Dichiarare i parametri come segue:

```
CLWLEXITPARMS CMQWXPA, Cluster workload exit parameter block
CURRENTRECORD CMQWDRA, Current record
NEXTOFFSET    DS F     Next offset
NEXTRECORD    DS F     Next record
COMPCODE      DS F     Completion code
REASON        DS F     Reason code qualifying COMPCODE
```

Riferimenti correlati

Parametri per MQXCLWLN - Esplora record del carico di lavoro del cluster
Descrizione dei parametri nella chiamata MQXCLWLN .

Note sull'utilizzo per MQXCLWLN - Esplora record del carico di lavoro del cluster
Utilizzare MQXCLWLN per navigare tra i record del cluster, anche se la cache è statica.

MQWXP - Struttura dei parametri di uscita del carico di lavoro cluster

La seguente tabella riepiloga i campi nella struttura del parametro di uscita del carico di lavoro del cluster MQWXP .

Tabella 826. Campi in MQWXP

Campo	Descrizione	Pagina
StrucId	Identificativo struttura	StrucId
Version	Numero di versione della struttura	Versione
ExitId	Tipo di uscita	ExitId
ExitReason	Motivo del richiamo dell'uscita	ExitReason
ExitResponse	Risposta dall'uscita	ExitResponse
ExitResponse2	Risposta secondaria dall'uscita	ExitResponse2
Feedback	Codice feedback	Feedback
Flags	Indica i valori. Questi indicatori di bit vengono utilizzati per indicare le informazioni sul messaggio da inserire	Flag
ExitUserArea	Esci dall'area utente	AreaExitUser
ExitData	Dati uscita	ExitData
MsgDescPtr	Indirizzo del descrittore del messaggio (MQMD)	MsgDescPtr
MsgBufferPtr	Indirizzo del buffer contenente alcuni o tutti i dati del messaggio	Ptr MsgBuffer
MsgBufferLength	Lunghezza del buffer contenente i dati del messaggio	MsgBufferMsgBuffer
MsgLength	Lunghezza del messaggio completo	MsgLength
QName	Nome della coda	QName

Tabella 826. Campi in MQWXP (Continua)		
Campo	**Descrizione**	**Pagina**
QMgrName	Nome del gestore code locale	QMgrName
DestinationCount	Numero di destinazioni possibili	DestinationCount
DestinationChosen	Destinazione scelta	DestinationChosen
DestinationArrayPtr	Indirizzo di un array di puntatori ai record di destinazione (MQWDR)	DestinationArrayptr
QArrayPtr	Indirizzo di un array di puntatori ai record della coda (MQWQR)	QArrayPtr
Nota: I restanti campi vengono ignorati se la versione è inferiore a MQWXP_VERSION_2.		
CacheContext	Informazioni contesto	CacheContext
CacheType	Tipo di cache cluster	CacheType
Nota: I restanti campi vengono ignorati se la versione è inferiore a MQWXP_VERSION_3.		
CLWLMRUChannels	Numero massimo di canali cluster in uscita attivi consentiti	CLWLMRUChannels
Nota: I restanti campi vengono ignorati se la versione è inferiore a MQWXP_VERSION_4.		
pEntryPoints	L'indirizzo della struttura MQIEP per consentire l'effettuazione di chiamate MQI e DCI	pEntryPunti

La struttura del parametro di uscita del carico di lavoro del cluster descrive le informazioni trasmesse all'uscita del carico di lavoro del cluster.

La struttura del parametro di uscita del carico di lavoro del cluster è supportato su tutte le piattaforme

Inoltre, le strutture MQWXP1, MQWXP2 e MQWXP3 sono disponibili per la compatibilità con le versioni precedenti.

Riferimenti correlati

MQ_CLUSTER_WORKLOAD_EXIT - Descrizione chiamata
L'uscita del carico di lavoro cluster viene richiamata dal gestore code per instradare un messaggio a un gestore code disponibile.

MQXCLWLN - Esplora record del carico di lavoro del cluster
La chiamata MQXCLWLN è utilizzata per esplorare le catene di record MQWDR, MQWQRe MQWCR memorizzati nella cache del cluster.

MQWDR - Struttura record di destinazione del carico di lavoro del cluster
La seguente tabella riepiloga i campi in MQWDR - Struttura di record di destinazione del carico di lavoro del cluster.

MQWQR - Struttura record coda carico di lavoro cluster
La seguente tabella riepiloga i campi in MQWQR - Struttura record della coda del workload del cluster.

MQWCR - Struttura di record del cluster di workload del cluster
La seguente tabella riepiloga i campi nella struttura di record del carico di lavoro del cluster MQWCR .

Campi in MQWXP - Struttura dei parametri di uscita del carico di lavoro del cluster

Descrizione dei campi in MQWXP - Struttura del parametro di uscita del carico di lavoro del cluster

StrucId (MQCHAR4) - input
L'identificativo della struttura per la struttura del parametro di uscita del carico di lavoro cluster.

- Il valore StrucId è MQWXP_STRUC_ID.

- Per il linguaggio di programmazione C, viene definita anche la costante MQWXP_STRUC_ID_ARRAY . Ha lo stesso valore di MQWXP_STRUC_ID. Si tratta di un array di caratteri invece di una stringa.

Versione (MQLONG) - input
Indica il numero di versione della struttura. Versione assume uno dei seguenti valori:

MQWXP_VERSION_1
Version-1 struttura del parametro di uscita del carico di lavoro del cluster.

MQWXP_VERSION_1 è supportato in tutti gli ambienti.

MQWXP_VERSION_2
Struttura del parametro di uscita del carico di lavoro del cluster Version-2 .

MQWXP_VERSION_2 è supportato nei seguenti ambienti:

- AIX
- IBM i
- Linux
- Windows

MQWXP_VERSION_3
Struttura del parametro di uscita del carico di lavoro del cluster Version-3 .

MQWXP_VERSION_3 è supportato nei seguenti ambienti:

- AIX
- IBM i
- Linux
- Windows

MQWXP_VERSION_4
Version-4 struttura del parametro di uscita del carico di lavoro del cluster.

MQWXP_VERSION_4 è supportato nei seguenti ambienti:

- AIX
- IBM i
- Linux
- Windows

MQWXP_CURRENT_VERSION
La versione corrente della struttura del parametro di uscita del carico di lavoro del cluster.

ExitId (MQLONG) - input
Indica il tipo di uscita richiamata. L'uscita del carico di lavoro del cluster è l'unica uscita supportata.

- Il valore ExitId deve essere MQXT_CLUSTER_WORKLOAD_EXIT

ExitReason (MQLONG) - input
Indica il motivo per richiamare l'uscita del carico di lavoro del cluster. ExitReason assume uno dei seguenti valori:

MQXR_INIT
Indica che l'uscita viene richiamata per la prima volta.

Acquisire e inizializzare tutte le risorse di cui l'uscita potrebbe aver bisogno, come la memoria principale.

MQXR_TERM
Indica che l'uscita sta per essere terminata.

Liberare tutte le risorse che l'uscita potrebbe aver acquisito da quando è stata inizializzata, come la memoria principale.

MQXR_CLWL_OPEN
Richiamato da MQOPEN.

MQXR_CLWL_PUT
Richiamato da MQPUT o MQPUT1.

MQXR_CLWL_MOVE
Richiamato da MCA quando lo stato del canale è cambiato.

MQXR_CLWL_REPOS
Richiamato da MQPUT o MQPUT1 per un messaggio PCF del gestore repository.

MQXR_CLWL_REPOS_MOVE
Richiamato da MCA per un messaggio PCF del gestore repository se lo stato del canale è stato modificato.

ExitResponse (MQLONG) - output
Impostare ExitResponse per indicare se l'elaborazione del messaggio continua. Deve essere uno dei seguenti valori:

MQXCC_OK
Continuare l'elaborazione del messaggio normalmente.

- DestinationChosen identifica la destinazione a cui deve essere inviato il messaggio.

MQXCC_SUPPRESS_FUNCTION
Interrompere l'elaborazione del messaggio.

- Le azioni intraprese dal gestore code dipendono dal motivo per cui è stata richiamata l'exit:

Tabella 827. Azioni intraprese dal gestore code

ExitReason	**Azione eseguita**
– MQXR_CLWL_OPEN – MQXR_CLWL_REPOS – MQXR_CLWL_PUT	La chiamata MQOPEN, MQPUTo MQPUT1 non riesce con codice di completamento MQCC_FAILED e codice motivo MQRC_STOPPED_BY_CLUSTER_EXIT.
– MQXR_CLWL_MOVE – MQXR_CLWL_REPOS_MOVE	Il messaggio viene inserito nella coda di messaggi non recapitabili.

MQXCC_SUPPRESS_EXIT
Continuare normalmente l'elaborazione del messaggio corrente. Non richiamare nuovamente l'uscita fino a quando il gestore code non viene arrestato.

Il gestore code elabora i messaggi successivi come se l'attributo del gestore code ClusterWorkloadExit fosse vuoto. DestinationChosen identifica la destinazione a cui viene inviato il messaggio corrente.

Qualsiasi altro valore
Elaborare il messaggio come se MQXCC_SUPPRESS_FUNCTION fosse specificato.

ExitResponse2 (MQLONG) - input/output
Impostare ExitResponse2 per fornire al gestore code ulteriori informazioni.

- MQXR2_STATIC_CACHE è il valore predefinito ed è impostato all'entrata dell'uscita.
- Quando ExitReason ha il valore MQXR_INIT, l'uscita può impostare uno dei seguenti valori in ExitResponse2:

MQXR2_STATIC_CACHE
L'uscita richiede una cache cluster statico.

– Se la cache del cluster è statica, l'uscita non deve utilizzare la chiamata MQXCLWLN per navigare nelle catene di record nella cache del cluster.
– Se la cache del cluster è dinamica, l'uscita non può navigare correttamente tra i record nella cache.

Nota: Il gestore code elabora la restituzione dalla chiamata MQXR_INIT come se l'uscita avesse restituito MQXCC_SUPPRESS_EXIT nel campo ExitResponse .

MQXR2_DYNAMIC_CACHE
L'uscita può funzionare con una cache statica o dinamica.

– Se l'uscita restituisce questo valore, l'uscita deve utilizzare la chiamata MQXCLWLN per navigare nelle catene di record nella cache del cluster.

Feedback (MQLONG) - input
Un campo riservato. Il valore è zero.

Indicatori (MQLONG) - input
Indica le informazioni sul messaggio da inserire.

- Il valore di Indicatori è MQWXP_PUT_BY_CLUSTER_CHL. Il messaggio ha origine da un canale cluster, piuttosto che localmente o da un canale non cluster. In altre parole, il messaggio proviene da un altro gestore code del cluster.

Riservato (MQLONG) - input
Un campo riservato. Il valore è zero.

ExitUserExitUser (MQBYTE16) - input/output
Impostare l'area ExitUser per comunicare tra le chiamate all'uscita.

- ExitUserArea viene inizializzata su zero binario prima del primo richiamo dell'uscita. Tutte le modifiche apportate a questo campo dall'uscita vengono conservate nei richiami dell'uscita che si verificano tra la chiamata MQCONN e la chiamata MQDISC corrispondente. Il campo viene reimpostato su zero binario quando si verifica la chiamata MQDISC .
- Il primo richiamo dell'exit è indicato dal campo ExitReason con valore MQXR_INIT.
- Sono definite le seguenti costanti:

  MQXUA_NONE - stringa
  MQXUA_NONE_ARRAY - array di caratteri
  Nessuna informazione utente. Entrambe le costanti sono zero binario per la lunghezza del campo.

  MQ_EXIT_USER_AREA_LENGTH
  La lunghezza dell'area ExitUser.

ExitData (MQCHAR32) - input
Il valore dell'attributo gestore code ClusterWorkloadData . Se non è stato definito alcun valore per tale attributo, questo campo è vuoto.

- La lunghezza di ExitData è fornita da MQ_EXIT_DATA_LENGTH.

MsgDescPtr (PMQMD) - input
L'indirizzo di una copia del descrittore del messaggio (MQMD) per il messaggio elaborato.

- Tutte le modifiche apportate al descrittore del messaggio dall'uscita vengono ignorate dal gestore code.
- Se ExitReason ha uno dei seguenti valori MsgDescPtr è impostato sul puntatore null e non viene passato alcun descrittore di messaggi all'uscita:
  – MQXR_INIT
  – MQXR_TERM
  – MQXR_CLWL_OPEN

MsgBufferMsgBuffer (PMQVOID) - input
L'indirizzo di un buffer contenente una copia dei primi MsgBufferLength byte dei dati del messaggio.

- Tutte le modifiche apportate ai dati del messaggio dall'uscita vengono ignorate dal gestore code.
- Nessun dato di messaggio viene passato all'uscita quando:
  - MsgDescPtr è il puntatore null.
  - Il messaggio non contiene dati.
  - L'attributo del gestore code ClusterWorkloadLength è zero.

  In questi casi, MsgBufferPtr è il puntatore null.

MsgBufferMsgBuffer (MQLONG) - input
La lunghezza del buffer contenente i dati del messaggio passati all'exit.

- La lunghezza è controllata dall'attributo gestore code ClusterWorkloadLength .
- La lunghezza potrebbe essere inferiore alla lunghezza del messaggio completo, vedere MsgLength.

MsgLength (MQLONG) - input
La lunghezza del messaggio completo passato all'uscita.

- MsgBufferLa lunghezza potrebbe essere inferiore alla lunghezza del messaggio completo.
- MsgLength è zero se ExitReason è MQXR_INIT, MQXR_TERMo MQXR_CLWL_OPEN.

QName (MQCHAR48) - input
Il nome della coda di destinazione. La coda è una coda cluster.

- La lunghezza di QName è MQ_Q_NAME_LENGTH.

QMgrName (MQCHAR48) - input
Il nome del gestore code locale che ha richiamato l'uscita del carico di lavoro cluster.

- La lunghezza di QMgrName è MQ_Q_MGR_NAME_LENGTH.

DestinationCount (MQLONG) - input
Il numero di destinazioni possibili. Le destinazioni sono istanze della coda di destinazione e sono descritte dai record di destinazione.

- Un record di destinazione è una struttura MQWDR . Esiste una struttura per ogni possibile instradamento a ciascuna istanza della coda.
- Le strutture MQWDR sono indirizzate da un array di puntatori; consultare DestinationArrayPtr.

DestinationChosen (MQLONG) - input/output
La destinazione scelta.

- Il numero della struttura MQWDR che identifica l'instradamento e l'istanza della coda a cui deve essere inviato il messaggio.
- Il valore è compreso nell'intervallo 1 - DestinationCount.
- All'input per l'uscita, DestinationChosen indica l'instradamento e l'istanza della coda selezionati dal gestore code. L'uscita può accettare questa scelta o scegliere un instradamento e un'istanza della coda differenti.
- Il valore impostato dall'uscita deve essere compreso nell'intervallo 1 - DestinationCount. Se viene restituito un qualsiasi altro valore, il gestore code utilizza il valore di DestinationChosen nell'input per l'uscita.

DestinationArrayPtr (PPMQWDR) - input
L'indirizzo di un array di puntatori ai record destinazione (MQWDR).

- Sono presenti DestinationCount record di destinazione.

QArrayPtr (PPMQWQR) - input
L'indirizzo di un array di puntatori ai record della coda (MQWQR).

- Se i record della coda sono disponibili, ci sono `DestinationCount` di essi.
- Se non sono disponibili record della coda, `QArrayPtr` è il puntatore null.

  Nota: `QArrayPtr` può essere il puntatore null anche quando `DestinationCount` è maggiore di zero.

`CacheContext` (`MQPTR`): Versione 2 - input
Il campo `CacheContext` è riservato per l'utilizzo da parte del gestore code. L'uscita non deve modificare il valore di questo campo.

CacheType (`MQLONG`): Versione 2 - input
La cache del cluster ha uno dei seguenti tipi:

`MQCLCT_STATIC`
La cache è statica.

- La dimensione della cache è fissa e non può aumentare con il funzionamento del gestore code.
- Non è necessario utilizzare la chiamata `MQXCLWLN` per esplorare i record in questo tipo di cache.

`MQCLCT_DYNAMIC`
La cache è dinamica.

- La dimensione della cache può aumentare in modo da contenere le informazioni sul cluster variabili.
- È necessario utilizzare la chiamata `MQXCLWLN` per esplorare i record in questo tipo di cache.

`CLWLMRUChannels` (`MQLONG`): Versione 3 - input
Indica il numero massimo di canali cluster in uscita attivi, da considerare per l'utilizzo da parte dell'algoritmo di scelta del carico di lavoro cluster.

- `CLWLMRUChannels` è un valore compreso tra 1 e 999 999 999.

`pEntryPunti` (`PMQIEP`): Versione 4
L'indirizzo di una struttura MQIEP tramite cui è possibile effettuare chiamate MQI e DCI.

Riferimenti correlati
Valori iniziali e dichiarazioni di lingua per MQWXP
Valori iniziali e dichiarazioni di linguaggio C e High Level Assembler per MQWXP - Struttura del parametro di uscita del workload del cluster.

Valori iniziali e dichiarazioni di lingua per MQWXP
Valori iniziali e dichiarazioni di linguaggio C e High Level Assembler per MQWXP - Struttura del parametro di uscita del workload del cluster.

Tabella 828. Campi in MQWXP

Nome campo	Nome della costante	Valore della costante
StrucId	MQWXP_STRUC_ID	'WXP¬'
Version	MQWXP_VERSION_2	2
ExitId	Nessuna	0
ExitReason	MQXCC_OK	0
ExitResponse	Nessuna	0
ExitResponse2	Nessuna	0
Flags	Nessuna	0
ExitUserArea	{MQXUA_NONE_ARRAY}	0
ExitData	Nessuna	""
MsgDescPtr	Nessuna	NULL

Tabella 828. Campi in MQWXP (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
MsgBufferPtr	Nessuna	NULL
MsgBufferLength	Nessuna	0
MsgBufferPtr	Nessuna	0
QName	Nessuna	""
QMgrName	Nessuna	""
DestinationCount	Nessuna	0
DestinationChosen	Nessuna	0
DestinationArrayPtr	Nessuna	NULL
QArrayPtr	Nessuna	NULL
CacheContext	Nessuna	NULL
CacheType	MQCLCT_DYNAMIC	1
CLWLMRUChannels	Nessuna	0
pEntryPoints	Nessuna	NULL

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macro MQWXP_DEFAULT contiene i valori predefiniti. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

```
MQWDR MyWXP = {MQWXP_DEFAULT};
```

Dichiarazione C

```
typedef struct tagMQWXP {
  MQCHAR4   StrucId;              /* Structure identifier */
  MQLONG    Version;              /* Structure version number */
  MQLONG    ExitId;               /* Type of exit */
  MQLONG    ExitReason;           /* Reason for invoking exit */
  MQLONG    ExitResponse;         /* Response from exit */
  MQLONG    ExitResponse2;        /* Reserved */
  MQLONG    Feedback;             /* Reserved */
  MQLONG    Flags;                /* Flags */
  MQBYTE16  ExitUserArea;         /* Exit user area */
  MQCHAR32  ExitData;             /* Exit data */
  PMQMD     MsgDescPtr;           /* Address of message descriptor */
  PMQVOID   MsgBufferPtr;         /* Address of buffer containing some
                                     or all of the message data */
  MQLONG    MsgBufferLength;      /* Length of buffer containing message
                                     data */
  MQLONG    MsgLength;            /* Length of complete message */
  MQCHAR48  QName;                /* Queue name */
  MQCHAR48  QMgrName;             /* Name of local queue manager */
  MQLONG    DestinationCount;     /* Number of possible destinations */
  MQLONG    DestinationChosen;    /* Destination chosen */
  PPMQWDR   DestinationArrayPtr;  /* Address of an array of pointers to
                                     destination records */
  PPMQWQR   QArrayPtr;            /* Address of an array of pointers to
                                     queue records */
  /* version 1 */
  MQPTR     CacheContext;         /* Context information */
  MQLONG    CacheType;            /* Type of cluster cache */
```

```
  /* version 2 */
  MQLONG    CLWLMRUChannnels;     /* Maximum number of most recently
                                     used cluster channels */
  /* version 3 */
  PMQIEP    pEntryPoints;         /* Address of the MQIEP structure */
  /* version 4 */
};
```

High Level Assembler

```
MQWXP                       DSECT
MQWXP_STRUCID               DS   CL4      Structure identifier
MQWXP_VERSION               DS   F        Structure version number
MQWXP_EXITID                DS   F        Type of exit
MQWXP_EXITREASON            DS   F        Reason for invoking exit
MQWXP_EXITRESPONSE          DS   F        Response from exit
MQWXP_EXITRESPONSE2         DS   F        Reserved
MQWXP_FEEDBACK              DS   F        Reserved
MQWXP_RESERVED              DS   F        Reserved
MQWXP_EXITUSERAREA          DS   XL16     Exit user area
MQWXP_EXITDATA              DS   CL32     Exit data
MQWXP_MSGDESCPTR            DS   F        Address of message
*                                         descriptor
MQWXP_MSGBUFFERPTR          DS   F        Address of buffer containing
*                                         some or all of the message
*                                         data
MQWXP_MSGBUFFERLENGTH       DS   F        Length of buffer containing
*                                         message data
MQWXP_MSGLENGTH             DS   F        Length of complete message
MQWXP_QNAME                 DS   CL48     Queue name
MQWXP_QMGRNAME              DS   CL48     Name of local queue manager
MQWXP_DESTINATIONCOUNT      DS   F        Number of possible
*                                         destinations
MQWXP_DESTINATIONCHOSEN     DS   F        Destination chosen
MQWXP_DESTINATIONARRAYPTR   DS   F        Address of an array of
*                                         pointers to destination
*                                         records
MQWXP_QARRAYPTR             DS   F        Address of an array of
*                                         pointers to queue records
MQWXP_CACHECONTEXT          DS   F        Context information
MQWXP_CACHETYPE             DS   F        Type of cluster cache
MQWXP_CLWLMRUCHANNELS       DS   F        Number of most recently used
*                                         channels for workload balancing

MQWXP_LENGTH                EQU  *-MQWXP  Length of structure
                            ORG  MQWXP
MQWXP_AREA                  DS   CL(MQWXP_LENGTH)
```

Riferimenti correlati

Campi in MQWXP - Struttura dei parametri di uscita del carico di lavoro del cluster
Descrizione dei campi in MQWXP - Struttura del parametro di uscita del carico di lavoro del cluster

MQWDR - Struttura record di destinazione del carico di lavoro del cluster

La seguente tabella riepiloga i campi in MQWDR - Struttura di record di destinazione del carico di lavoro del cluster.

Tabella 829. Campi in MQWDR

Campo	Descrizione	Pagina
StrucId	Identificativo struttura	StrucId
Version	Numero di versione della struttura	Versione
StrucLength	Lunghezza della struttura MQWDR	StrucLength
QMgrFlags	Indicatori del gestore code	QMgrFlags
QMgrIdentifier	Identificativo gestore code	QMgrIdentifier
QMgrName	Nome del gestore code	QMgrName

Tabella 829. Campi in MQWDR (Continua)		
Campo	**Descrizione**	**Pagina**
ClusterRecOffset	Offset logico del primo record del cluster (MQWCR)	ClusterRecClusterRec
ChannelState	Stato canale	ChannelState
ChannelDefOffset	Offset logico della struttura di definizione del canale (MQCD)	ChannelDefOffset
Nota: I restanti campi vengono ignorati se la versione è inferiore a MQWDR_VERSION_2.		
DestSeqNumber	Numero di sequenza destinazione canale	DestSeqDestSeq
DestSeqFactor	Fattore di sequenza destinazione canale per il peso	DestSeqDestSeq

La struttura del record di destinazione del carico di lavoro del cluster contiene informazioni relative a una delle destinazioni possibili per il messaggio. Esiste una struttura record di destinazione del carico di lavoro del cluster per ciascuna istanza della coda di destinazione.

La struttura record di destinazione del carico di lavoro del cluster è supportata in tutti gli ambienti.

Inoltre, le strutture MQWDR1 e MQWDR2 sono disponibili per la compatibilità con le versioni precedenti.

Riferimenti correlati

MQ_CLUSTER_WORKLOAD_EXIT - Descrizione chiamata
L'uscita del carico di lavoro cluster viene richiamata dal gestore code per instradare un messaggio a un gestore code disponibile.

MQXCLWLN - Esplora record del carico di lavoro del cluster
La chiamata MQXCLWLN è utilizzata per esplorare le catene di record MQWDR, MQWQRe MQWCR memorizzati nella cache del cluster.

MQWXP - Struttura dei parametri di uscita del carico di lavoro cluster
La seguente tabella riepiloga i campi nella struttura del parametro di uscita del carico di lavoro del cluster MQWXP .

MQWQR - Struttura record coda carico di lavoro cluster
La seguente tabella riepiloga i campi in MQWQR - Struttura record della coda del workload del cluster.

MQWCR - Struttura di record del cluster di workload del cluster
La seguente tabella riepiloga i campi nella struttura di record del carico di lavoro del cluster MQWCR .

Campi in MQWDR - Struttura record di destinazione del carico di lavoro del cluster

Descrizione dei parametri in MQWDR - Struttura record di destinazione del carico di lavoro del cluster.

StrucId (MQCHAR4) - immissione
L'identificativo della struttura per la struttura record di destinazione del carico di lavoro cluster.

- Il valore StrucId è MQWDR_STRUC_ID.
- Per il linguaggio di programmazione C, viene definita anche la costante MQWDR_STRUC_ID_ARRAY . Ha lo stesso valore di MQWDR_STRUC_ID. Si tratta di un array di caratteri invece di una stringa.

Versione (MQLONG) - immissione
Il numero di versione della struttura. Versione assume uno dei seguenti valori:

MQWDR_VERSION_1
Record di destinazione del carico di lavoro del cluster Version-1 .

MQWDR_VERSION_2
Record di destinazione del carico di lavoro del cluster Version-2 .

MQWDR_CURRENT_VERSION
Versione corrente del record di destinazione del carico di lavoro del cluster.

StrucLength (MQLONG) - immissione
La lunghezza della struttura MQWDR . StrucLength assume uno dei seguenti valori:

MQWDR_LENGTH_1
Lunghezza del record di destinazione del carico di lavoro del cluster version-1 .

MQWDR_LENGTH_2
Lunghezza del record di destinazione del carico di lavoro del cluster version-2 .

MQWDR_CURRENT_LENGTH
Lunghezza della versione corrente del record di destinazione del carico di lavoro del cluster.

QMgrFlags (MQLONG) - immissione
Indicatori del gestore code che indicano le proprietà del gestore code su cui è presente l'istanza della coda di destinazione descritta dalla struttura MQWDR . Sono definiti i seguenti indicatori:

MQQMF_REPOSITORY_Q_MGR
La destinazione è un gestore code del repository completo.

MQQMF_CLUSSDR_USER_DEFINED
Il canale mittente del cluster è stato definito manualmente.

MQQMF_CLUSSDR_AUTO_DEFINED
Il canale mittente del cluster è stato definito automaticamente.

MQQMF_AVAILABLE
Il gestore code di destinazione è disponibile per ricevere i messaggi.

Altri valori
Altri indicatori nel campo potrebbero essere impostati dal gestore code per scopi interni.

QMgrIdentifier (MQCHAR48) - immissione
L'identificativo del gestore code è un identificativo univoco per il gestore code che contiene l'istanza della coda di destinazione descritta dalla struttura MQWDR .

- L'identificativo viene generato dal gestore code.
- La lunghezza di QMgrIdentifier è MQ_Q_MGR_IDENTIFIER_LENGTH.

QMgrName (MQCHAR48) - immissione
Il nome del gestore code su cui è presente l'istanza della coda di destinazione descritta dalla struttura MQWDR .

- QMgrName può essere il nome del gestore code locale e un altro gestore code nel cluster.
- La lunghezza di QMgrName è MQ_Q_MGR_NAME_LENGTH.

ClusterRecClusterRec (MQLONG) - immissione
L'offset logico della prima struttura MQWCR che appartiene alla struttura MQWDR .

- Per le cache statiche, ClusterRecClusterRec è l'offset della prima struttura MQWCR che appartiene alla struttura MQWDR .
- L'offset viene misurato in byte dall'inizio della struttura MQWDR .
- Non utilizzare lo scostamento logico per l'aritmetica del puntatore con cache dinamiche. Per ottenere l'indirizzo del record successivo, è necessario utilizzare la chiamata MQXCLWLN .

ChannelState (MQLONG) - immissione
Lo stato del canale che collega il gestore code locale al gestore code identificato dalla struttura MQWDR . Sono possibili i seguenti valori:

MQCHS_BINDING
Channel sta negoziando con il partner.

MQCHS_INACTIVE
Il canale non è attivo.

MQCHS_INITIALIZING
Il canale è in fase di inizializzazione.

MQCHS_PAUSED
Il canale è stato sospeso.

MQCHS_REQUESTING
Il canale richiedente sta richiedendo la connessione.

MQCHS_RETRYING
Il canale sta tentando di ristabilire la connessione.

MQCHS_RUNNING
Il canale è in fase di trasferimento o in attesa di messaggi.

MQCHS_STARTING
Il canale è in attesa di diventare attivo.

MQCHS_STOPPING
Il canale è in fase di arresto.

MQCHS_STOPPED
Il canale è stato arrestato.

ChannelDefChannelDef (MQLONG) - immissione
L'offset logico della definizione di canale (MQCD) per il canale che collega il gestore code locale a quello identificato dalla struttura MQWDR .

- ChannelDefOffset è come ClusterRecOffset
- Lo scostamento logico non può essere utilizzato nell'aritmetica del puntatore. Per ottenere l'indirizzo del record successivo, è necessario utilizzare la chiamata MQXCLWLN .

DestSeqDestSeq (MQLONG) - immissione
Il fattore di sequenza di destinazione che permette una scelta del canale in base al peso.

- DestSeqDestSeq viene utilizzato prima che il gestore code lo modifichi.
- Il gestore del carico di lavoro aumenta DestSeqFactor in modo da garantire che i messaggi vengano distribuiti nei canali in base al loro peso.

DestSeqDestSeq (MQLONG) - immissione
Il valore di destinazione del canale cluster prima che venga modificato dal gestore code.

- Il gestore del carico di lavoro aumenta DestSeqNumero ogni volta che un messaggio viene inserito in quel canale.
- Le uscite del carico di lavoro possono utilizzare DestSeqNumber per decidere quale canale disinserire un messaggio.

Riferimenti correlati
Valori iniziali e dichiarazioni di lingua per MQWDR
Valori iniziali e dichiarazioni di linguaggio C e High Level Assembler per MQWDR - Record di destinazione del carico di lavoro del cluster.

Valori iniziali e dichiarazioni di lingua per MQWDR
Valori iniziali e dichiarazioni di linguaggio C e High Level Assembler per MQWDR - Record di destinazione del carico di lavoro del cluster.

Tabella 830. Campi in MQWDR

Nome campo	Nome della costante	Valore della costante
StrucId	MQWDR_STRUC_ID	'WDR¬'
Version	MQWDR_VERSION_1	1
StrucLength	MQWDR_CURRENT_LENGTH [3]	136
QMgrFlags	MQWDR_NONE	0
QMgrIdentifier	Nessuna	""

Tabella 830. Campi in MQWDR (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
QMgrName	Nessuna	""
ClusterRecOffset	Nessuna	0
ChannelState	Nessuna	0
ChannelDefOffset	Nessuna	0
DestSeqNumber	Nessuna	0
DestSeqFactor	Nessuna	0

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macro MQWDR_DEFAULT contiene i valori predefiniti. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

   ```
   MQWDR MyWDR = {MQWDR_DEFAULT};
   ```

3. I valori iniziali impostano intenzionalmente la lunghezza della struttura sulla lunghezza della versione corrente e non sulla versione 1 della struttura.

High Level Assembler

```
MQWDR                           DSECT
MQWDR_STRUCID                   DS   CL4      Structure identifier
MQWDR_VERSION                   DS   F        Structure version number
MQWDR_STRUCLENGTH               DS   F        Length of MQWDR structure
MQWDR_QMGRFLAGS                 DS   F        Queue manager flags
MQWDR_QMGRIDENTIFIER            DS   CL48     Queue manager identifier
MQWDR_QMGRNAME                  DS   CL48     Queue manager name
MQWDR_CLUSTERRECOFFSET          DS   F        Offset of first cluster
*                                            record
MQWDR_CHANNELSTATE              DS   F        Channel state
MQWDR_CHANNELDEFOFFSET          DS   F        Offset of channel definition
*                                            structure
MQWDR_LENGTH                    EQU  *-MQWDR  Length of structure
                                ORG  MQWDR
MQWDR_AREA                      DS   CL(MQWDR_LENGTH)
```

Dichiarazione C

```
typedef struct tagMQWDR {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    StrucLength;       /* Length of MQWDR structure */
  MQLONG    QMgrFlags;         /* Queue managerflags */
  MQCHAR48  QMgrIdentifier;    /* Queue manageridentifier */
  MQCHAR48  QMgrName;          /* Queue manager name */
  MQLONG    ClusterRecOffset;  /* Offset of first cluster record */
  MQLONG    ChannelState;      /* Channel state */
  MQLONG    ChannelDefOffset;  /* Offset of channel definition structure */
  /* Ver:1 */
  MQLONG    DestSeqNumber;     /* Cluster channel destination sequence number */
  MQINT64   DestSeqFactor;     /* Cluster channel factor sequence number */
  /* Ver:2 */
};
```

Riferimenti correlati

Campi in MQWDR - Struttura record di destinazione del carico di lavoro del cluster
Descrizione dei parametri in MQWDR - Struttura record di destinazione del carico di lavoro del cluster.

MQWQR - Struttura record coda carico di lavoro cluster

La seguente tabella riepiloga i campi in MQWQR - Struttura record della coda del workload del cluster.

Tabella 831. Campi in MQWQR		
Campo	**Descrizione**	**Pagina**
StrucId	Identificativo struttura	StrucId
Version	Numero di versione della struttura	Versione
StrucLength	Lunghezza della struttura MQWQR	StrucLength
QFlags	Indicatori coda	QFlag
QName	Nome coda	QName
QMgrIdentifier	Identificativo gestore code	QMgrIdentifier
ClusterRecOffset	Offset del primo record cluster (MQWCR)	ClusterRecClusterRec
QType	Tipo coda	QTYPE
QDesc	Descrizione coda	QDesc
DefBind	Binding predefinito	DefBind
DefPersistence	Persistenza predefinita messaggio	DefPersistence
DefPriority	Priorità predefinita messaggio	DefPriority
InhibitPut	Se le operazioni di inserimento sulla coda sono consentite	InhibitPut
Nota: I restanti campi vengono ignorati se la versione è inferiore a MQWQR_VERSION_2.		
CWLQueuePriority	Un valore compreso tra 0 e 9 che rappresenta la priorità della coda	CLWLQueuePriority
CLWLQueueRank	Un valore compreso tra 0 e 9 che rappresenta la classificazione della coda	CLWLQueueRank
Nota: I restanti campi vengono ignorati se la versione è inferiore a MQWQR_VERSION_3.		
DefPutResponse	Risposta inserimento predefinita	RispostaDefPut

La struttura di record della coda del carico di lavoro del cluster contiene informazioni relative a una delle possibili destinazioni per il messaggio. Esiste una struttura di record della coda del carico di lavoro del cluster per ciascuna istanza della coda di destinazione.

La struttura di record della coda del carico di lavoro del cluster è supportata in tutti gli ambienti.

Inoltre, le strutture MQWQR1 e MQWQR2 sono disponibili per la compatibilità con le versioni precedenti.

Riferimenti correlati

MQ_CLUSTER_WORKLOAD_EXIT - Descrizione chiamata
L'uscita del carico di lavoro cluster viene richiamata dal gestore code per instradare un messaggio a un gestore code disponibile.

MQXCLWLN - Esplora record del carico di lavoro del cluster
La chiamata MQXCLWLN è utilizzata per esplorare le catene di record MQWDR, MQWQRe MQWCR memorizzati nella cache del cluster.

MQWXP - Struttura dei parametri di uscita del carico di lavoro cluster
La seguente tabella riepiloga i campi nella struttura del parametro di uscita del carico di lavoro del cluster MQWXP .

MQWDR - Struttura record di destinazione del carico di lavoro del cluster
La seguente tabella riepiloga i campi in MQWDR - Struttura di record di destinazione del carico di lavoro del cluster.

MQWCR - Struttura di record del cluster di workload del cluster
La seguente tabella riepiloga i campi nella struttura di record del carico di lavoro del cluster MQWCR .

Campi in MQWQR - Struttura record della coda del carico di lavoro del cluster

Descrizione dei campi in MQWQR - Struttura record della coda del carico di lavoro del cluster.

StrucId (MQCHAR4) - immissione
L'identificativo della struttura per la struttura di record della coda del carico di lavoro del cluster.

- Il valore StrucId è MQWQR_STRUC_ID.
- Per il linguaggio di programmazione C, viene definita anche la costante MQWQR_STRUC_ID_ARRAY . Ha lo stesso valore di MQWQR_STRUC_ID. Si tratta di un array di caratteri invece di una stringa.

Versione (MQLONG) - immissione
Il numero di versione della struttura. Versione assume uno dei seguenti valori:

MQWQR_VERSION_1
Version-1 record della coda del carico di lavoro del cluster.

MQWQR_VERSION_2
Version-2 record della coda del carico di lavoro del cluster.

MQWQR_VERSION_3
Version-3 record della coda del workload del cluster.

MQWQR_CURRENT_VERSION
Versione corrente del record della coda del carico di lavoro del cluster.

StrucLength (MQLONG) - immissione
La lunghezza della struttura MQWQR . StrucLength assume uno dei seguenti valori:

MQWQR_LENGTH_1
Lunghezza del record della coda del carico di lavoro cluster version-1 .

MQWQR_LENGTH_2
Lunghezza del record della coda del carico di lavoro del cluster version-2 .

MQWQR_LENGTH_3
Lunghezza del record della coda del carico di lavoro del cluster version-3 .

MQWQR_CURRENT_LENGTH
Lunghezza della versione corrente del record della coda del carico di lavoro del cluster.

QFlag (MQLONG) - immissione
Gli indicatori della coda indicano le proprietà della coda. Sono definiti i seguenti indicatori:

MQQF_LOCAL_Q
La destinazione è una coda locale.

MQQF_CLWL_USEQ_ANY
Consente l'utilizzo di code locali e remote negli inserimenti.

MQQF_CLWL_USEQ_LOCAL
Consenti solo inserimenti coda locale.

Altri valori
Altri indicatori nel campo potrebbero essere impostati dal gestore code per scopi interni.

QName (MQCHAR48) - immissione
Il nome della coda che è una delle possibili destinazioni del messaggio.

- La lunghezza di QName è MQ_Q_NAME_LENGTH.

QMgrIdentifier (MQCHAR48) - immissione
L'identificativo del gestore code è un identificativo univoco per il gestore code che ospita l'istanza della coda descritta dalla struttura MQWQR .

- L'identificativo viene generato dal gestore code.
- La lunghezza di QMgrIdentifier è MQ_Q_MGR_IDENTIFIER_LENGTH.

ClusterRecClusterRec (MQLONG) - immissione
L'offset logico della prima struttura MQWCR appartenente alla struttura MQWQR .

- Per le cache statiche, ClusterRecOffset è l'offset della prima struttura MQWCR appartenente alla struttura MQWQR .
- L'offset viene misurato in byte dall'inizio della struttura MQWQR .
- Non utilizzare lo scostamento logico per l'aritmetica del puntatore con cache dinamiche. Per ottenere l'indirizzo del record successivo, è necessario utilizzare la chiamata MQXCLWLN .

QType (MQLONG) - immissione
Il tipo di coda della coda di destinazione. Sono possibili i seguenti valori:

MQCQT_LOCAL_Q
Coda locale.

MQCQT_ALIAS_Q
Coda alias.

MQCQT_REMOTE_ Q
Coda remota.

MQCQT_Q_MGR_ALIAS
Alias del gestore code.

QDesc (MQCHAR64) - immissione
L'attributo della coda di descrizione della coda definito sul gestore code che ospita l'istanza della coda di destinazione descritta dalla struttura MQWQR .

- La lunghezza di QDesc è MQ_Q_DESC_LENGTH.

DefBind (MQLONG) - immissione
L'attributo predefinito della coda di bind definito nel gestore code che ospita l'istanza della coda di destinazione descritta dalla struttura MQWQR . È necessario specificare MQBND_BIND_ON_OPEN o MQBND_BIND_ON_GROUP quando si utilizzano gruppi con cluster. Sono possibili i seguenti valori:

MQBND_BIND_ON_OPEN
Collegamento corretto dalla chiamata MQOPEN .

MQBND_BIND_NOT_FIXED
Collegamento non corretto.

MQBND_BIND_ON_GROUP
Consente a una applicazione di richiedere che un gruppo di messaggi sia assegnato alla stessa istanza di destinazione.

DefPersistence (MQLONG) - immissione
L'attributo predefinito della coda di persistenza dei messaggi definito sul gestore code che ospita l'istanza della coda di destinazione descritta nella struttura MQWQR . Sono possibili i seguenti valori:

MQPER_PERSISTENT
Il messaggio è persistente.

MQPER_NOT_PERSISTENT
Il messaggio non è persistente.

DefPriority (MQLONG) - immissione

L'attributo predefinito della coda di priorità dei messaggi definito sul gestore code che ospita l'istanza della coda di destinazione descritta nella struttura MQWQR . L'intervallo di priorità è 0 - MaxPriority.

- 0 è la priorità più bassa.
- MaxPriority è l'attributo del gestore code che ospita questa istanza della coda di destinazione.

InhibitPut (MQLONG) - immissione
L'attributo della coda di immissione inibita definito nel gestore code che ospita l'istanza della coda di destinazione descritta dalla struttura MQWQR . Sono possibili i seguenti valori:

MQQA_PUT_INHIBITED
Le operazioni di inserimento sono inibite.

MQQA_PUT_ALLOWED
Le operazioni di inserimento sono consentite.

CLWLQueuePriority (MQLONG) - immissione
L'attributo della priorità della coda del carico di lavoro del cluster definito sul gestore code che ospita l'istanza della coda di destinazione descritta dalla struttura MQWQR .

CLWLQueueRank (MQLONG) - immissione
La classificazione della coda del carico di lavoro del cluster definita sul gestore code che ospita l'istanza della coda di destinazione descritta nella struttura MQWQR .

Risposta DefPut (MQLONG) - immissione
L'attributo predefinito della coda di risposta di inserimento definito nel gestore code che ospita l'istanza della coda di destinazione descritta dalla struttura MQWQR . Sono possibili i seguenti valori:

MQPRT_SYNC_RESPONSE
Risposta sincrona alle chiamate MQPUT o MQPUT1 .

MQPRT_ASYNC_RESPONSE
Risposta asincrona alle chiamate MQPUT o MQPUT1 .

Riferimenti correlati

Valori iniziali e dichiarazioni di lingua per MQWQR
Valori iniziali e dichiarazioni di linguaggio C e High Level Assembler per il record della coda del workload del cluster MQWQR .

Valori iniziali e dichiarazioni di lingua per MQWQR

Valori iniziali e dichiarazioni di linguaggio C e High Level Assembler per il record della coda del workload del cluster MQWQR .

Tabella 832. Campi in MQWQR

Nome campo	Nome della costante	Valore della costante
StrucId	MQWQR_STRUC_ID_ARRAY	'WQR¬'
Version	MQWQR_VERSION_1	1
StrucLength	MQWQR_CURRENT_LENGTH [3]	212
QFlags	Nessuna	0
QName	Nessuna	""
QMgrIdentifier	Nessuna	""
ClusterRecOffset	Nessuna	0
QType	Nessuna	0
QDesc	Nessuna	""
DefBind	Nessuna	0
DefPersistence	Nessuna	0
DefPriority	Nessuna	0
InhibitPut	Nessuna	0
CLWLQueuePriority	Nessuna	0

Tabella 832. Campi in MQWQR (Continua)		
Nome campo	**Nome della costante**	**Valore della costante**
CLWLQueueRank	Nessuna	0
DefPutResponse	Nessuna	1

Note:

1. Il simbolo ¬ rappresenta un singolo carattere vuoto.
2. Nel linguaggio di programmazione C, la variabile macro MQWQR_DEFAULT contiene i valori predefiniti. Utilizzarlo nel modo riportato di seguito per fornire i valori iniziali per i campi nella struttura:

   ```
   MQWQR MyWQR = {MQWQR_DEFAULT};
   ```

3. I valori iniziali impostano intenzionalmente la lunghezza della struttura sulla lunghezza della versione corrente e non sulla versione 1 della struttura.

Dichiarazione C

```
typedef struct tagMQWQR {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    StrucLength;       /* Length of MQWQR structure */
  MQLONG    QFlags;            /* Queue flags */
  MQCHAR48  QName;             /* Queue name */
  MQCHAR48  QMgrIdentifier;    /* Queue manageridentifier */
  MQLONG    ClusterRecOffset;  /* Offset of first cluster record */
  MQLONG    QType;             /* Queue type */
  MQCHAR64  QDesc;             /* Queue description */
  MQLONG    DefBind;           /* Default binding */
  MQLONG    DefPersistence;    /* Default message persistence */
  MQLONG    DefPriority;       /* Default message priority */
  MQLONG    InhibitPut;        /* Whether put operations on the queue
                                  are allowed */
  /* version 2 */
  MQLONG    CLWLQueuePriority; /* Queue priority */
  MQLONG    CLWLQueueRank;     /* Queue rank */
  /* version 3 */
  MQLONG    DefPutResponse;    /* Default put response */
};
```

High Level Assembler

```
MQWQR                     DSECT
MQWQR_STRUCID             DS   CL4      Structure identifier
MQWQR_VERSION             DS   F        Structure version number
MQWQR_STRUCLENGTH         DS   F        Length of MQWQR structure
MQWQR_QFLAGS              DS   F        Queue flags
MQWQR_QNAME               DS   CL48     Queue name
MQWQR_QMGRIDENTIFIER      DS   CL48     Queue manager identifier
MQWQR_CLUSTERRECOFFSET    DS   F        Offset of first cluster
*                                       record
MQWQR_QTYPE               DS   F        Queue type
MQWQR_QDESC               DS   CL64     Queue description
MQWQR_DEFBIND             DS   F        Default binding
MQWQR_DEFPERSISTENCE      DS   F        Default message persistence
MQWQR_DEFPRIORITY         DS   F        Default message priority
MQWQR_INHIBITPUT          DS   F        Whether put operations on
*                                       the queue are allowed
MQWQR_DEFPUTRESPONSE      DS   F        Default put response
MQWQR_LENGTH              EQU  *-MQWQR  Length of structure
                          ORG  MQWQR
MQWQR_AREA                DS   CL(MQWQR_LENGTH)
```

Riferimenti correlati

Campi in MQWQR - Struttura record della coda del carico di lavoro del cluster
Descrizione dei campi in MQWQR - Struttura record della coda del carico di lavoro del cluster.

MQWCR - Struttura di record del cluster di workload del cluster

La seguente tabella riepiloga i campi nella struttura di record del carico di lavoro del cluster MQWCR .

Tabella 833. Campi in MQWCR		
Campo	**Descrizione**	**Pagina**
ClusterName	Nome del cluster	ClusterName
ClusterRecOffset	Offset del record cluster successivo (MQWCR)	ClusterRecClusterRec
ClusterFlags	Indicatori cluster	ClusterFlags

La struttura di record del cluster di workload del cluster contiene informazioni su un cluster. Per ogni cluster a cui appartiene la coda di destinazione, esiste una struttura di record del cluster di workload del cluster.

La struttura del record cluster di workload del cluster è supportata in tutti gli ambienti.

Riferimenti correlati

MQ_CLUSTER_WORKLOAD_EXIT - Descrizione chiamata
L'uscita del carico di lavoro cluster viene richiamata dal gestore code per instradare un messaggio a un gestore code disponibile.

MQXCLWLN - Esplora record del carico di lavoro del cluster
La chiamata MQXCLWLN è utilizzata per esplorare le catene di record MQWDR, MQWQRe MQWCR memorizzati nella cache del cluster.

MQWXP - Struttura dei parametri di uscita del carico di lavoro cluster
La seguente tabella riepiloga i campi nella struttura del parametro di uscita del carico di lavoro del cluster MQWXP .

MQWDR - Struttura record di destinazione del carico di lavoro del cluster
La seguente tabella riepiloga i campi in MQWDR - Struttura di record di destinazione del carico di lavoro del cluster.

MQWQR - Struttura record coda carico di lavoro cluster
La seguente tabella riepiloga i campi in MQWQR - Struttura record della coda del workload del cluster.

Campi in MQWCR - Struttura di record del cluster di workload del cluster di cluster.

Descrizione dei campi in MQWCR - Struttura record cluster di workload del cluster.

ClusterName (MQCHAR48) - immissione
Il nome di un cluster a cui appartiene l'istanza della coda di destinazione proprietaria della struttura MQWCR . L'istanza della coda di destinazione è descritta da una struttura MQWDR .

- La lunghezza di ClusterName è MQ_CLUSTER_NAME_LENGTH.

ClusterRecClusterRec (MQLONG) - immissione
L'offset logico della successiva struttura MQWCR .

- Se non ci sono più strutture MQWCR , ClusterRecOffset è zero.
- L'offset viene misurato in byte dall'inizio della struttura MQWCR .

ClusterFlags (MQLONG) - immissione
Gli indicatori del cluster indicano le proprietà del gestore code identificato dalla struttura MQWCR . Sono definiti i seguenti indicatori:

MQQMF_REPOSITORY_Q_MGR
La destinazione è un gestore code del repository completo.

MQQMF_CLUSSDR_USER_DEFINED
Il canale mittente del cluster è stato definito manualmente.

MQQMF_CLUSSDR_AUTO_DEFINED
Il canale mittente del cluster è stato definito automaticamente.

MQQMF_AVAILABLE
Il gestore code di destinazione è disponibile per ricevere i messaggi.

Altri valori
Altri indicatori nel campo potrebbero essere impostati dal gestore code per scopi interni.

Riferimenti correlati

Valori iniziali e dichiarazioni di lingua per MQWCR
Valori iniziali e dichiarazioni linguaggio C e High Level Assembler per MQWCR - Struttura di record del cluster di workload del cluster di cluster.

Valori iniziali e dichiarazioni di lingua per MQWCR

Valori iniziali e dichiarazioni linguaggio C e High Level Assembler per MQWCR - Struttura di record del cluster di workload del cluster di cluster.

Tabella 834. Campi in MQWCR

Nome campo	Nome della costante	Valore della costante
ClusterName	Nessuna	""
ClusterRecOffset	Nessuna	0
ClusterFlags	Nessuna	0

Dichiarazione C

```
typedef struct tagMQWCR {
  MQCHAR48  ClusterName;       /* Cluster name */
  MQLONG    ClusterRecOffset;  /* Offset of next cluster record */
  MQLONG    ClusterFlags;      /* Cluster flags */
};
```

High Level Assembler

```
MQWCR                           DSECT
MQWCR_CLUSTERNAME               DS   CL48     Cluster name
MQWCR_CLUSTERRECOFFSET          DS   F        Offset of next cluster
*                                            record
MQWCR_CLUSTERFLAGS              DS   F        Cluster flags
MQWCR_LENGTH                    EQU  *-MQWCR  Length of structure
                                ORG  MQWCR
MQWCR_AREA                      DS   CL(MQWCR_LENGTH)
```

Riferimenti correlati

Campi in MQWCR - Struttura di record del cluster di workload del cluster di cluster.
Descrizione dei campi in MQWCR - Struttura record cluster di workload del cluster.

Riferimento uscita API

Questa sezione fornisce informazioni di riferimento principalmente di interesse per un programmatore che scrive uscite API.

Note generali sull'utilizzo

::

1. Tutte le funzioni di uscita possono emettere la chiamata MQXEP; questa chiamata è progettata specificamente per essere utilizzata dalle funzioni di uscita API.
2. La funzione MQ_INIT_EXIT non può emettere chiamate MQ diverse da MQXEP.
3. Non è possibile eseguire la chiamata MQDISC per la connessione corrente.
4. Se una funzione di uscita emette la chiamata MQCONN o la chiamata MQCONNX con l'opzione MQCNO_HANDLE_SHARE_NONE, la chiamata viene completata con il codice motivo MQRC_ALREADY_CONNECTED e l'handle restituito è lo stesso di quello passato all'uscita come parametro.
5. In generale, quando una funzione di uscita API emette una chiamata MQI, le uscite API non vengono richiamate in modo ricorsivo. Tuttavia, se una funzione di uscita emette la chiamata MQCONNX con le opzioni MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK, la chiamata restituisce un nuovo handle condiviso. Ciò fornisce alla suite di uscita un proprio handle di connessione, e quindi un'unità di lavoro indipendente dall'unità di lavoro dell'applicazione. La suite di uscita può utilizzare questo handle per inserire e richiamare i messaggi all'interno della propria unità di lavoro ed eseguire il commit o il backout di tale unità di lavoro; tutto ciò può essere fatto senza influenzare in alcun modo l'unità di lavoro dell'applicazione.

   Poiché la funzione di uscita sta utilizzando un handle di connessione diverso da quello utilizzato dall'applicazione, le chiamate MQ emesse dalla funzione di uscita risultano nel richiamo delle funzioni di uscita API pertinenti. Le funzioni di uscita possono essere richiamate in modo ricorsivo. Tenere presente che sia il campo *ExitUserArea* in MQAXP che l'area della catena di uscita hanno un ambito di gestione connessione. Di conseguenza, una funzione di uscita non può utilizzare tali aree per segnalare a un'altra istanza di se stessa richiamata in maniera ricorsiva che è già attiva.
6. Le funzioni di uscita possono anche inserire e richiamare messaggi all'interno dell'unità di lavoro dell'applicazione. Quando l'applicazione esegue il commit o il backout dell'unità di lavoro, viene eseguito il commit o il backout di tutti i messaggi all'interno dell'unità di lavoro, indipendentemente da chi li ha inseriti nell'unità di lavoro (applicazione o funzione di uscita). Tuttavia, l'uscita può far sì che l'applicazione superi i limiti del sistema prima di quanto non sarebbe altrimenti (ad esempio, superando il numero massimo di messaggi senza commit in un'unità di lavoro).

   Quando una funzione di uscita utilizza l'unità di lavoro dell'applicazione in questo modo, la funzione di uscita in genere dovrebbe evitare di emettere la chiamata MQCMIT, poiché ciò esegue il commit dell'unità di lavoro dell'applicazione e potrebbe compromettere il corretto funzionamento dell'applicazione. Tuttavia, la funzione exit potrebbe a volte dover emettere la chiamata MQBACK, se la funzione exit rileva un errore grave che impedisce il commit dell'unità di lavoro (ad esempio, un errore che inserisce un messaggio come parte dell'unità di lavoro dell'applicazione). Quando viene richiamato MQBACK, assicurarsi che i limiti dell'unità di lavoro dell'applicazione non vengano modificati. In questa situazione la funzione di uscita deve impostare i valori appropriati per garantire che il codice di completamento MQCC_WARNING e il codice motivo MQRC_BACKED_OUT vengano restituiti all'applicazione, in modo che l'applicazione possa rilevare il fatto che è stato eseguito il backout dell'unità di lavoro.

   Se una funzione di uscita utilizza l'handle di connessione dell'applicazione per emettere chiamate MQ , tali chiamate non determinano ulteriori richiami delle funzioni di uscita API.
7. Se una funzione di uscita MQXR_BEFORE viene terminata in modo anomalo, il gestore code potrebbe essere in grado di eseguire il ripristino dall'errore. Se possibile, il gestore code continua l'elaborazione come se la funzione di uscita avesse restituito MQXCC_FAILED. Se il gestore code non è in grado di eseguire il ripristino, l'applicazione viene terminata.
8. Se una funzione di uscita MQXR_AFTER termina in maniera anomala, il gestore code potrebbe essere in grado di eseguire il ripristino dall'errore. Se possibile, il gestore code continua l'elaborazione come se la funzione di uscita avesse restituito MQXCC_FAILED. Se il gestore code non è in grado di eseguire il ripristino, l'applicazione viene terminata. Tenere presente che in quest' ultimo caso, i messaggi richiamati al di fuori di un'unità di lavoro vengono persi (questa è la stessa situazione dell'errore dell'applicazione immediatamente dopo la rimozione di un messaggio dalla coda).
9. Il processo MCA esegue un commit a due fasi.

Se un'uscita API intercetta un MQCMIT da un processo MCA preparato e tenta di eseguire un'azione all'interno dell'unità di lavoro, l'azione avrà esito negativo con codice motivo MQRC_UOW_NOT_AVAILABLE.

10. Se sono disponibili più installazioni di IBM MQ , utilizzare le uscite scritte per una versione precedente di IBM MQ, poiché la nuova funzionalità aggiunta nella versione successiva potrebbe non funzionare con le versioni precedenti. Per ulteriori informazioni sulle modifiche tra le release, consultare What's changed in IBM MQ 8.0.

Struttura del parametro di uscita API IBM MQ (MQAXP)

La struttura MQAXP, un blocco di controllo esterno, viene utilizzata come parametro di input o output per l'uscita API. Questo argomento fornisce anche informazioni su come i gestori code elaborano le funzioni di uscita.

MQAXP ha la dichiarazione C seguente:

```
typedef struct tagMQAXP {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    ExitId;            /* Exit Identifier */
  MQLONG    ExitReason;        /* Exit invocation reason */
  MQLONG    ExitResponse;      /* Response code from exit */
  MQLONG    ExitResponse2;     /* Secondary response code from exit */
  MQLONG    Feedback;          /* Feedback code from exit */
  MQLONG    APICallerType;     /* MQSeries API caller type */
  MQBYTE16  ExitUserArea;      /* User area for use by exit */
  MQCHAR32  ExitData;          /* Exit data area */
  MQCHAR48  ExitInfoName;      /* Exit information name */
  MQBYTE48  ExitPDArea;        /* Problem determination area */
  MQCHAR48  QMgrName;          /* Name of local queue manager */
  PMQACH    ExitChainAreaPtr;  /* Inter exit communication area */
  MQHCONFIG Hconfig;           /* Configuration handle */
  MQLONG    Function;          /* Function Identifier */
  /* Ver:1 */
  MQHMSG    ExitMsgHandle      /* Exit message handle
  /* Ver:2 */
};
```

Il seguente elenco di parametri viene passato quando vengono richiamate le funzioni in un'uscita API:

StrucId (MQCHAR4) - input
L'identificativo della struttura del parametro di uscita, con un valore di:

```
MQAXP_STRUC_ID.
```

Il gestore uscite imposta questo campo all'entrata di ciascuna funzione di uscita.

Versione (MQLONG) - input
Il numero di versione della struttura, con un valore di:

MQAXP_VERSION_1
Struttura del parametro di uscita API versione 1.

MQAXP_VERSION_2
Struttura del parametro di uscita API versione 2.

VERSIONE MQAXP_CURRENT_
Numero di versione corrente per la struttura del parametro di uscita API.

Il gestore uscite imposta questo campo all'entrata di ciascuna funzione di uscita.

ExitId (MQLONG) - input
L'identificativo di uscita, impostato all'entrata della routine di uscita, che indica il tipo di uscita:

MQXT_API_EXIT
Uscita API.

ExitReason (MQLONG) - input
Il motivo per richiamare l'exit, impostato all'entrata per ciascuna funzione di exit:

MQXR_CONNECZIONE
L'uscita viene richiamata per inizializzare se stessa prima di una chiamata MQCONN o MQCONNX o per terminare se stessa dopo una chiamata MQDISC.

MQXR_BEFORE
L'uscita viene richiamata prima di eseguire una chiamata API o prima di convertire i dati in un MQGET.

MQXR_XX_ENCODE_CASE_ONE dopo
L'uscita viene richiamata dopo l'esecuzione di una chiamata API.

ExitResponse (MQLONG) - output
La risposta dall'uscita, inizializzata all'ingresso in ogni funzione di uscita per:

MQXCC_OK
Continuare normalmente.

Questo campo deve essere impostato dalla funzione di uscita, per comunicare al gestore code il risultato dell'esecuzione della funzione di uscita. Il valore deve essere uno dei seguenti.

MQXCC_OK
La funzione di uscita è stata completata correttamente. Continuare normalmente.

Questo valore può essere impostato da tutte le funzioni di uscita MQXR_ *. ExitResponse2 viene utilizzato per decidere se richiamare le funzioni di uscita successivamente nella catena.

MQXCC_NON RIUSCITO
La funzione di uscita non è riuscita a causa di un errore.

Questo valore può essere impostato da tutte le funzioni di uscita MQXR_ *. Il gestore code imposta CompCode su MQCC_FAILED e Reason su:

- MQRC_API_EXIT_INIT_ERROR se la funzione è MQ_INIT_EXIT
- MQRC_API_EXIT_TERM_ERROR se la funzione è MQ_TERM_EXIT
- MQRC_API_EXIT_ERROR per tutte le altre funzioni di uscita

I valori impostati possono essere modificati da una funzione di uscita successivamente nella catena.

ExitResponse2 viene ignorato; il gestore code continua l'elaborazione come se fosse stato restituito MQXR2_SUPPRESS_CHAIN .

MQXCC_SUPPRESS_FUNZIONE
Sopprimere la funzione API IBM MQ .

Questo valore può essere impostato solo da una funzione di uscita MQXR_BEFORE. Ignora la chiamata API. Se viene restituito da MQ_DATA_CONV_ON_GET_EXIT, la conversione dei dati viene ignorata. Il gestore code imposta CompCode su MQCC_FAILED e Motivo su MQRC_SUPPRESSED_BY_EXIT, ma i valori impostati possono essere modificati da una funzione di uscita successivamente nella catena. Altri parametri per la chiamata rimangono come l'uscita li ha lasciati. ExitResponse2 viene utilizzato per decidere se richiamare le funzioni di uscita successivamente nella catena.

Se questo valore è impostato da una funzione di uscita MQXR_AFTER o MQXR_CONNECTION, il gestore code continua l'elaborazione come se fosse stato restituito MQXCC_FAILED.

FUNZIONE SKIP_MQXCC
Ignorare la funzione API IBM MQ .

Questo valore può essere impostato solo da una funzione di uscita MQXR_BEFORE. Ignora la chiamata API. Se viene restituito da MQ_DATA_CONV_ON_GET_EXIT, la conversione dei dati viene ignorata. La funzione di exit deve impostare CompCode e Reason sui valori da restituire all'applicazione, ma i valori impostati possono essere modificati da una funzione di exit successivamente nella catena. Altri parametri per la chiamata rimangono come l'uscita li ha lasciati. ExitResponse2 viene utilizzato per decidere se richiamare le funzioni di uscita successivamente nella catena.

Se questo valore è impostato da una funzione di uscita MQXR_AFTER o MQXR_CONNECTION, il gestore code continua l'elaborazione come se fosse stato restituito MQXCC_FAILED.

MQXCC_SUPPRESS_EXIT
Elimina tutte le funzioni di uscita appartenenti alla serie di uscite.

Questo valore può essere impostato solo dalle funzioni di uscita MQXR_BEFORE e MQXR_AFTER. Ignora *tutte* le chiamate successive delle funzioni di uscita appartenenti a questa serie di uscite per questa connessione logica. Questo errore continua fino a quando si verifica la richiesta di disconnessione logica, quando la funzione MQ_TERM_EXIT viene richiamata con un ExitReason di MQXR_CONNECTION.

La funzione di exit deve impostare CompCode e Reason sui valori da restituire all'applicazione, ma i valori impostati possono essere modificati da una funzione di exit successivamente nella catena. Altri parametri per la chiamata rimangono come l'uscita li ha lasciati. ExitResponse2 viene ignorato.

Se questo valore è impostato da una funzione di uscita MQXR_CONNECTION, il gestore code continua l'elaborazione come se fosse stato restituito MQXCC_FAILED.

Per informazioni sull'interazione tra ExitResponse e ExitResponse2e il suo effetto sull'elaborazione dell'uscita, consultare "Come i gestori code elaborano le funzioni di uscita" a pagina 1610.

ExitResponse2 (MQLONG) - output
Questo è un codice di risposta di uscita secondario che qualifica il codice di risposta di uscita principale per le funzioni di uscita MQXR_BEFORE. Viene inizializzato per:

```
MQXR2_DEFAULT_CONTINUATION
```

all'entrata in una funzione di uscita chiamata API IBM MQ . Può quindi essere impostato su uno dei valori:

MQXR2_DEFAULT_CONTINUATION
Indica se continuare con la successiva uscita nella catena, in base al valore di ExitResponse.

Se ExitResponse è MQXCC_SUPPRESS_FUNCTION o MQXCC_SKIP_FUNCTION, ignorare le funzioni di uscita in un secondo tempo nella catena MQXR_BEFORE e le funzioni di uscita corrispondenti nella catena MQXR_AFTER. Richiamare le funzioni di uscita nella catena MQXR_AFTER che corrispondono alle funzioni di uscita precedenti nella catena MQXR_BEFORE.

Altrimenti, richiamare l'uscita successiva nella catena.

MQXR2_SUPPRESS_CHAIN
Sopprimere la catena.

Ignorare le funzioni di uscita in un momento successivo nella concatenazione MQXR_BEFORE e le corrispondenti funzioni di uscita nella concatenazione MQXR_AFTER per questa chiamata API. Richiamare le funzioni di uscita nella catena MQXR_AFTER che corrispondono alle funzioni di uscita precedenti nella catena MQXR_BEFORE.

MQXR2_CONTINUE_CHAIN
Continuare con la successiva uscita nella catena.

Per informazioni sull'interazione tra ExitResponse e ExitResponse2e il suo effetto sull'elaborazione dell'uscita, consultare "Come i gestori code elaborano le funzioni di uscita" a pagina 1610.

Feedback (MQLONG) - input/output
Comunicare i codici di feedback tra i richiami della funzione di uscita. Viene inizializzato per:

```
MQFB_NONE (0)
```

prima di richiamare la prima funzione della prima uscita in una catena.

Le uscite possono impostare questo campo su qualsiasi valore, incluso qualsiasi valore MQFB_ * o MQRC_ * valido. Le uscite possono inoltre impostare questo campo su un valore di feedback definito dall'utente compreso nell'intervallo tra MQFB_APPL_FIRST e MQFB_APPL_LAST.

APICallerType (MQLONG) - input
Il tipo di chiamante API, che indica se il chiamante API IBM MQ è esterno o interno al gestore code: MQXACT_EXTERNAL o MQXACT_INTERNAL.

Area ExitUser(MQBYTE16) - input/output
Un'area utente, disponibile per tutte le uscite associate a uno specifico oggetto ExitInfo. Viene inizializzato in MQXUA_NONE (zeri binari per la lunghezza dell'Area ExitUser) prima di richiamare la prima funzione di uscita (MQ_INIT_EXIT) per hconn. Da quel momento in poi, tutte le modifiche apportate a questo campo da una funzione di uscita vengono conservate nei richiami delle funzioni della stessa uscita.

Questo campo è allineato a un multiplo di 4 MQLONG.

Le uscite possono anche ancorare qualsiasi memoria allocata da questa area.

Per ogni hconn, ogni uscita in una catena di uscite ha un'area ExitUserdifferente. L'area ExitUsernon può essere condivisa dalle uscite in una catena e il contenuto dell'area ExitUserper un'uscita non è disponibile per un'altra uscita in una catena.

Per i programmi C, la costante MQXUA_NONE_ARRAY è definita anche con lo stesso valore di MQXUA_NONE, ma come array di caratteri invece di una stringa.

La lunghezza di questo campo è fornita da MQ_EXIT_USER_AREA_LENGTH.

ExitData (MQCHAR32) - input
Dati di uscita, impostati sull'input di ciascuna funzione di uscita sui 32 caratteri dei dati specifici di uscita forniti nell'uscita. Se non si definisce alcun valore nell'uscita, questo campo è vuoto.

La lunghezza di questo campo è fornita da MQ_EXIT_DATA_LENGTH.

Nome ExitInfo(MQCHAR48) - input
Il nome delle informazioni di uscita, impostato sull'input per ciascuna funzione di uscita per ApiExit_name specificato nelle definizioni di uscita nelle stanze.

ExitPDArea (MQBYTE48) - input/output
Un'area di determinazione dei problemi, inizializzata su MQXPDA_NONE (zeri binari per la lunghezza del campo) per ogni richiamo di una funzione di uscita.

Per i programmi C, anche la costante MQXPDA_NONE_ARRAY è definita con lo stesso valore di MQXPDA_NONE, ma come un array di caratteri invece di una stringa.

Il gestore di uscita scrive sempre questa area nella traccia IBM MQ alla fine di un'uscita, anche quando la funzione ha esito positivo.

La lunghezza di questo campo è fornita da MQ_EXIT_PD_AREA_LENGTH.

QMgrName (MQCHAR48) - input
Il nome del gestore code a cui è connessa l'applicazione, che ha richiamato un'uscita come risultato dell'elaborazione di una chiamata API IBM MQ .

Se il nome di un gestore code fornito su chiamate MQCONN o MQCONNX è vuoto, questo campo è ancora impostato sul nome del gestore code a cui è connessa l'applicazione, indipendentemente dal fatto che l'applicazione sia server o client.

Il gestore uscite imposta questo campo all'entrata di ciascuna funzione di uscita.

La lunghezza di questo campo viene fornita da MQ_Q_MGR_NAME_LENGTH.

ExitChainAreaPtr (PMQACH) - input/output
Viene utilizzato per comunicare i dati tra i richiami di diverse uscite in una catena. Viene impostato su un puntatore NULL prima di richiamare la prima funzione (MQ_INIT_EXIT con ExitReason MQXR_CONNECTION) della prima uscita in una catena di uscite. Il valore restituito dall'uscita su una chiamata viene passato alla chiamata successiva.

Fare riferimento a “L'area della catena di uscita e l'intestazione dell'area della catena di uscita (MQACH)” a pagina 1614 per ulteriori informazioni su come utilizzare l'area della catena di uscita.

Hconfig (MQHCONFIG) - input
L'handle di configurazione, che rappresenta la serie di funzioni in fase di inizializzazione. Questo valore viene generato dal gestore code sulla funzione MQ_INIT_EXIT e successivamente viene passato alla funzione di uscita API. Viene impostato all'entrata di ogni funzione di uscita.

È possibile utilizzare Hconfig come puntatore alla struttura MQIEP per effettuare chiamate MQI e DCI. È necessario verificare che i primi 4 byte di HConfig corrispondano al StrucId della struttura MQIEP prima di utilizzare il parametro HConfig come puntatore alla struttura MQIEP.

Funzione (MQLONG) - input
L'identificativo della funzione, i valori validi per cui sono le costanti MQXF_ * descritte in “Costanti esterne” a pagina 1615.

Il gestore di uscite imposta questo campo sul valore corretto, all'ingresso di ciascuna funzione di uscita, a seconda della chiamata API IBM MQ che ha determinato il richiamo dell'uscita.

ExitMsgHandle (MQHMSG) - input/output
Quando Funzione è MQXF_GET e ExitReason è MQXR_AFTER, viene restituito un handle del messaggio valido in questo campo che consente all'uscita API di accedere ai campi del descrittore del messaggio e a tutte le altre proprietà che corrispondono alla stringa ExitProperties specificata nella struttura MQXEPO durante la registrazione dell'uscita API.

Qualsiasi proprietà del descrittore non di messaggi restituita nell'handle ExitMsgnon sarà disponibile da MsgHandle nella struttura MQGMO se ne è stata specificata una o nei dati del messaggio.

Quando Funzione è MQXF_GET e ExitReason è MQXR_BEFORE, se il programma di uscita imposta questo campo su MQHM_NONE, eliminerà il popolamento delle proprietà Handle ExitMsg.

Questo campo non è impostato se la versione è inferiore a MQAXP_VERSION_2.

Come i gestori code elaborano le funzioni di uscita

L'elaborazione eseguita dal gestore code al ritorno da una funzione di uscita dipende da ExitResponse e ExitResponse2.

Tabella 835 a pagina 1611 riepiloga le combinazioni possibili e i relativi effetti per una funzione di uscita MQXR_BEFORE, mostrando:

- Chi imposta i parametri CompCode e Reason della chiamata API
- Se vengono richiamate le funzioni di uscita rimanenti nella catena MQXR_BEFORE e le funzioni di uscita corrispondenti nella catena MQXR_AFTER
- Se viene richiamata la chiamata API

Per una funzione di uscita MQXR_AFTER:

- CompCode e Reason sono impostati nello stesso modo di MQXR_BEFORE
- ExitResponse2 viene ignorato (le rimanenti funzioni di uscita nella catena MQXR_AFTER vengono sempre richiamate)
- MQXCC_SUPPRESS_FUNCTION e MQXCC_SKIP_FUNCTION non sono valide

Per una funzione di uscita MQXR_CONNECTION:

- CompCode e Reason sono impostati nello stesso modo di MQXR_BEFORE
- ExitResponse2 viene ignorato
- MQXCC_SUPPRESS_FUNCTION, MQXCC_SKIP_FUNCTION e MQXCC_SUPPRESS_EXIT non valide

In tutti i casi, in cui un'uscita o il gestore code imposta CompCode e Reason, i valori impostati possono essere modificati da un'uscita richiamata in seguito o dalla chiamata API (se la chiamata API viene richiamata in seguito).

Tabella 835. Elaborazione uscita MQXR_BEFORE			
Valore di ExitResponse	**CompCode e Motivo impostato da**	**Valore della catena ExitResponse2 (continuazione predefinita)**	**Valore dell'API ExitResponse2 (continuazione predefinita)**
MQXCC_OK	uscita	Y	Y
MQXCC_SUPPRESS_EXIT	uscita	Y	Y
MQXCC_SUPPRESS_FUNZIONE	gestore code	N	N
FUNZIONE MQXCC_SKIP	uscita	N	N
MQXCC_NON RIUSCITO	gestore code	N	N

Come i client elaborano le funzioni di uscita

In generale, i client elaborano le funzioni di uscita nello stesso modo delle applicazioni server e l'attributo *QMgrName* in questa struttura si applica se la funzione si trova su un server o su un client.

Tuttavia, il client non ha alcun concetto del file *mqs.ini* , pertanto le stanze *ApiExitCommon* e *APIExitTemplate* non si applicano. Si applica solo la stanza *ApiExitLocal* e questa stanza è configurata nel file *mqclient.ini* .

MQAXC (IBM MQ API exit context structure)

La struttura MQAXC, un blocco di controllo esterno, viene utilizzata come parametro di input per un'uscita API.

MQAXC ha la dichiarazione C seguente:

```
typedef struct tagMQAXC {
   MQCHAR4   StrucId;               /* Structure identifier */
   MQLONG    Version;               /* Structure version number */
   MQLONG    Environment;           /* Environment */
   MQCHAR12  UserId;                /* UserId associated with appl */
   MQBYTE40  SecurityId             /* Extension to UserId running appl */
   MQCHAR264 ConnectionName;        /* Connection name */
   MQLONG    LongMCAUserIdLength;   /* long MCA user identifier length */
   MQLONG    LongRemoteUserIdLength; /* long remote user identifier length */
   MQPTR     LongMCAUserIdPtr;      /* long MCA user identifier address */
   MQPTR     LongRemoteUserIdPtr;   /* long remote user identifier address */
   MQCHAR28  ApplName;              /* Application name */
   MQLONG    ApplType;              /* Application type */
   MQPID     ProcessId;             /* Process identifier */
   MQTID     ThreadId;              /* Thread identifier */

   /* Ver:1 */
   MQCHAR    ChannelName[20]        /* Channel Name */
   MQBYTE4   Reserved1;             /* Reserved */
   PMQCD     pChannelDefinition;    /* Channel Definition pointer */
 };
```

I parametri di MQAXC sono:

StrucId (MQCHAR4) - input
L'identificativo della struttura del contesto di uscita, con un valore di MQAXC_STRUC_ID. Per programmi C, viene definita anche la costante MQAXC_STRUC_ID_ARRAY, con lo stesso valore di MQAXC_STRUC_ID, ma come un array di caratteri invece di una stringa.

Il gestore uscite imposta questo campo all'entrata di ciascuna funzione di uscita.

Versione (MQLONG) - input
Il numero di versione della struttura, con un valore di:

MQAXC_VERSION_2
Numero di versione per la struttura del contesto di uscita.

VERSIONE MQAXC_CURRENT_
Numero di versione corrente per la struttura del contesto di uscita.

Il gestore uscite imposta questo campo all'entrata di ciascuna funzione di uscita.

Ambiente (MQLONG) - input
L'ambiente da cui è stata emessa una chiamata API IBM MQ che ha generato una funzione di uscita guidata. I valori validi per questo campo sono:

MQXE_ALTRO
Questo valore è congruente con i richiami che un'uscita API vede se l'uscita viene richiamata da un'applicazione server. Ciò significa che un'uscita API viene eseguita non modificata su un client e non visualizza nulla di diverso.

Se l'uscita deve realmente determinare se è in esecuzione sul client, l'uscita può farlo esaminando i campi *ChannelName* e *ChannelDefinition* .

MQXE_MCA
Agent del canale messaggi

MQXE_MCA_SVRCONN
Un agente del canale dei messaggi che agisce per conto di un client

SERVER MQXE_COMMAND_
Il server comandi

MQXE_MQSC
Interprete del comando runmqsc

Il gestore uscite imposta questo campo all'entrata di ciascuna funzione di uscita.

UserId (MQCHAR12) - input
L'ID utente associato all'applicazione. In particolare, nel caso di connessioni client, questo campo contiene l'ID utente dell'utente adottato rispetto all'ID utente con cui è in esecuzione il codice del canale. Se un ID utente vuoto passa dal client, non viene apportata alcuna modifica all'ID utente già utilizzato. In altre parole, non viene adottato alcun nuovo ID utente.

Il gestore uscite imposta questo campo all'entrata di ciascuna funzione di uscita. La lunghezza di questo campo è fornita da MQ_USER_ID_LENGTH.

Nel caso di un client, questo è l'ID utente inviato dal client al server. Tenere presente che questo potrebbe non essere l'ID utente effettivo con cui il client è in esecuzione nel gestore code, poiché potrebbe essere presente una configurazione MCAUser o CHLAUTH che modifica l'ID utente.

SecurityId (MQBYTE40) - input
Un'estensione dell'ID utente che esegue l'applicazione. La sua lunghezza è fornita da MQ_SECURITY_ID_LENGTH.

Nel caso di un client, questo è l'ID utente inviato dal client al server. Tenere presente che questo potrebbe non essere l'ID utente effettivo con cui il client è in esecuzione nel gestore code, poiché potrebbe essere presente una configurazione MCAUser o CHLAUTH che modifica l'ID utente.

ConnectionName (MQCHAR264) - input
Il campo del nome connessione, impostato sull'indirizzo del client. Ad esempio, per TCP/IP, è l'indirizzo IP del client.

La lunghezza di questo campo è fornita da MQ_CONN_NAME_LENGTH.

Nel caso di un client, questo è l'indirizzo partner del gestore code.

LongMCAUserIdLength (MQLONG) - input
La lunghezza dell'identificativo utente MCA lungo.

Quando MCA si connette al gestore code, questo campo è impostato sulla lunghezza dell'identificativo utente MCA lungo (o zero se non esiste tale identificativo).

Nel caso di un client, questo è l'identificativo utente lungo del client.

LongRemoteUserIdLength (MQLONG) - input
La lunghezza dell'identificativo utente remoto lungo.

Quando MCA si connette al gestore code, questo campo è impostato sulla lunghezza dell'identificativo utente remoto lungo. Altrimenti questo campo verrà impostato su zero

Nel caso di un cliente, impostare questo campo su zero.

LongMCAUserIdPtr (MQPTR) - input
Indirizzo dell'identificativo utente MCA lungo.

Quando MCA si connette al gestore code, questo campo è impostato sull'indirizzo dell'identificativo utente MCA lungo (o su un puntatore null se non esiste tale identificativo).

Nel caso di un client, questo è l'identificativo utente lungo del client.

LongRemoteUserIdPtr (MQPTR) - input
L'indirizzo dell'identificativo utente remoto lungo.

Quando MCA si connette al gestore code, questo campo viene impostato sull'indirizzo dell'identificativo utente remoto lungo (o su un puntatore null se non esiste tale identificativo).

Nel caso di un cliente, impostare questo campo su zero.

ApplName (MQCHAR28) - input
Il nome dell'applicazione o del componente che ha emesso la chiamata API IBM MQ .

Le regole per la generazione di ApplName sono le stesse per la generazione del nome predefinito per un MQPUT.

Il valore di questo campo viene trovato interrogando il sistema operativo per il nome del programma. La sua lunghezza è fornita da MQ_APPL_NAME_LENGTH.

ApplType (MQLONG) - input
Il tipo di applicazione o componente che ha emesso la chiamata API IBM MQ .

Il valore è MQAT_DEFAULT per la piattaforma su cui l'applicazione è compilata oppure equivale a uno dei valori MQAT_ * definiti.

Il gestore uscite imposta questo campo all'entrata di ciascuna funzione di uscita.

ProcessId (MQPID) - input
L'identificativo del processo del sistema operativo.

Dove applicabile, il gestore di uscita imposta questo campo all'entrata per ogni funzione di uscita.

ThreadId (MQTID) - input
L'identificativo del thread MQ . Questo è lo stesso identificativo utilizzato nella traccia di MQ e nei dump FFST , ma potrebbe essere diverso dall'identificativo del thread del sistema operativo.

Dove applicabile, il gestore di uscita imposta questo campo all'entrata per ogni funzione di uscita.

ChannelName (MQCHAR) - input
Il nome del canale, riempito con spazi vuoti, se applicabile e noto.

Se non applicabile, questo campo è impostato su caratteri NULL.

Reserved1 (MQBYTE4) - input
Questo campo è riservato.

ChanneDefinition (PMQCD) - input
Un puntatore alla definizione di canale utilizzata, se applicabile e nota.

Se non applicabile, questo campo è impostato su caratteri NULL.

Tenere presente che il puntatore viene completato solo se la connessione è in elaborazione per conto di un canale IBM MQ e che la definizione del canale è stata letta.

In particolare, la definizione del canale non viene data sul server quando viene effettuata la prima chiamata MQCONN per il canale. Inoltre, se il puntatore è pieno, la struttura (e qualsiasi struttura

secondaria) indicata dal puntatore deve essere trattata come di sola lettura; qualsiasi aggiornamento della struttura porterebbe a risultati imprevedibili e non è supportato.

Nel caso di un client, i campi diversi da quelli con un valore specificato per un client contengono valori appropriati per un'applicazione client.

L'area della catena di uscita e l'intestazione dell'area della catena di uscita (MQACH)

Se necessario, una funzione di uscita può acquisire memoria per un'area della catena di uscite e impostare ExitChainAreaPtr in MQAXP per puntare a questa memoria.

Le uscite (le stesse o diverse funzioni di uscita) possono acquisire più aree della catena di uscita e collegarle tra loro. Le aree della catena di uscita devono essere aggiunte o rimosse solo da questo elenco mentre vengono richiamate dal gestore di uscita. Ciò garantisce che non vi siano problemi di serializzazione causati da thread differenti che aggiungono o rimuovono aree dall'elenco contemporaneamente.

Un'area della catena di uscite deve iniziare con una struttura di intestazione MQACH, la cui dichiarazione C è:

```
typedef struct tagMQACH {
   MQCHAR4   StrucId;             /* Structure identifier */
   MQLONG    Version;             /* Structure version number */
   MQLONG    StrucLength;         /* Length of the MQACH structure */
   MQLONG    ChainAreaLength;     /* Exit chain area length */
   MQCHAR48  ExitInfoName         /* Exit information name */
   PMQACH    NextChainAreaPtr;    /* Pointer to next exit chain area */
 };
```

I campi nell'intestazione dell'area della catena di uscita sono:

StrucId (MQCHAR4) - input
L'identificativo della struttura dell'area della catena di uscita, con un valore iniziale, definito da MQACH_DEFAULT, di MQACH_STRUC_ID.

Per i programmi C, viene definita anche la costante MQACH_STRUC_ID_ARRAY, che ha lo stesso valore di MQACH_STRUC_ID, ma come un array di caratteri invece di una stringa.

Versione (MQLONG) - input
Il numero di versione della struttura, come segue:

MQACH_VERSION_1
Il numero di versione per la struttura del parametro di uscita.

VERSIONE MQACH_CURRENT_
Il numero di versione corrente per la struttura del contesto di uscita.

Il valore iniziale di questo campo, definito da MQACH_DEFAULT, è MQACH_CURRENT_VERSION.

Nota: Se si introduce una nuova versione di questa struttura, il layout della parte esistente non cambia. Le funzioni di uscita devono controllare che il numero di versione sia uguale o superiore alla versione più bassa contenente i campi che la funzione di uscita deve utilizzare.

StrucLength (MQLONG) - input
La lunghezza della struttura MQACH. Le uscite possono utilizzare questo campo per determinare l'avvio dei dati di uscita, impostandolo sulla lunghezza della struttura creata dall'uscita.

Il valore iniziale di questo campo, definito da MQACH_DEFAULT, è MQACH_CURRENT_LENGTH.

Lunghezza ChainArea(MQLONG) - input
La lunghezza dell'area della catena di uscita, impostata sulla lunghezza complessiva dell'area della catena di uscita corrente, inclusa l'intestazione MQACH.

Il valore iniziale di questo campo, definito da MQACH_DEFAULT, è zero.

Nome ExitInfo(MQCHAR48) - input
Il nome delle informazioni di uscita.

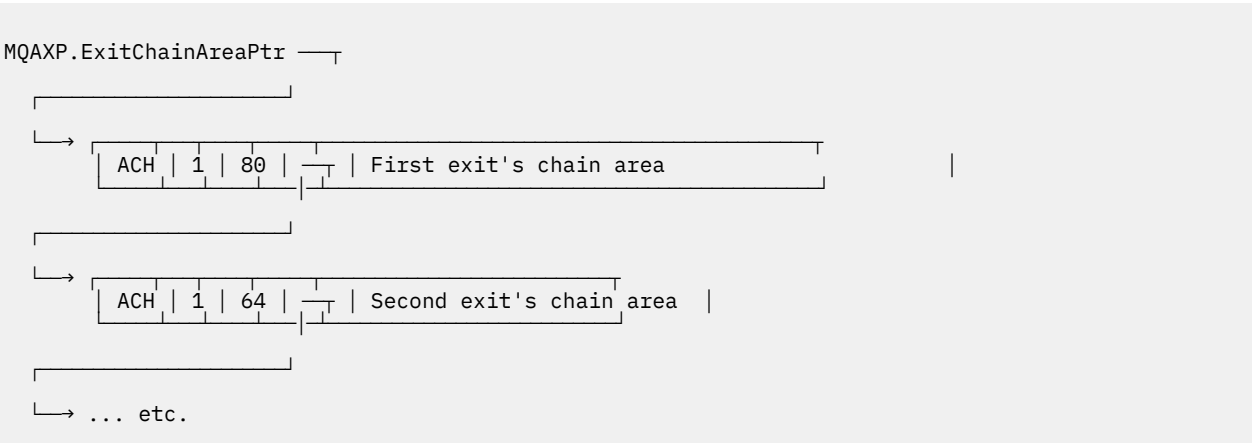
Quando un'uscita crea una struttura MQACH, deve inizializzare questo campo con il proprio nome ExitInfo, in modo che successivamente questa struttura MQACH possa essere trovata da un'altra istanza di questa uscita o da un'uscita collaborativa.

Il valore iniziale di questo campo, definito da MQACH_DEFAULT, è una stringa di lunghezza zero ({""}).

NextChainAreaPtr (PMQACH) - input
Un puntatore all'area della catena di uscita successiva con un valore iniziale, definito da MQACH_DEFAULT, di puntatore null (NULL).

Le funzioni di uscita devono rilasciare la memoria per tutte le aree della catena di uscita che acquisiscono e manipolare i puntatori della catena per rimuovere le aree della catena di uscita dall'elenco.

Un'area della catena di uscita può essere costruita come segue:



Costanti esterne

Utilizzare questo argomento come informazioni di riferimento per le costanti esterne disponibili per l'API.

Le seguenti costanti esterne sono disponibili per le uscite API:

MQXF_ * (identificativi funzione di uscita)

```
MQXF_INIT                       1       X'00000001'
MQXF_TERM                       2       X'00000002'
MQXF_CONN                       3       X'00000003'
MQXF_CONNX                      4       X'00000004'
MQXF_DISC                       5       X'00000005'
MQXF_OPEN                       6       X'00000006'
MQXF_CLOSE                      7       X'00000007'
MQXF_PUT1                       8       X'00000008'
MQXF_PUT                        9       X'00000009'
MQXF_GET                        10      X'0000000A'
MQXF_DATA_CONV_ON_GET           11      X'0000000B'
MQXF_INQ                        12      X'0000000C'
MQXF_SET                        13      X'0000000D'
MQXF_BEGIN                      14      X'0000000E'
MQXF_CMIT                       15      X'0000000F'
MQXF_BACK                       16      X'00000010'
MQXF_STAT                       18      X'00000012'
MQXF_CB                         19      X'00000013'
MQXF_CTL                        20      X'00000014'
MQXF_CALLBACK                   21      X'00000015'
MQXF_SUB                        22      X'00000016'
MQXF_SUBRQ                      23      X'00000017'
MQXF_XACLOSE                    24      X'00000018'
MQXF_XACOMMIT                   25      X'00000019'
MQXF_XACOMPLETE                 26      X'0000001A'
MQXF_XAEND                      27      X'0000001B'
MQXF_XAFORGET                   28      X'0000001C'
MQXF_XAOPEN                     29      X'0000001D'
MQXF_XAPREPARE                  30      X'0000001E'
MQXF_XARECOVER                  31      X'0000001F'
```

```
        MQXF_XAROLLBACK                 32      X'00000020'
        MQXF_XASTART                    33      X'00000021'
        MQXF_AXREG                      34      X'00000022'
        MQXF_AXUNREG                    35      X'00000023'
```

MQXR_ * (motivi uscita)

```
        MQXR_BEFORE                     1       X'00000001'
        MQXR_AFTER                      2       X'00000002'
        MQXR_CONNECTION                 3       X'00000003'
```

MQXE_ * (ambienti)

```
        MQXE_OTHER                      0       X'00000000'
        MQXE_MCA                        1       X'00000001'
        MQXE_MCA_SVRCONN                2       X'00000002'
        MQXE_COMMAND_SERVER             3       X'00000003'
        MQXE_MQSC                       4       X'00000004'
```

MQ* _ * (costanti aggiuntive)

```
        MQAXP_VERSION_1                 1
        MQAXP_VERSION_2                 2
        MQAXC_VERSION_1                 1
        MQACH_VERSION_1                 1

        MQAXP_CURRENT_VERSION           1
        MQAXC_CURRENT_VERSION           1
        MQACH_CURRENT_VERSION           1

        MQXACT_EXTERNAL                 1
        MQXACT_INTERNAL                 2

        MQXT_API_EXIT                   2

        MQACH_LENGTH_1                  68 (32-bit platforms)
                                        72 (64-bit platforms)
                                        80 (128-bit platforms)

        MQACH_CURRENT_LENGTH            68 (32-bit platforms)
                                        72 (64-bit platforms)
                                        80 (128-bit platforms)
```

MQ* _ * (costanti null)

```
        MQXPDA_NONE                     X'00...00' (48 nulls)
        MQXPDA_NONE_ARRAY               '\0','\0',...,'\0','\0'
```

MQXCC_ * (codici di completamento)

```
       MQXCC_FAILED                    -8
```

MQRC_ * (codici motivo)

MQRC_API_EXIT_ERROR 2374 X'00000946'
Un richiamo della funzione di uscita ha restituito un codice di risposta non valido o ha avuto esito negativo in qualche modo e il gestore code non può determinare l'azione successiva da intraprendere.

Esaminare entrambi i campi ExitResponse e ExitResponse2 di MQAXP per determinare il codice di risposta non valido e modificare l'uscita per restituire un codice di risposta valido.

MQRC_API_EXIT_INIT_ERROR 2375 X'00000947'
Il gestore code ha rilevato un errore durante l'inizializzazione dell'ambiente di esecuzione per una funzione di uscita API.

MQRC_API_EXIT_TERM_ERROR 2376 X'00000948'
Il gestore code ha rilevato un errore durante la chiusura dell'ambiente di esecuzione per una funzione di uscita API.

MQRC_EXIT_REASON_ERROR 2377 X'00000949'
Il valore del campo ExitReason fornito in una chiamata MQXEP (exit entry point registration call) è in errore.

Esaminare il valore del campo ExitReason per determinare e correggere il valore del motivo di uscita errato.

MQRC_RESERVED_VALUE_ERROR 2378 X'0000094A'
Il valore del campo Riservato è errato.

Esaminare il valore del campo Riservato per determinare e correggere il valore Riservato.

Typedef linguaggio C

Questo argomento fornisce informazioni sulle typedef associate alle uscite API disponibili in linguaggio C.

Di seguito sono riportate le typedef di linguaggio C associate alle uscite API:

```
typedef PMQLONG  MQPOINTER PPMQLONG;
typedef PMQBYTE  MQPOINTER PPMQBYTE;
typedef PMQHOBJ  MQPOINTER PPMQHOBJ;
typedef PMQOD    MQPOINTER PPMQOD;
typedef PMQMD    MQPOINTER PPMQMD;
typedef PMQPMO   MQPOINTER PPMQPMO;
typedef PMQGMO   MQPOINTER PPMQGMO;
typedef PMQCNO   MQPOINTER PPMQCNO;
typedef PMQBO    MQPOINTER PPMQBO;

typedef MQAXP    MQPOINTER PMQAXP;
typedef MQACH    MQPOINTER PMQACH;
typedef MQAXC    MQPOINTER PMQAXC;

typedef MQCHAR   MQCHAR16[16];
typedef MQCHAR16 MQPOINTER PMQCHAR16;

typedef MQLONG   MQPID;
typedef MQLONG   MQTID;
```

La chiamata di registrazione del punto di ingresso di uscita (MQXEP)

Utilizzare queste informazioni per informazioni su MQXEP, sul richiamo del linguaggio C MQXEP e sul prototipo della funzione C MQXEP.

Utilizzare la chiamata MQXEP per:

1. Registrare i punti di richiamo dell'uscita API prima e dopo IBM MQ in cui richiamare le funzioni di uscita
2. Specificare i punti di ingresso della funzione di uscita
3. Annulla registrazione dei punti di ingresso della funzione di uscita

Generalmente, si codificano le chiamate MQXEP nella funzione di uscita MQ_INIT_EXIT, ma è possibile specificarle in qualsiasi funzione di uscita successiva.

Se si utilizza una chiamata MQXEP per registrare una funzione di uscita già registrata, la seconda chiamata MQXEP viene completata correttamente, sostituendo la funzione di uscita registrata.

Se si utilizza una chiamata MQXEP per registrare una funzione di uscita NULL, la chiamata MQXEP viene completata correttamente e la funzione di uscita viene annullata.

Se le chiamate MQXEP vengono utilizzate per registrare, annullare la registrazione e registrare nuovamente una particolare funzione di uscita durante la durata di una richiesta di connessione, la funzione di uscita precedentemente registrata viene riattivata. Qualsiasi memoria ancora assegnata e associata a questa istanza della funzione di uscita è disponibile per l'utilizzo da parte delle funzioni

dell'uscita. (Questa memoria viene generalmente rilasciata durante il richiamo della funzione di uscita di terminazione.)

L'interfaccia di MQXEP è:

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason)
```

dove:

Hconfig (MQHCONFIG) - input
L'handle di configurazione, che rappresenta l'uscita API che include la serie di funzioni in fase di inizializzazione. Questo valore viene generato dal gestore code immediatamente, prima di richiamare la funzione MQ_INIT_EXIT e viene passato in MQAXP a ogni funzione di uscita API.

ExitReason (MQLONG) - input
Il motivo per cui il punto di ingresso è stato registrato, tra i seguenti motivi:

- Inizializzazione o terminazione del livello di connessione (MQXR_CONNECTION)
- Prima di una chiamata API IBM MQ API (MQXR_BEFORE)
- Dopo una chiamata API IBM MQ (MQXR_AFTER)

Funzione (MQLONG) - input
L'identificativo della funzione, i valori validi per cui sono le costanti MQXF_ * (consultare “Costanti esterne” a pagina 1615).

EntryPoint (PMQFUNC) - input
L'indirizzo del punto di entrata per la funzione di uscita da registrare. Il valore NULL indica che la funzione di uscita non è stata fornita o che è in corso l'annullamento della registrazione di una registrazione precedente della funzione di uscita.

ExitOpts(MQXEPO)
Le uscite API possono specificare le opzioni che controllano la modalità di registrazione delle uscite API. Se viene specificato un puntatore null per questo campo, vengono assunti i valori predefiniti della struttura MQXEPO.

CompCode (MQLONG) - output
Il codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo (MQLONG) - output
Il codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED:

ERRORE MQRC_HCONFIG
(2280, X'8E8') L'handle di configurazione fornito non è valido. Utilizzare l'handle di configurazione da MQAXP.

ERRORE REASON_MQRC_EXIT_
(2377, X' 949 ') Il motivo del richiamo della funzione di uscita fornito non è valido o non è valido per l'identificativo della funzione di uscita fornito.

Utilizzare uno dei motivi di richiamo della funzione di uscita validi (valore MQXR_ *) oppure utilizzare una combinazione di identificativo della funzione e motivo dell'uscita valida. (Consultare Tabella 836 a pagina 1619.)

ERRORE MQRC_FUNCTION_
(2281, X'8E9') L'identificativo della funzione fornito non è valido per il motivo dell'uscita API. La seguente tabella mostra combinazioni valide di identificativi di funzione e ExitReasons.

Tabella 836. Combinazioni valide di identificativi funzione e ExitReasons

Funzione	ExitReason
INIT MQXF TERM_MQXF	MQXR_CONNECZIONE
CONN MQXF MQXF_CONNX DISC MQXF MQXF_OPEN CLOSE MQXF MQXF_PUT1 MQXF_PUT GET MQXF INQ MQXF SET MQXF BEGIN MQXF CMIT_MQXF MQXF_BACK STAT_MQXF CB MQXF CTL MQXF CALLBACK MQXF SUB MQXF MQXF_SUBRQ	MQXR_BEFORE MQXR_XX_ENCODE_CASE_ONE dopo
CONV_DATA_MQXF_GET	MQXR_BEFORE

PROBLEMA_RISORSA_MQRC_
(2102, X'836 ') Un tentativo di registrare o annullare la registrazione di una funzione di uscita ha avuto esito negativo a causa di un problema di risorsa.

ERRORE MQRC_UNEXPECTED_
(2195, X'893 ') Un tentativo di registrare o annullare la registrazione di una funzione di uscita ha avuto esito negativo in modo imprevisto.

ERRORE MQRC_PROPERTY_NAME_
(2442, X'098A') Nome ExitProperties non valido.

ERRORE MQRC_XEPO_
(2507, X'09CB') Struttura delle opzioni di chiusura non valida.

Richiamo linguaggio C MQXEP

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason);
```

Dichiarazione per l'elenco di parametri:

```
    MQHCONFIG       Hconfig;        /* Configuration handle */
    MQLONG          ExitReason;     /* Exit reason */
    MQLONG          Function;       /* Function identifier */
    PMQFUNC         EntryPoint;     /* Function entry point */
    MQXEPO          ExitOpts;       /* Options that control the action of MQXEP */
    MQLONG          CompCode;       /* Completion code */
```

```
    MQLONG          Reason;         /* Reason code qualifying completion
                                       code */
```

Prototipo della funzione MQXEP C

```
void MQXEP (
MQHCONFIG    Hconfig,        /* Configuration handle */
MQLONG       ExitReason,     /* Exit reason */
MQLONG       Function,       /* Function identifier */
PMQFUNC      EntryPoint,     /* Function entry point */
PMQXEPO      pExitOpts;      /* Options that control the action of MQXEP */
PMQLONG      pCompCode,      /* Address of completion code */
PMQLONG      pReason);       /* Address of reason code qualifying completion
                                code */
```

Funzioni di uscita

Questa sezione fornisce alcune informazioni generali che consentono di utilizzare le chiamate di funzione e descrive come richiamare le funzioni di uscita individuali.

Utilizzare queste informazioni per comprendere le regole generali per le routine di uscita API e per impostare e ripulire l'ambiente di esecuzione di uscita.

Regole generali per le routine di uscita API

Le seguenti regole generali si applicano quando si richiamano le routine di uscita API:

- In tutti i casi, le funzioni di uscita API vengono guidate prima della convalida dei parametri di chiamata API e prima dei controlli di sicurezza (nel caso di MQCONN, MQCONNX o MQOPEN).
- I valori dei campi immessi e emessi da una routine di uscita sono:
  - All'immissione in una funzione di uscita API *prima di* IBM MQ , il valore di un campo può essere impostato dal programma applicativo o da un precedente richiamo della funzione di uscita.
  - In caso di output da una funzione di uscita API *prima di* IBM MQ , il valore di un campo può essere lasciato invariato o impostato su un altro valore dalla funzione di uscita.
  - All'input di una funzione API exit *after* IBM MQ , il valore di un campo può essere il valore impostato dal gestore code dopo l'elaborazione della chiamata API IBM MQ o può essere impostato su un valore da un precedente richiamo della funzione exit nella catena di funzioni exit.
  - In caso di output da una funzione di uscita chiamata API *dopo* IBM MQ , il valore di un campo può essere lasciato invariato o impostato su un altro valore dalla funzione di uscita.
- Le funzioni di uscita devono comunicare con il gestore code utilizzando i campi ExitResponse e ExitResponse2 .
- I campi CompCode e Reason code comunicano nuovamente all'applicazione. Le funzioni di uscita e del gestore code possono impostare i campi CompCode e Codice motivo.
- La chiamata MQXEP restituisce nuovi codici motivo alle funzioni di uscita che richiamano MQXEP. Tuttavia, le funzioni di uscita possono convertire questi nuovi codici di errore in tutti i codici di errore esistenti che le applicazioni nuove ed esistenti possono comprendere.
- Ogni prototipo di funzione di uscita ha parametri simili alla funzione API con un ulteriore livello di riferimento, ad eccezione di CompCode e Reason.
- Le uscite API possono emettere chiamate MQI (tranne MQDISC), ma queste chiamate MQI non richiamano esse stesse le uscite API.

Notare che, se l'applicazione si trova su un server o su un client, non è possibile prevedere la sequenza delle chiamate di uscita API. Una chiamata BEFORE dell'uscita API potrebbe non essere seguita immediatamente da una chiamata AFTER .

La chiamata BEFORE può essere seguita da un'altra chiamata BEFORE . Ad esempio:

PRIMA di MQCTL

PRIMA del callback
PRIMA di MQPUT
DOPO MQPUT
DOPO il callback
DOPO MQCTL

o

PRIMA DI XAOPEN
PRIMA di MQCONNX
DOPO MQCONNX
DOPO XAOPEN

Sul client, esiste un'uscita che può modificare il comportamento della chiamata MQCONN o MQCONNX, denominata `PreConnect` exit. L'uscita `PreConnect` può modificare uno qualsiasi dei parametri sulla chiamata MQCONN o MQCONNX incluso il nome del gestore code. Il client richiama prima questa uscita e poi la chiamata MQCONN o MQCONNX. Notare che solo la chiamata MQCONN o MQCONNX iniziale richiama l'uscita API; le successive chiamate di riconnessione non hanno alcun effetto.

L'ambiente di esecuzione

In generale, tutti gli errori delle funzioni di uscita vengono comunicati al gestore di uscita utilizzando i campi ExitResponse e ExitResponse2 in MQAXP.

Questi errori a loro volta vengono convertiti in valori MQCC_ * e MQRC_ * e comunicati all'applicazione nei campi CompCode e Reason. Tuttavia, eventuali errori riscontrati nella logica del gestore di uscita vengono comunicati all'applicazione come valori MQCC_ * e MQRC_ * nei campi CompCode e Reason.

Se una funzione MQ_TERM_EXIT restituisce un errore:

- La chiamata MQDISC è già stata eseguita
- Non c'è altra opportunità di guidare la funzione di uscita *dopo* MQ_TERM_EXIT (e quindi eseguire la ripulitura dell'ambiente di esecuzione dell'uscita)
- La ripulitura dell'ambiente di esecuzione di uscita non viene eseguita

L'uscita non può essere scaricata perché potrebbe essere ancora in uso. Inoltre, le altre uscite registrate più in basso nella catena di uscita per cui l'uscita *prima* ha avuto esito positivo, verranno guidate nell'ordine inverso.

Impostazione dell'ambiente di esecuzione di uscita

Durante l'elaborazione di una chiamata MQCONN o MQCONNX esplicita, la logica di gestione delle uscite imposta l'ambiente di esecuzione delle uscite prima di richiamare la funzione di inizializzazione delle uscite (MQ_INIT_EXIT). L'impostazione dell'ambiente di esecuzione dell'uscita implica il caricamento dell'uscita, l'acquisizione della memoria e l'inizializzazione delle strutture dei parametri di uscita. Viene assegnato anche l'handle di configurazione di uscita.

Se si verificano degli errori durante questa fase, la chiamata MQCONN o MQCONNX ha esito negativo con CompCode MQCC_FAILED e uno dei seguenti codici di errore:

ERRORE USCITA MQRC_API
Un tentativo di caricare un modulo di uscita API non è riuscito.

MQRC_API_EXIT_NOT_FOUND
Non è stato possibile trovare una funzione API exit nel modulo API exit.

MQRC_STORAGE_NON_DISPONIBILE
Un tentativo di inizializzare l'ambiente di esecuzione per una funzione di uscita API non è riuscito perché non era disponibile memoria sufficiente.

MQRC_API_EXIT_INIT_ERROR
Si è verificato un errore durante l'inizializzazione dell'ambiente di esecuzione per una funzione di uscita API.

Ripulitura dell'ambiente di esecuzione di uscita

Durante l'elaborazione di una chiamata MQDISC esplicita o di una richiesta di disconnessione implicita come risultato della chiusura di un'applicazione, la logica di gestione dell'uscita potrebbe dover ripulire l'ambiente di esecuzione dell'uscita dopo aver richiamato la funzione di terminazione dell'uscita (MQ_TERM_EXIT), se registrata.

La ripulitura dell'ambiente di esecuzione dell'uscita implica il rilascio della memoria per le strutture dei parametri di uscita, possibilmente eliminando tutti i moduli precedentemente caricati in memoria.

Se si verificano errori durante questa fase, una chiamata MQDISC esplicita non riesce con CompCode MQCC_FAILED e il seguente codice motivo (gli errori non vengono evidenziati nelle richieste di disconnessione implicite):

ERRORE USCITA MQRC_API
Si è verificato un errore durante la chiusura dell'ambiente di esecuzione per una funzione di uscita API. L'uscita non deve restituire alcun errore da MQDISC prima o dopo le chiamate della funzione di uscita API MQ_TERM*.

Uscite API sui client

Un client utilizza l'uscita PreConnect per modificare il comportamento delle chiamate MQCONN e MQCONNX e non supporta le proprietà dell'uscita API.

Uscita PreConnect

Su un client, l'exit PreConnect può essere utilizzata per ricercare la definizione del canale da un repository centrale, ad esempio un server LDAP.

L'uscita PreConnect può anche modificare qualsiasi parametro, o tutti i parametri, su una chiamata MQCONN o MQCONNX stessa, ad esempio, il nome del gestore code.

Nel caso di applicazioni client, l'uscita PreConnect deve essere richiamata prima dell'uscita API perché l'uscita API MQCONN o MQCONNX viene richiamata solo una volta che il nome del gestore code è noto e questo nome può essere modificato dall'uscita PreConnect .

Si noti che solo la chiamata MQCONN o MQCONNX iniziale richiama l'exit.

Proprietà uscita API

Su un server, le uscite API possono registrare una struttura MQXEPO al momento dell'inizializzazione. La struttura MQXEPO contiene il campo ExitProperties che descrive il gruppo di proprietà a cui è interessata l'uscita. Ciò ha l'effetto di generare un handle della proprietà del messaggio separato che l'uscita può manipolare separatamente da qualsiasi handle della proprietà del messaggio dell'applicazione.

Su un client, le proprietà dell'uscita API non sono supportate. Se viene effettuato un tentativo di registrazione di un nome gruppo di proprietà su un client, la funzione ha esito negativo con un codice motivo MQRC_EXIT_PROPS_NOT_SUPPORTED.

Backout - MQ_BACK_EXIT

MQ_BACK_EXIT fornisce una funzione di uscita di backout per eseguire *prima* e *dopo* l'elaborazione di backout. Utilizzare l'identificativo funzione MQXF_BACK con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni di uscita chiamata di backout.

L'interfaccia per questa funzione è:

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
MQAXP    ExitParms;      /* Exit parameter structure */
MQAXC    ExitContext;    /* Exit context structure */
MQHCONN  Hconn;          /* Connection handle */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying completion code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_BACK_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

Inizio - MQ_BEGIN_EXIT

MQ_BEGIN_EXIT fornisce una funzione di inizio uscita per eseguire *prima* e *dopo* l'elaborazione delle chiamate MQBEGIN. Utilizzare l'identificatore funzione MQXF_BEGIN con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni di uscita chiamata MQBEGIN.

L'interfaccia per questa funzione è:

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pBeginOptions (PMQBO) - input/output
Puntatore alle opzioni iniziali.

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
MQAXP    ExitParms;      /* Exit parameter structure */
MQAXC    ExitContext;    /* Exit context structure */
MQHCONN  Hconn;          /* Connection handle */
PMQBO    pBeginOptions;  /* Ptr to begin options */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying completion code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_BEGIN_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PPMQBO    ppBeginOptions,   /* Address of ptr to begin options */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

Callback - MQ_CALLBACK_EXIT

MQ_CALLBACK_EXIT fornisce una funzione di uscita per eseguire *prima* e *dopo* l'elaborazione del callback. Utilizzare l'ID funzione MQXF_CALLBACK con i motivi dell'uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni di uscita della chiamata di callback.

L'interfaccia per questa funzione è:

```
MQ_CALLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts,
                  &pBuffer, &pMQCBContext)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura parametro di uscita

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita

Hconn (MQHCONN) - input/output
ID interno connessioni

pMsgDesc
Descrittore messaggio

pGetMsgOpts
Opzioni che controllano l'azione di MQGET

pBuffer
Area per contenere i dati del messaggio

pMQCBContext
Dati di contesto per il callback

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
        MQAXP    ExitParms;      /* Exit parameter structure */
        MQAXC    ExitContext;    /* Exit context structure */
        MQHCONN  Hconn;          /* Connection handle */
        PMQMD    pMsgDesc;       /* Message descriptor */
        PMQGMO   pGetMsgOpts;    /* Options that define the operation of the consumer */
        PMQVOID  pBuffer;        /* Area to contain the message data */
        PMQCBC   pContext;       /* Context data for the callback */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts, &pBuffer,
               &pContext);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_CALLBACK_EXIT (
PMQAXP    pExitParms;    /* Exit parameter structure */
PMQAXC    pExitContext;  /* Exit context structure */
PMQHCONN  pHconn;        /* Connection handle */
PPMQMD    ppMsgDesc;     /* Message descriptor */
PPMQGMO   ppGetMsgOpts;  /* Options that define the operation of the consumer */
PPMQVOID  ppBuffer;      /* Area to contain the message data */
PPMQCBC   ppContext;)    /* Context data for the callback */
```

Note d'utilizzo

1. L'uscita di callback viene richiamata prima del richiamo del consumer e dopo il completamento della funzione consumer del consumer. Anche se le strutture MQMD e MQGMO sono modificabili, la modifica dei valori nell'uscita precedente non riguida il richiamo di un messaggio dalla coda poiché il messaggio è già stato rimosso dalla coda per essere consegnato alla funzione consumer

Gestisci funzioni di callback - MQ_CB_EXIT

MQ_CB_EXIT fornisce una funzione di uscita per eseguire *prima* e *dopo* la chiamata MQCB. Utilizzare l'identificatore funzione MQXF_CB con i motivi dell'uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni dell'uscita chiamata MQCB.

L'interfaccia per questa funzione è:

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &pCallbackDesc,
            &Hobj, &pMsgDesc, &pGetMsgOpts, &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura parametro di uscita

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita

Hconn (MQHCONN) - input/output
ID interno connessioni

Operazione (MQLONG) - input/output
Valore dell'operazione

pCallbackDesc (PMQCBD) - input/output
Descrittore callback

Hobj (MQHOBJ) - input/output
Handle di oggetti

pMsgDesc (PMQMD) - input/output
Descrittore messaggio

pGetMsgOpts (PMQGMO) - input/output
Opzioni che controllano l'azione di MQCB

CompCode (MQLONG) - input/output
Codice di completamento

Motivo (MQLONG) - input/output
Codice di errore qualificante CompCode

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
MQAXP    ExitParms;      /* Exit parameter structure */
MQAXC    ExitContext;    /* Exit context structure */
MQHCONN  Hconn;          /* Connection handle */
MQLONG   Operation;      /* Operation value. */
MQCBD    pMsgDesc;       /* Callback descriptor. */
MQHOBJ   Hobj;           /* Object handle. */
PMQMD    pMsgDesc;       /* Message descriptor */
PMQGMO   pGetMsgOpts;    /* Options that define the operation of the consumer */
PMQLONG  CompCode;       /* Completion code. */
PMQLONG  Reason;         /* Reason code qualifying CompCode. */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &Hobj, &pMsgDesc,
            &pGetMsgOpts, &CompCode, &Reason);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_CB_EXIT (
PMQAXP     pExitParms;     /* Exit parameter structure */
PMQAXC     pExitContext;   /* Exit context structure */
PMQHCONN   pHconn;         /* Connection handle */
```

```
PMQLONG    pOperation;     /* Callback operation */
PMQHOBJ    pHobj;          /* Object handle */
PPMQMD     ppMsgDesc;      /* Message descriptor */
PPMQGMO    ppGetMsgOpts;   /* Options that control the action of MQCB */
PMQLONG    pCompCode;      /* Completion code */
PMQLONG    pReason;        /* Reason code qualifying CompCode */
```

Chiudi - MQ_CLOSE_EXIT

MQ_CLOSE_EXIT fornisce una funzione di chiusura dell'uscita per eseguire *prima* e *dopo* l'elaborazione della chiamata MQCLOSE. Utilizzare l'identificativo della funzione MQXF_CLOSE con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni di uscita chiamata MQCLOSE.

L'interfaccia per questa funzione è:

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHobj,
               &Options, &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pHobj (PMQHOBJ) - input
Puntatore alla gestione oggetto.

Opzioni (MQLONG) - input/output
Opzioni di chiusura.

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
MQHCONN         Hconn;          /* Connection handle */
PMQHOBJ         pHobj;          /* Ptr to object handle */
MQLONG          Options;        /* Close options */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext,&Hconn, &pHobj, &Options,
               &CompCode, &Reason);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_CLOSE_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQHCONN        pHconn,         /* Address of connection handle */
PPMQHOBJ        ppHobj,         /* Address of ptr to object handle */
PMQLONG         pOptions,       /* Address of close options */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Commit - MQ_CMIT_EXIT

MQ_CMIT_EXIT fornisce una funzione di uscita commit per eseguire *prima* e *dopo* l'elaborazione del commit. Utilizzare l'ID funzione MQXF_CMIT con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni di uscita della chiamata di commit.

Se un'operazione di commit ha esito negativo e viene eseguito il backout della transazione, la chiamata MQCMIT ha esito negativo con MQCC_WARNING e MQRC_BACKED_OUT. Questi codici di ritorno e di motivo vengono passati in qualsiasi *dopo* le funzioni di uscita MQCMIT per fornire alle funzioni di uscita un'indicazione che è stato eseguito il backout dell'unità di lavoro.

L'interfaccia per questa funzione è:

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
        MQAXP    ExitParms;      /* Exit parameter structure */
        MQAXC    ExitContext;    /* Exit context structure */
        MQHCONN  Hconn;          /* Connection handle */
        MQLONG   CompCode;       /* Completion code */
        MQLONG   Reason;         /* Reason code qualifying completion code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext,&Hconn, &CompCode, &Reason);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_CMIT_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

Note d'utilizzo

1. L'interfaccia della funzione MQ_GET_EXIT qui descritta viene utilizzata sia per la funzione di uscita MQXF_GET che per quella di uscita “CONV_DATA_MQXF_GET” a pagina 1635 .

   Per queste due funzioni di uscita sono definiti punti di ingresso separati, per cui per intercettare *entrambi* la chiamata MQXEP deve essere utilizzata due volte; per questa chiamata utilizzare l'identificativo funzione MQXF_GET.

   Poiché l'interfaccia MQ_GET_EXIT è la stessa per MQXF_GET e MQXF_DATA_CONV_ON_GET, è possibile utilizzare una singola funzione di uscita per entrambi; il campo *Function* nella struttura MQAXP indica quale funzione di uscita è stata richiamata. In alternativa, la chiamata MQXEP può essere utilizzata per registrare funzioni di uscita differenti per i due casi.

Estensione di connessione e connessione - MQ_CONNX_EXIT

MQ_CONNX_EXIT fornisce la funzione di uscita della connessione per eseguire *prima* e *dopo* l'elaborazione MQCONN e la funzione di uscita dell'estensione della connessione per eseguire *prima* e *dopo* l'elaborazione MQCONNX.

La stessa interfaccia, come descritta qui, viene richiamata sia per le funzioni di uscita chiamata MQCONN che MQCONNX.

Quando l'agent MCA (message channel agent) risponde a una connessione client in entrata, l'MCA può connettersi ed effettuare un numero di chiamate API IBM MQ prima che lo stato client sia completamente noto. Queste chiamate API richiamano le funzioni di uscita API con MQAXC in base al programma MCA stesso (ad esempio nei campi UserId e ConnectionName di MQAXC).

Quando l'MCA risponde alle successive chiamate API del client in entrata, la struttura MQAXC si basa sul client in entrata, impostando i campi UserId e ConnectionName in modo appropriato.

Il nome del gestore code impostato dall'applicazione su una chiamata MQCONN o MQCONNX viene passato alla chiamata di connessione sottostante. Qualsiasi tentativo da parte di *prima di* MQ_CONNX_EXIT di modificare il nome del gestore code non ha alcun effetto.

Utilizzare gli identificativi di funzione MQXF_CONN e MQXF_CONNX con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni di uscita chiamata MQCONN e MQCONNX.

Un'uscita MQ_CONNX_EXIT richiamata per il motivo per cui MQXR_BEFORE *non deve* emettere alcuna chiamata API IBM MQ , poiché l'ambiente corretto non è stato impostato in questo momento.

Un MQ_CONNX_EXIT non può chiamare MQDISC da una chiamata di uscita API per la connessione per cui viene richiamato. Questa limitazione è applicabile sia alle uscite API client che server.

L'interfaccia per MQCONN e MQCONNX è identica:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOpts,
&pHconn, &CompCode, &Reason);
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

pQMgrNome (PMQCHAR) - input
Puntatore al nome gestore code fornito nella chiamata MQCONNX. L'uscita non deve modificare questo nome nella chiamata MQCONN o MQCONNX.

pConnectOpts (PMQCNO) - input/output
Puntatore alle opzioni che controllano l'azione della chiamata MQCONNX.

Vedi "MQCNO - Opzioni di connessione" a pagina 321 per i dettagli.

Per la funzione di uscita MQXF_CONN, l'opzione pConnectpunta alla struttura delle opzioni di connessione predefinita (MQCNO_DEFAULT).

pHconn (PMQHCONN) - input
Puntatore all'handle di connessione.

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale)

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
        MQAXP           ExitParms;      /* Exit parameter structure */
        MQAXC           ExitContext;    /* Exit context structure */
        PMQCHAR         pQMgrName;      /* Ptr to Queue manager name */
        PMQCNO          pConnectOpts;   /* Ptr to Connection options */
        PMQHCONN        pHconn;         /* Ptr to Connection handle */
        MQLONG          CompCode;       /* Completion code */
        MQLONG          Reason;         /* Reason code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOps,
               &pHconn, &CompCode, &Reason);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_CONNX_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PPMQCHAR        ppQMgrName,     /* Address of ptr to queue manager name */
PPMQCNO         ppConnectOpts,  /* Address of ptr to connection options */
PPMQHCONN       ppHconn,        /* Address of ptr to connection handle */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Note d'utilizzo

1. L'interfaccia della funzione MQ_CONNX_EXIT qui descritta viene utilizzata sia per la chiamata MQCONN che per la chiamata MQCONNX. Tuttavia, per queste due chiamate vengono definiti punti di ingresso separati. Per intercettare *entrambe* le chiamate, la chiamata MQXEP deve essere utilizzata almeno due volte - una volta con l'ID funzione MQXF_CONN e di nuovo con MQXF_CONNX.

   Poiché l'interfaccia MQ_CONNX_EXIT è la stessa per MQCONN e MQCONNX, è possibile utilizzare una singola funzione di uscita per entrambe le chiamate; il campo *Function* nella struttura MQAXP indica quale chiamata è in corso. In alternativa, la chiamata MQXEP può essere utilizzata per registrare diverse funzioni di uscita per le due chiamate.
2. Quando un MCA (Message Channel Agent) risponde a una connessione client in entrata, l'MCA può emettere un certo numero di chiamate MQ prima che lo stato client sia completamente noto. Queste chiamate MQ risultano nelle funzioni di uscita API richiamate con la struttura MQAXC che contiene i dati relativi all'MCA e non al client (ad esempio, l'identificativo utente e il nome connessione). Tuttavia, una volta che lo stato del client è completamente noto, le successive richiami di MQ determinano il richiamo delle funzioni di uscita API con i dati client appropriati nella struttura MQAXC.
3. Tutte le funzioni di uscita MQXR_BEFORE vengono richiamate prima che la convalida dei parametri venga eseguita dal gestore code. I parametri potrebbero non essere validi (inclusi i puntatori non validi per gli indirizzi dei parametri).

   La funzione MQ_CONNX_EXIT viene richiamata prima che il gestore code esegua i controlli di autorizzazione.
4. La funzione di uscita non deve modificare il nome del gestore code specificato nella chiamata MQCONN o MQCONNX. Se il nome viene modificato dalla funzione di uscita, i risultati non sono definiti.
5. Una funzione di uscita MQXR_BEFORE per MQ_CONNX_EXIT non può emettere chiamate MQ diverse da MQXEP.

Callback di controllo - MQ_CTL_EXIT

MQ_CTL_EXIT fornisce una funzione di uscita della richiesta di sottoscrizione per eseguire *prima* e *dopo* l'elaborazione del callback di controllo. Utilizzare l'identificativo funzione MQXF_CTL con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni di uscita chiamata di controllo callback.

L'interfaccia per questa funzione è:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason)
```

dove i parametri sono:

Hconn (MQHCONN) - input/output
Handle di connessione.

Input / output operazione (MQLONG)
L'operazione in fase di elaborazione sul callback definito per il gestore oggetti specificato

Input / output di ControlOpts (MQCTLO)
Opzioni che controllano l'azione di MQCTL

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
        MQHCONN  Hconn;          /* Connection handle */
        MQLONG   Operation;      /* Operation being processed */
        MQCTLO   ControlOpts;    /* Options that control the action of MQCTL  */
        MQLONG   CompCode;       /* Completion code */
        MQLONG   Reason;         /* Reason code qualifying completion code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_CTL_EXIT (
PMQHCONN  pHconn;          /* Address of connection handle */
PMQLONG   pOperation;      /* Address of operation being processed */
PMQCTLO   pControlOpts;    /* Address of options that control the action of MQCTL  */
PMQLONG   pCompCode;       /* Address of completion code */
PMQLONG   pReason;)        /* Address of reason code qualifying completion code */
```

Disconnetti - MQ_DISC_EXIT

MQ_DISC_EXIT fornisce una funzione di uscita di disconnessione per eseguire *prima* e *dopo* l'elaborazione dell'uscita MQDISC. Utilizzare l'identificativo funzione MQXF_DISC con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni di uscita chiamata MQDISC.

L'interfaccia per questa funzione è

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
&CompCode, &Reason);
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

pHconn (PMQHCONN) - input
Puntatore all'handle di connessione.

Per la chiamata MQDISC precedente, il valore di questo campo è uno tra:

- L'handle di connessione restituito sulla chiamata MQCONN o MQCONNX
- Zero, per gli ambienti in cui un adattatore specifico dell'ambiente si è connesso al gestore code
- Un valore impostato da un richiamo di funzione di uscita precedente

Per la chiamata MQDISC successiva, il valore di questo campo è zero o un valore impostato da un richiamo della funzione di uscita precedente.

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
        MQAXP           ExitParms;      /* Exit parameter structure */
        MQAXC           ExitContext;    /* Exit context structure */
        PMQHCONN        pHconn;         /* Ptr to Connection handle */
        MQLONG          CompCode;       /* Completion code */
        MQLONG          Reason;         /* Reason code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
              &CompCode, &Reason);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_DISC_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PPMQHCONN       ppHconn,        /* Address of ptr to connection handle */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Ottieni - MQ_GET_EXIT

MQ_GET_EXIT fornisce una funzione get exit per eseguire *prima* e *dopo* l'elaborazione della chiamata MQGET.

Esistono due identificativi di funzioni:

1. Utilizzare MQXF_GET con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni di uscita chiamata MQGET.

2. Consultare “CONV_DATA_MQXF_GET” a pagina 1635 per informazioni sull'utilizzo dell'ID funzione MQXF_DATA_CONV_ON_GET.

L'interfaccia per questa funzione è:

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

Hobj (MQHOBJ) - input/output
Handle oggetto.

pMsgDesc (PMQMD) - input/output
Puntatore al descrittore messaggi.

pGetMsgOpts (PMQGMO) - input/output
Puntatore per richiamare le opzioni del messaggio.

BufferLength (MQLONG) - input/output
Lunghezza buffer messaggi.

pBuffer (PMQBYTE) - input/output
Puntatore al buffer messaggi.

pDataLength (PMQLONG) - input/output
Puntatore al campo lunghezza dati.

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
MQHCONN         Hconn;          /* Connection handle */
MQHOBJ          Hobj;           /* Object handle */
PMQMD           pMsgDesc;       /* Ptr to message descriptor */
```

```
        PMQPMO          pGetMsgOpts;    /* Ptr to get message options */
        MQLONG          BufferLength;   /* Message buffer length */
        PMQBYTE         pBuffer;        /* Ptr to message buffer */
        PMQLONG         pDataLength;    /* Ptr to data length field */
        MQLONG          CompCode;       /* Completion code */
        MQLONG          Reason;         /* Reason code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_GET_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQHCONN        pHconn,         /* Address of connection handle */
PMQHOBJ         pHobj,          /* Address of object handle */
PPMQMD          ppMsgDesc,      /* Address of ptr to message descriptor */
PPMQGMO         ppGetMsgOpts,   /* Address of ptr to get message options */
PMQLONG         pBufferLength,  /* Address of message buffer length */
PPMQBYTE        ppBuffer,       /* Address of ptr to message buffer */
PPMQLONG        ppDataLength,   /* Address of ptr to data length field */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Note d'utilizzo

1. L'interfaccia della funzione MQ_GET_EXIT qui descritta viene utilizzata sia per la funzione di uscita MQXF_GET che per quella di uscita “CONV_DATA_MQXF_GET” a pagina 1635 .

   Per queste due funzioni di uscita sono definiti punti di ingresso separati, per cui per intercettare *entrambi* la chiamata MQXEP deve essere utilizzata due volte; per questa chiamata utilizzare l'identificativo funzione MQXF_GET.

   Poiché l'interfaccia MQ_GET_EXIT è la stessa per MQXF_GET e MQXF_DATA_CONV_ON_GET, è possibile utilizzare una singola funzione di uscita per entrambi; il campo *Function* nella struttura MQAXP indica quale funzione di uscita è stata richiamata. In alternativa, la chiamata MQXEP può essere utilizzata per registrare funzioni di uscita differenti per i due casi.

CONV_DATA_MQXF_GET

L'identificativo della funzione MQXF_DATA_CONV_ON_GET viene utilizzato con MQ_GET_EXIT.

Consultare MQ_GET_EXIT per informazioni sull'interfaccia per questa chiamata e una dichiarazione di linguaggio C di esempio.

Note d'utilizzo

Se registrato, questo punto di ingresso viene richiamato quando i messaggi arrivano all'applicazione ma prima che si verifichi qualsiasi conversione di dati. Ciò può essere utile se l'uscita API deve eseguire l'elaborazione, come la decodifica o la decompressione, prima che il messaggio venga passato alla conversione dati. L'uscita può, se necessario, impedire la conversione dei dati restituendo MQXCC_SUPPRESS_FUNCTION; per ulteriori informazioni, consultare la struttura MQAXP .

La registrazione di questo punto di ingresso su un client ha l'effetto di far sì che la conversione dati venga eseguita localmente sulla macchina client. Per un corretto funzionamento potrebbe quindi essere necessario installare le uscite di conversione dell'applicazione sul client. Notare che MQXF_DATA_CONV_ON_GET viene utilizzato anche per il consumo asincrono.

Quando si utilizza la chiamata MQ_GET_EXIT, utilizzare MQXF_DATA_CONV_ON_GET, con motivo di uscita MQXR_BEFORE, per registrare una funzione di uscita di conversione dati MQGET *prima* .

Non vi è alcuna funzione di uscita MQXR_AFTER per MQXF_DATA_CONV_ON_GET; la funzione di uscita MQXR_AFTER per MQXF_GET fornisce la funzionalità richiesta per l'elaborazione dell'uscita dopo la conversione dei dati.

I punti di ingresso separati sono definiti per la chiamata MQ_GET_EXIT, quindi per intercettare *entrambe* le funzioni di uscita, la chiamata MQXEP deve essere utilizzata due volte; per questa chiamata utilizzare l'identificativo della funzione MQXF_DATA_CONV_ON_GET.

Poiché l'interfaccia MQ_GET_EXIT è la stessa per MQXF_GET e MQXF_DATA_CONV_ON_GET, è possibile utilizzare una singola funzione di uscita per entrambi; il campo *Function* nella struttura MQAXP indica quale funzione di uscita è stata richiamata. In alternativa, la chiamata MQXEP può essere utilizzata per registrare funzioni di uscita differenti per i due casi.

Inizializzazione - MQ_INIT_EXIT

MQ_INIT_EXIT fornisce l'inizializzazione del livello di connessione, indicato impostando ExitReason in MQAXP su MQXR_CONNECTION.

Durante l'inizializzazione, tenere presente quanto segue:

- La funzione MQ_INIT_EXIT richiama MQXEP per registrare i verbi API IBM MQ e i punti ENTRY ed EXIT a cui è interessato.
- Le uscite non devono intercettare tutti i verbi API IBM MQ . Le funzioni di uscita vengono richiamate solo se è stato registrato un interesse.
- La memoria che deve essere utilizzata dall'uscita può essere acquisita durante l'inizializzazione.
- Se una chiamata a questa funzione non riesce, la chiamata MQCONN o MQCONNX che la ha richiamata ha esito negativo anche con un CompCode e un Reason che dipendono dal valore del campo ExitResponse in MQAXP.
- Un'uscita MQ_INIT_EXIT non deve emettere chiamate API IBM MQ , perché l'ambiente corretto non è stato configurato in questo momento.
- Se un MQ_INIT_EXIT ha esito negativo con MQXCC_FAILED, il gestore code ritorna dalla chiamata MQCONN o MQCONNX che lo ha richiamato con MQCC_FAILED e MQRC_API_EXIT_ERROR.
- Se il gestore code rileva un errore durante l'inizializzazione dell'ambiente di esecuzione della funzione di uscita API prima di richiamare il primo MQ_INIT_EXIT, il gestore code ritorna dalla chiamata MQCONN o MQCONNX che ha richiamato il MQ_INIT_EXIT con MQCC_FAILED e MQRC_API_EXIT_INIT_ERROR.

L'interfaccia di MQ_INIT_EXIT è:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

CompCode (MQLONG) - input/output
Puntatore al codice di completamento, i valori validi per cui sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Puntatore al codice causa che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Il CompCode e il motivo restituiti all'applicazione dipendono dal valore del campo ExitResponse in MQAXP.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
        MQAXP           ExitParms;      /* Exit parameter structure */
        MQAXC           ExitContext;    /* Exit context structure */
        MQLONG          CompCode;       /* Completion code */
        MQLONG          Reason;         /* Reason code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_INIT_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Note d'utilizzo

1. La funzione MQ_INIT_EXIT può emettere la chiamata MQXEP per registrare gli indirizzi delle funzioni di uscita per le specifiche chiamate MQ da intercettare. Non è necessario intercettare tutte le chiamate MQ o intercettare entrambe le chiamate MQXR_BEFORE e MQXR_AFTER. Ad esempio, una suite di uscita potrebbe decidere di intercettare solo la chiamata MQXR_BEFORE di MQPUT.
2. La memoria che deve essere utilizzata dalle funzioni di uscita nella suite di uscita può essere acquisita dalla funzione MQ_INIT_EXIT. In alternativa, le funzioni di uscita possono acquisire memoria quando vengono richiamate, come e quando necessario. Tuttavia, tutta la memoria deve essere liberata prima che la suite di uscita venga terminata; la funzione MQ_TERM_EXIT può liberare la memoria o una funzione di uscita richiamata in precedenza.
3. Se MQ_INIT_EXIT restituisce MQXCC_FAILED nel campo `ExitResponse` di MQAXP o non riesce in altro modo, la chiamata MQCONN o MQCONNX che ha causato il richiamo di MQ_INIT_EXIT non riesce, con i parametri **CompCode** e **Reason** impostati sui valori appropriati.
4. Una funzione MQ_INIT_EXIT non può emettere chiamate di MQ diverse da MQXEP.

Interrogazione - MQ_INQ_EXIT

MQ_INQ_EXIT fornisce una funzione di uscita di interrogazione per eseguire *prima* e *dopo* l'elaborazione delle chiamate MQINQ. Utilizzare l'identificativo della funzione MQXF_INQ con i motivi dell'exit MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni dell'exit di chiamata MQINQ.

L'interfaccia per questa funzione è:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

Hobj (MQHOBJ) - input
Handle oggetto.

SelectorCount (MQLONG) - input
Conteggio dei selettori

pSelectors (PMQLONG) - input/output
Puntatore alla schiera di valori del selettore.

Conteggio IntAttr(MQLONG) - input
Conteggio degli attributi interi.

pIntAttrs (PMQLONG) - input/output
Puntatore alla schiera di valori di attributo interi.

CharAttrLength (MQLONG) - input/output
Lunghezza schiera attributi carattere.

pCharAttrs (PMQCHAR) - input/output
Puntatore alla schiera attributi carattere.

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
MQAXP    ExitParms;       /* Exit parameter structure */
MQAXC    ExitContext;     /* Exit context structure */
MQHCONN  Hconn;           /* Connection handle */
MQHOBJ   Hobj;            /* Object handle */
MQLONG   SelectorCount;   /* Count of selectors */
PMQLONG  pSelectors;      /* Ptr to array of attribute selectors */
MQLONG   IntAttrCount;    /* Count of integer attributes */
PMQLONG  pIntAttrs;       /* Ptr to array of integer attributes */
MQLONG   CharAttrLength;  /* Length of char attributes array */
PMQCHAR  pCharAttrs;      /* Ptr to character attributes */
MQLONG   CompCode;        /* Completion code */
MQLONG   Reason;          /* Reason code qualifying completion code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_INQ_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PMQHOBJ   pHobj,            /* Address of object handle */
PMQLONG   pSelectorCount,   /* Address of selector count */
PPMQLONG  ppSelectors,      /* Address of ptr to array of selectors */
PMQLONG   pIntAttrCount;    /* Address of count of integer attributes */
PPMQLONG  ppIntAttrs,       /* Address of ptr to array of integer attributes */
PMQLONG   pCharAttrLength,  /* Address of character attribute length */
PPMQCHAR  ppCharAttrs,      /* Address of ptr to character attributes array */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

Apri - MQ_OPEN_EXIT

MQ_OPEN_EXIT fornisce una funzione di uscita aperta per eseguire *prima* e *dopo* l'elaborazione delle chiamate MQOPEN. Utilizzare l'identificativo funzione MQXF_OPEN con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni di uscita chiamata MQOPEN.

L'interfaccia per questa funzione è

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
&pHobj, &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pObjDesc (PMQOD) - input/output
Puntatore al descrittore oggetto.

Opzioni (MQLONG) - input/output
Opzioni di apertura.

pHobj (PMQHOBJ) - input
Puntatore alla gestione oggetto.

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
MQHCONN         Hconn;          /* Connection handle */
PMQOD           pObjDesc;       /* Ptr to object descriptor */
MQLONG          Options;        /* Open options */
PMQHOBJ         pHobj;          /* Ptr to object handle */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
              &pHobj, &CompCode, &Reason);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_OPEN_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQHCONN        pHconn,         /* Address of connection handle */
PPMQOD          ppObjDesc,      /* Address of ptr to object descriptor */
PMQLONG         pOptions,       /* Address of open options */
PPMQHOBJ        ppHobj,         /* Address of ptr to object handle */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Inserimento - MQ_PUT_EXIT

MQ_PUT_EXIT fornisce una funzione put exit per eseguire *prima* e *dopo* l'elaborazione delle chiamate MQPUT. Utilizzare l'identificativo funzione MQXF_PUT con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni di uscita di chiamata MQPUT.

L'interfaccia per questa funzione è:

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

Hobj (MQHOBJ) - input/output
Handle oggetto.

pMsgDesc (PMQMD) - input/output
Puntatore al descrittore messaggi.

pPutMsgOpts (PMQPMO) - input/output
Puntatore per inserire le opzioni del messaggio.

BufferLength (MQLONG) - input/output
Lunghezza buffer messaggi.

pBuffer (PMQBYTE) - input/output
Puntatore al buffer messaggi.

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
MQHCONN         Hconn;          /* Connection handle */
MQHOBJ          Hobj;           /* Object handle */
PMQMD           pMsgDesc;       /* Ptr to message descriptor */
PMQPMO          pPutMsgOpts;    /* Ptr to put message options */
MQLONG          BufferLength;   /* Message buffer length */
PMQBYTE         pBuffer;        /* Ptr to message data */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_PUT_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQHCONN        pHconn,         /* Address of connection handle */
PMQHOBJ         pHobj,          /* Address of object handle */
PPMQMD          ppMsgDesc,      /* Address of ptr to message descriptor */
PPMQPMO         ppPutMsgOpts,   /* Address of ptr to put message options */
PMQLONG         pBufferLength,  /* Address of message buffer length */
PPMQBYTE        ppBuffer,       /* Address of ptr to message buffer */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Note d'utilizzo

- I messaggi di report generati dal gestore code ignorano la normale elaborazione della chiamata. Di conseguenza, tali messaggi non possono essere intercettati dalla funzione MQ_PUT_EXIT o

MQPUT1 . Tuttavia, i messaggi di report generati dall'agent del canale dei messaggi vengono elaborati normalmente e, di conseguenza, possono essere intercettati dalla funzione MQ_PUT_EXIT o dalla funzione MQ_PUT1_EXIT . Per assicurarsi di intercettare tutti i messaggi di report generati dall'MCA, è necessario utilizzare sia MQ_PUT_EXIT che MQ_PUT1_EXIT .

Put1 - MQ_PUT1_EXIT

MQ_PUT1_EXIT fornisce la funzione di uscita *Inserisci solo un messaggio* per eseguire *prima* e *dopo l'elaborazione della chiamata* MQPUT1 . Utilizzare l'identificativo funzione MQXF_PUT1 con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo le funzioni di uscita chiamata* MQPUT1 .

L'interfaccia per questa funzione è:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
&pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pObjDesc (PMQOD) - input/output
Puntatore al descrittore oggetto.

pMsgDesc (PMQMD) - input/output
Puntatore al descrittore messaggi.

pPutMsgOpts (PMQPMO) - input/output
Puntatore per inserire le opzioni del messaggio.

BufferLength (MQLONG) - input/output
Lunghezza buffer messaggi.

pBuffer (PMQBYTE) - input/output
Puntatore al buffer messaggi.

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
        MQAXP           ExitParms;      /* Exit parameter structure */
        MQAXC           ExitContext;    /* Exit context structure */
        MQHCONN         Hconn;          /* Connection handle */
        PMQOD           pObjDesc;       /* Ptr to object descriptor */
        PMQMD           pMsgDesc;       /* Ptr to message descriptor */
        PMQPMO          pPutMsgOpts;    /* Ptr to put message options */
        MQLONG          BufferLength;   /* Message buffer length */
        PMQBYTE         pBuffer;        /* Ptr to message data */
        MQLONG          CompCode;       /* Completion code */
        MQLONG          Reason;         /* Reason code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
              &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_PUT1_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQHCONN        pHconn,         /* Address of connection handle */
PPMQOD          ppObjDesc,      /* Address of ptr to object descriptor */
PPMQMD          ppMsgDesc,      /* Address of ptr to message descriptor */
PPMQPMO         ppPutMsgOpts,   /* Address of ptr to put message options */
PMQLONG         pBufferLength,  /* Address of message buffer length */
PPMQBYTE        ppBuffer,       /* Address of ptr to message buffer */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Imposta - MQ_SET_EXIT

MQ_SET_EXIT fornisce una funzione di uscita impostata per eseguire *prima* e *dopo* l'elaborazione della chiamata MQSET. Utilizzare l'identificativo della funzione MQXF_SET con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni di uscita chiamata MQSET.

L'interfaccia per questa funzione è:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttr, &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

Hobj (MQHOBJ) - input
Handle oggetto.

SelectorCount (MQLONG) - input
Conteggio dei selettori

pSelectors (PMQLONG) - input/output
Puntatore alla schiera di valori del selettore.

Conteggio IntAttr(MQLONG) - input
Conteggio degli attributi interi.

pIntAttrs (PMQLONG) - input/output
Puntatore alla schiera di valori di attributo interi.

CharAttrLength (MQLONG) - input/output
Lunghezza schiera attributi carattere.

pCharAttrs (PMQCHAR) - input/output
Puntatore ai valori dell'attributo carattere.

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
MQAXP    ExitParms;       /* Exit parameter structure */
MQAXC    ExitContext;     /* Exit context structure */
MQHCONN  Hconn;           /* Connection handle */
MQHOBJ   Hobj;            /* Object handle */
MQLONG   SelectorCount;   /* Count of selectors */
PMQLONG  pSelectors;      /* Ptr to array of attribute selectors */
MQLONG   IntAttrCount;    /* Count of integer attributes */
PMQLONG  pIntAttrs;       /* Ptr to array of integer attributes */
MQLONG   CharAttrLength;  /* Length of char attributes array */
PMQCHAR  pCharAttrs;      /* Ptr to character attributes */
MQLONG   CompCode;        /* Completion code */
MQLONG   Reason;          /* Reason code qualifying completion code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_SET_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PMQHOBJ   pHobj,            /* Address of object handle */
PMQLONG   pSelectorCount,   /* Address of selector count */
PPMQLONG  ppSelectors,      /* Address of ptr to array of selectors */
PMQLONG   pIntAttrCount;    /* Address of count of integer attributes */
PPMQLONG  ppIntAttrs,       /* Address of ptr to array of integer attributes */
PMQLONG   pCharAttrLength,  /* Address of character attribute length */
PPMQCHAR  ppCharAttrs,      /* Address of ptr to character attributes array */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

Stato - MQ_STAT_EXIT

MQ_STAT_EXIT fornisce una funzione di uscita di stato per eseguire *prima* e *dopo* l'elaborazione della chiamata MQSTAT. Utilizzare l'identificativo della funzione MQXF_STAT con i motivi dell'uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni dell'uscita della chiamata MQSTAT.

L'interfaccia per questa funzione è:

```
MQ_STAT_EXIT (&ExitParms, &ExitContext, &Hconn, &Type, &pStatus
              &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

Tipo (MQLONG) - input
Tipo di informazioni di stato da recuperare.

pStatus (PMQSTS) - output
Puntatore al buffer di stato.

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_STAT_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PMQLONG   pType             /* Address of status type */
PPMQSTS   ppStatus          /* Address of status buffer */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

Terminazione - MQ_TERM_EXIT

MQ_TERM_EXIT fornisce la terminazione del livello di connessione, registrata con l'ID funzione MQXF_TERM e ExitReason MQXR_CONNECTION. Se registrato, MQ_TERM_EXIT viene richiamato una volta per ogni richiesta di disconnessione.

Come parte della terminazione, la memoria non più richiesta dall'uscita può essere rilasciata e qualsiasi ripulitura richiesta può essere eseguita.

Se un MQ_TERM_EXIT ha esito negativo con MQXCC_FAILED, il gestore code ritorna da MQDISC che lo ha richiamato con MQCC_FAILED e MQRC_API_EXIT_ERROR.

Se il gestore code rileva un errore durante la chiusura dell'ambiente di esecuzione della funzione di uscita API dopo aver richiamato l'ultimo MQ_TERM_EXIT, il gestore code restituisce il messaggio dalla chiamata MQDISC che ha richiamato MQ_TERM_EXIT con MQCC_FAILED e MQRC_API_EXIT_TERM_ERROR

L'interfaccia per questa funzione è:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Il CompCode e il motivo restituiti all'applicazione dipendono dal valore del campo ExitResponse in MQAXP.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_TERM_EXIT (
PMQAXP          pExitParms,     /* Address of exit parameter structure */
PMQAXC          pExitContext,   /* Address of exit context structure */
PMQLONG         pCompCode,      /* Address of completion code */
PMQLONG         pReason);       /* Address of reason code qualifying
                                   completion code */
```

Note d'utilizzo

1. La funzione MQ_TERM_EXIT è facoltativa. Non è necessario che una suite di uscita registri un'uscita di terminazione se non vi è alcuna elaborazione di terminazione da eseguire.

   Se le funzioni appartenenti alla suite di uscita acquisiscono le risorse durante la connessione, una funzione MQ_TERM_EXIT è un punto conveniente in cui liberare tali risorse, ad esempio, liberando la memoria ottenuta dinamicamente.
2. Se una funzione MQ_TERM_EXIT viene registrata quando viene emessa la chiamata MQDISC, la funzione di uscita viene richiamata dopo che sono state richiamate tutte le funzioni di uscita MQDISC.
3. Se MQ_TERM_EXIT restituisce MQXCC_FAILED nel campo `ExitResponse` di MQAXP o non riesce in altro modo, anche la chiamata MQDISC che ha causato il richiamo di MQ_TERM_EXIT ha esito negativo, con i parametri **CompCode** e **Reason** impostati sui valori appropriati.

Registra sottoscrizione - MQ_SUB_EXIT

MQ_SUB_EXIT fornisce una funzione di uscita per eseguire *prima* e *dopo* l'elaborazione della riregistrazione della sottoscrizione. Utilizzare l'identificativo funzione MQXF_SUB con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni di uscita registrationcall della sottoscrizione.

L'interfaccia per questa funzione è:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub, &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input/output
Handle di connessione.

pSubDesc - input/output
Array di selettori di attributo.

pHobj - input/output
Handle di oggetti

Input / output pHsub (MQHOBJ)
Handle sottoscrizione

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
MQAXP    ExitParms;      /* Exit parameter structure */
MQAXC    ExitContext;    /* Exit context structure */
MQHCONN  Hconn;          /* Connection handle */
PMQSD    pSubDesc;       /* Subscription descriptor */
PMQHOBJ  pHobj;          /* Object Handle */
PMQHOBJ  pHsub;          /* Subscription handle */
MQLONG   CompCode;       /* Completion code */
MQLONG   Reason;         /* Reason code qualifying completion code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub,
             &CompCode, &Reason);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
PMQAXP    pExitParms;      /* Exit parameter structure */
PMQAXC    pExitContext;    /* Exit context structure */
PMQHCONN  pHconn;          /* Connection handle */
PPMQSD    ppSubDesc;       /* Subscription descriptor */
PPMQHOBJ  ppHobj;          /* Object Handle */
PPMQHOBJ  ppHsub;          /* Subscription handle */
PMQLONG   pCompCode;       /* Completion code */
PMQLONG   pReason;         /* Reason code qualifying completion code */
```

Richiesta di sottoscrizione - MQ_SUBRQ_EXIT

MQ_SUBRQ_EXIT fornisce una funzione di uscita richiesta di sottoscrizione per eseguire *prima* e *dopo* l'elaborazione della richiesta di sottoscrizione. Utilizzare l'identificativo della funzione MQXF_SUBRQ con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare *prima* e *dopo* le funzioni di uscita di chiamata della richiesta di sottoscrizione.

L'interfaccia per questa funzione è:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
               &CompCode, &Reason)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input/output
Handle di connessione.

Input / output pHsub (MQHOBJ)
Handle sottoscrizione

Input / output dell'azione (MQLONG)
Azione

pSubRqOpts (MQSRO) input/output

CompCode (MQLONG) - input/output
Codice di completamento, i cui valori validi sono:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata non riuscita

Motivo (MQLONG) - input/output
Codice motivo che qualifica il codice di completamento.

Se il codice di completamento è MQCC_OK, l'unico valore valido è:

MQRC_NONE
(0, x '000') Nessun motivo per segnalare.

Se il codice di completamento è MQCC_FAILED o MQCC_WARNING, la funzione di uscita può impostare il campo del codice motivo su qualsiasi valore MQRC_ * valido.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
        MQAXP    ExitParms;      /* Exit parameter structure */
        MQAXC    ExitContext;    /* Exit context structure */
        MQHCONN  Hconn;          /* Connection handle */
        PMQLONG  pHsub;          /* Subscription handle */
        MQLONG   Action;         /* Action */
        PMQSRO   pSubRqOpts;     /* Subscription Request Options */
        MQLONG   CompCode;       /* Completion code */
        MQLONG   Reason;         /* Reason code qualifying completion code */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
               &CompCode, &Reason);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
void MQENTRY MQ_SUBRQ_EXIT (
PMQAXP    pExitParms,       /* Address of exit parameter structure */
PMQAXC    pExitContext,     /* Address of exit context structure */
PMQHCONN  pHconn,           /* Address of connection handle */
PPMQHOBJ  ppHsub;           /* Address of Subscription handle */
PMQLONG   pAction;          /* Address of Action */
PPMQSRO   ppSubRqOpts;      /* Address of Subscription Request Options */
PMQLONG   pCompCode,        /* Address of completion code */
PMQLONG   pReason);         /* Address of reason code qualifying completion
                               code */
```

xa_close - XA_CLOSE_EXIT

XA_CLOSE_EXIT fornisce una funzione di uscita xa_close da eseguire prima e dopo l'elaborazione di xa_close. Utilizzare l'identificativo della funzione MQXF_XACLOSE con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare le funzioni di uscita di chiamata prima e dopo xa_close.

L'interfaccia per questa funzione è:

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pXa_info (PMQCHAR) - input/output
Informazioni sul gestore risorse specifiche dell'istanza.

Rmid (MQLONG) - input/output
Identificativo gestore risorse.

Indicatori (MQLONG) - input/output
Opzioni del gestore risorse.

XARetCode (MQLONG) - input/output
Risposta dalla chiamata XA.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
     MQAXP    ExitParms;    /* Exit parameter structure */
     MQAXC    ExitContext;  /* Exit context structure */
     MQHCONN  Hconn;        /* Connection handle */
     PMQCHAR  pXa_info;     /* Instance-specific RM info */
     MQLONG   Rmid;         /* Resource manager identifier */
     MQLONG   Flags;        /* Resource manager options*/
     MQLONG   XARetCode;    /* Response from XA call */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
typedef void MQENTRY XA_CLOSE_EXIT (
     PMQAXP    pExitParms,    /* Address of exit parameter structure */
     PMQAXC    pExitContext,  /* Address of exit context structure */
     PMQHCONN  pHconn,        /* Address of connection handle */
     PPMQCHAR  ppXa_info,     /* Address of instance-specific RM info */
     PMQLONG   pRmid,         /* Address of resource manager identifier */
     PMQLONG   pFlags,        /* Address of resource manager options*/
     PMQLONG   pXARetCode);   /* Address of response from XA call */
```

xa_commit - XA_COMMIT_EXIT

XA_COMMIT_EXIT fornisce una funzione di uscita xa_commit da eseguire prima e dopo l'elaborazione di xa_commit. Utilizzare l'identificativo funzione MQXF_XACOMMIT con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare le funzioni di uscita chiamata prima e dopo xa_commit.

L'interfaccia per questa funzione è:

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pXID (MQPTR) - input/output
ID ramo transazione.

Rmid (MQLONG) - input/output
Identificativo gestore risorse.

Indicatori (MQLONG) - input/output
Opzioni del gestore risorse.

XARetCode (MQLONG) - input/output
Risposta dalla chiamata XA.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
     MQAXP    ExitParms;   /* Exit parameter structure */
     MQAXC    ExitContext; /* Exit context structure */
     MQHCONN  Hconn;       /* Connection handle */
     MQPTR    pXID;        /* Transaction branch ID */
     MQLONG   Rmid;        /* Resource manager identifier */
     MQLONG   Flags;       /* Resource manager options*/
     MQLONG   XARetCode;   /* Response from XA call */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
typedef void MQENTRY XA_COMMIT_EXIT (
     PMQAXP    pExitParms,   /* Address of exit parameter structure */
     PMQAXC    pExitContext, /* Address of exit context structure */
     PMQHCONN  pHconn,       /* Address of connection handle */
     PMQPTR    ppXID,        /* Address of transaction branch ID */
     PMQLONG   pRmid,        /* Address of resource manager identifier */
     PMQLONG   pFlags,       /* Address of resource manager options*/
     PMQLONG   pXARetCode);  /* Address of response from XA call */
```

xa_complete - XA_COMPLETE_EXIT

XA_COMPLETE_EXIT fornisce una funzione exit xa_complete da eseguire prima e dopo l'elaborazione di xa_complete. Utilizzare l'identificativo della funzione MQXF_XACOMPLETE con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare le funzioni di uscita chiamata prima e dopo xa_complete.

L'interfaccia per questa funzione è:

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetval, &Rmid, &Flags,
&XARetCode)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pHandle (PMQLONG) - input/output
Puntatore all'operazione asincrona.

pRetVal (PMQLONG) - input/output
Valore di ritorno dell'operazione asincrona.

Rmid (MQLONG) - input/output
Identificativo gestore risorse.

Indicatori (MQLONG) - input/output
Opzioni del gestore risorse.

XARetCode (MQLONG) - input/output
Risposta dalla chiamata XA.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
     MQAXP   ExitParms;   /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     PMQLONG pHandle;     /* Ptr to asynchronous op */
     PMQLONG pRetval;     /* Return value of async op */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetval, &Rmid, &Flags,
&XARetCode);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
typedef void MQENTRY XA_COMPLETE_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PPMQLONG ppHandle,     /* Address of ptr to asynchronous op */
     PPMQLONG ppRetval,     /* Address of return value of async op */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_end - XA_END_EXIT

XA_END_EXIT fornisce una funzione di uscita xa_end da eseguire prima e dopo l'elaborazione xa_end. Utilizzare l'identificativo della funzione MQXF_XAEND con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare le funzioni di uscita di chiamata prima e dopo xa_end.

L'interfaccia per questa funzione è:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pXID (MQPTR) - input/output
ID ramo transazione.

Rmid (MQLONG) - input/output
Identificativo gestore risorse.

Indicatori (MQLONG) - input/output
Opzioni del gestore risorse.

XARetCode (MQLONG) - input/output
Risposta dalla chiamata XA.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
     MQAXP   ExitParms;   /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     MQPTR   pXID;        /* Transaction branch ID */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
typedef void MQENTRY XA_END_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PMQPTR   ppXID,        /* Address of transaction branch ID */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_forget - XA_FORGET_EXIT

XA_FORGET_EXIT fornisce una funzione di uscita xa_forget da eseguire prima e dopo l'elaborazione xa_forget. Utilizzare l'identificativo della funzione MQXF_XAFORGET con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare le funzioni di uscita chiamata prima e dopo xa_forget.

L'interfaccia per questa funzione è:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pXID (MQPTR) - input/output
ID ramo transazione.

Rmid (MQLONG) - input/output
Identificativo gestore risorse.

Indicatori (MQLONG) - input/output
Opzioni del gestore risorse.

XARetCode (MQLONG) - input/output
Risposta dalla chiamata XA.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
     MQAXP   ExitParms;   /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     MQPTR   pXID;        /* Transaction branch ID */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
typedef void MQENTRY XA_FORGET_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PMQPTR   ppXID,        /* Address of transaction branch ID */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_open - XA_OPEN_EXIT

XA_OPEN_EXIT fornisce una funzione di uscita xa_open da eseguire prima e dopo l'elaborazione di xa_open. Utilizzare l'identificativo funzione MQXF_XAOPEN con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare le funzioni di uscita chiamata prima e dopo xa_open.

L'interfaccia per questa funzione è:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pXa_info (PMQCHAR) - input/output
Informazioni sul gestore risorse specifiche dell'istanza.

Rmid (MQLONG) - input/output
Identificativo gestore risorse.

Indicatori (MQLONG) - input/output
Opzioni del gestore risorse.

XARetCode (MQLONG) - input/output
Risposta dalla chiamata XA.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
     MQAXP   ExitParms;   /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
```

```
     PMQCHAR pXa_info;    /* Instance-specific RM info */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
typedef void MQENTRY XA_OPEN_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PPMQCHAR ppXa_info,    /* Address of instance-specific RM info */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_prepare - XA_PREPARE_EXIT

XA_PREPARE_EXIT fornisce una funzione di uscita xa_prepare da eseguire prima e dopo l'elaborazione di xa_prepare. Utilizzare l'identificativo della funzione MQXF_XAPREPARE con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare le funzioni di uscita chiamata prima e dopo xa_prepare.

L'interfaccia per questa funzione è:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pXID (MQPTR) - input/output
ID ramo transazione.

Rmid (MQLONG) - input/output
Identificativo gestore risorse.

Indicatori (MQLONG) - input/output
Opzioni del gestore risorse.

XARetCode (MQLONG) - input/output
Risposta dalla chiamata XA.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
MQAXP  ExitParms;  /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     MQPTR   pXID;        /* Transaction branch ID */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
typedef void MQENTRY XA_PREPARE_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PMQPTR   ppXID,        /* Address of transaction branch ID */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_recover - XA_RECOVER_EXIT

XA_RECOVER_EXIT fornisce una funzione di uscita xa_recover da eseguire prima e dopo l'elaborazione di xa_recover. Utilizzare l'identificativo funzione MQXF_XARECOVER con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare le funzioni di uscita della chiamata prima e dopo xa_recover.

L'interfaccia per questa funzione è:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pXID (MQPTR) - input/output
ID ramo transazione.

Conteggio (MQLONG) - input/output
Numero massimo di XID nell'array XID

Rmid (MQLONG) - input/output
Identificativo gestore risorse.

Indicatori (MQLONG) - input/output
Opzioni del gestore risorse.

XARetCode (MQLONG) - input/output
Risposta dalla chiamata XA.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
     MQAXP   ExitParms;   /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     MQPTR   pXID;        /* Transaction branch ID */
     MQLONG  Count;       /* Max XIDs in XID array */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
typedef void MQENTRY XA_RECOVER_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PMQPTR   ppXID,        /* Address of transaction branch ID */
     PMQLONG  pCount,       /* Address of max XIDs in XID array */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_rollback - XA_ROLLBACK_EXIT

XA_ROLLBACK_EXIT fornisce una funzione di uscita xa_rollback da eseguire prima e dopo l'elaborazione xa_rollback. Utilizzare l'ID funzione MQXF_XAROLLBACK con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare le funzioni di uscita chiamata prima e dopo xa_rollback.

L'interfaccia per questa funzione è:

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pXID (MQPTR) - input/output
ID ramo transazione.

Rmid (MQLONG) - input/output
Identificativo gestore risorse.

Indicatori (MQLONG) - input/output
Opzioni del gestore risorse.

XARetCode (MQLONG) - input/output
Risposta dalla chiamata XA.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
     MQAXP   ExitParms;   /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     MQPTR   pXID;        /* Transaction branch ID */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
typedef void MQENTRY XA_ROLLBACK_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
```

```
     PMQPTR   ppXID,        /* Address of transaction branch ID */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_start - XA_START_EXIT

XA_START_EXIT fornisce una funzione di uscita xa_start da eseguire prima e dopo l'elaborazione di xa_start. Utilizzare l'identificativo della funzione MQXF_XASTART con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare le funzioni di uscita di chiamata prima e dopo xa_start.

L'interfaccia per questa funzione è:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pXID (MQPTR) - input/output
ID ramo transazione.

Rmid (MQLONG) - input/output
Identificativo gestore risorse.

Indicatori (MQLONG) - input/output
Opzioni del gestore risorse.

XARetCode (MQLONG) - input/output
Risposta dalla chiamata XA.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
     MQAXP   ExitParms;   /* Exit parameter structure */
     MQAXC   ExitContext; /* Exit context structure */
     MQHCONN Hconn;       /* Connection handle */
     MQPTR   pXID;        /* Transaction branch ID */
     MQLONG  Rmid;        /* Resource manager identifier */
     MQLONG  Flags;       /* Resource manager options*/
     MQLONG  XARetCode;   /* Response from XA call */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
typedef void MQENTRY XA_START_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PMQPTR   ppXID,        /* Address of transaction branch ID */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

ax_reg - AX_REG_EXIT

AX_REG_EXIT fornisce una funzione di uscita ax_reg da eseguire prima e dopo l'elaborazione ax_reg. Utilizzare l'identificatore funzione MQXF_AXREG con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare le funzioni di uscita chiamata prima e dopo ax_reg.

L'interfaccia per questa funzione è:

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode)
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Hconn (MQHCONN) - input
Handle di connessione.

pXID (MQPTR) - input/output
ID ramo transazione.

Rmid (MQLONG) - input/output
Identificativo gestore risorse.

Indicatori (MQLONG) - input/output
Opzioni del gestore risorse.

XARetCode (MQLONG) - input/output
Risposta dalla chiamata XA.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
     MQAXP  ExitParms;   /* Exit parameter structure */
     MQAXC  ExitContext; /* Exit context structure */
     MQPTR  pXID;        /* Transaction branch ID */
     MQLONG Rmid;        /* Resource manager identifier */
     MQLONG Flags;       /* Resource manager options*/
     MQLONG XARetCode;   /* Response from XA call */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
typedef void MQENTRY AX_REG_EXIT (
     PMQAXP  pExitParms,   /* Address of exit parameter structure */
     PMQAXC  pExitContext, /* Address of exit context structure */
     PMQPTR  ppXID,        /* Address of transaction branch ID */
     PMQLONG pRmid,        /* Address of resource manager identifier */
     PMQLONG pFlags,       /* Address of resource manager options*/
     PMQLONG pXARetCode);  /* Address of response from XA call */
```

ax_unreg - AX_UNREG_EXIT

AX_UNREG_EXIT fornisce una funzione di uscita ax_unreg da eseguire prima e dopo l'elaborazione ax_unreg. Utilizzare l'identificatore funzione MQXF_AXUNREG con i motivi di uscita MQXR_BEFORE e MQXR_AFTER per registrare le funzioni di uscita chiamata prima e dopo ax_unreg.

L'interfaccia per questa funzione è:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

dove i parametri sono:

ExitParms (MQAXP) - input/output
Struttura del parametro di uscita.

ExitContext (MQAXC) - input/output
Struttura del contesto di uscita.

Rmid (MQLONG) - input/output
Identificativo gestore risorse.

Indicatori (MQLONG) - input/output
Opzioni del gestore risorse.

XARetCode (MQLONG) - input/output
Risposta dalla chiamata XA.

Richiamo linguaggio C

Il gestore code definisce logicamente le seguenti variabili:

```
     MQAXP  ExitParms;   /* Exit parameter structure */
     MQAXC  ExitContext; /* Exit context structure */
     MQLONG Rmid;        /* Resource manager identifier */
     MQLONG Flags;       /* Resource manager options*/
     MQLONG XARetCode;   /* Response from XA call */
```

Il gestore code richiama logicamente l'uscita nel modo seguente:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

L'uscita deve corrispondere al seguente prototipo di funzione C:

```
typedef void MQENTRY AX_UNREG_EXIT (
     PMQAXP  pExitParms,   /* Address of exit parameter structure */
     PMQAXC  pExitContext, /* Address of exit context structure */
     PMQLONG pRmid,        /* Address of resource manager identifier */
     PMQLONG pFlags,       /* Address of resource manager options*/
     PMQLONG pXARetCode);  /* Address of response from XA call */
```

Informazioni generali sul richiamo delle funzioni di uscita

Questo argomento fornisce una guida generale che consente di pianificare le uscite, in particolare per quanto riguarda la gestione degli errori e degli eventi imprevisti.

Errore di chiusura

Se una funzione di uscita termina in modo anomalo dopo una chiamata MQGET distruttiva, fuori dal punto di sincronizzazione, ma prima che il messaggio sia stato passato all'applicazione, il gestore di uscita può eseguire il ripristino dall'errore e passare il controllo all'applicazione.

In questo caso, il messaggio potrebbe essere perso. Ciò è simile a quello che accade quando un'applicazione ha esito negativo immediatamente dopo aver ricevuto un messaggio da una coda.

La chiamata MQGET potrebbe essere completata con MQCC_FAILED e MQRC_API_EXIT_ERROR.

Se una funzione di uscita della chiamata API *prima* termina in modo anomalo, il gestore di uscita può eseguire il ripristino dall'errore e passare il controllo all'applicazione senza elaborare la chiamata API. In questo caso, la funzione di uscita deve recuperare tutte le risorse di cui è proprietaria.

Se le uscite concatenate sono in uso, le uscite di chiamata API *dopo* per qualsiasi *prima delle uscite di chiamata API* che sono state guidate correttamente possono essere guidate. La chiamata API potrebbe avere esito negativo con MQCC_FAILED e MQRC_API_EXIT_ERROR.

Esempio di gestione degli errori per le funzioni di uscita

Il seguente diagramma mostra i punti (e *N*) in cui possono verificarsi errori. È solo un esempio per mostrare come si comportano le uscite e deve essere letto insieme alla seguente tabella. In questo esempio, vengono richiamate due funzioni di uscita sia prima che dopo ogni chiamata API per mostrare il comportamento con uscite concatenate.

```
Application ErrPt     Exit function            API call
----------- -----     -------------            --------

Start

MQCONN  -->
            e1
                          MQ_INIT_EXIT
            e2
                    before MQ_CONNX_EXIT  1
            e3
                    before MQ_CONNX_EXIT  2
            e4
                                              -->  MQCONN
            e5
                    after  MQ_CONNX_EXIT  2
            e6
                    after  MQ_CONNX_EXIT  1
            e7
        <--
MQOPEN  -->
                    before MQ_OPEN_EXIT  1
            e8
                    before MQ_OPEN_EXIT  2
            e9
                                              -->  MQOPEN
            e10
                    after  MQ_OPEN_EXIT  2
            e11
                    after  MQ_OPEN_EXIT  1
            e12
        <--
MQPUT   -->
                    before MQ_PUT_EXIT   1
            e13
                    before MQ_PUT_EXIT   2
            e14
                                              -->  MQPUT
            e15
                    after  MQ_PUT_EXIT   2
            e16
                    after  MQ_PUT_EXIT   1
            e17
        <--
MQCLOSE -->
                    before MQ_CLOSE_EXIT 1
            e18
                    before MQ_CLOSE_EXIT 2
            e19
                                              -->  MQCLOSE
            e20
                    after  MQ_CLOSE_EXIT 2
            e21
                    after  MQ_CLOSE_EXIT 1
            e22
        <--
MQDISC  -->
                    before MQ_DISC_EXIT  1
            e23
                    before MQ_DISC_EXIT  2
            e24
                                              -->  MQDISC
            e25
                    after  MQ_DISC_EXIT  2
            e26
                    after  MQ_DISC_EXIT  1
            e27

        <--

end
```

La seguente tabella elenca le azioni da intraprendere in ogni punto di errore. Solo un sottoinsieme dei punti di errore è stato coperto, poiché le regole qui mostrate possono essere applicate a tutti gli altri. Sono le azioni che specificano il comportamento previsto in ogni caso.

Tabella 837. Errori di uscita API e azioni appropriate da intraprendere

Err Pt	**Descrizione**	**Azioni**
e1	Errore durante l'impostazione dell'ambiente.	1. Annulla l'impostazione dell'ambiente come richiesto 2. Nessuna funzione di uscita unità 3. Errore MQCONN con MQCC_FAILED, MQRC_API_EXIT_LOAD_ERROR
e2	La funzione MQ_INIT_EXIT viene completata con: • MQXCC_NON RIUSCITO • MQXCC_ *	• Per MQXCC_FAILED: 1. Ripulisci ambiente 2. Errore MQCONN con MQCC_FAILED, MQRC_API_EXIT_INIT_ERROR • Per MQXCC_ * 1. Agire come per i valori di MQXCC_ * e MQXR2_ * [1] 2. Ripulisci ambiente
e3	*Prima* La funzione MQ_CONNX_EXIT 1 viene completata con: • MQXCC_NON RIUSCITO • MQXCC_ *	• Per MQXCC_FAILED: 1. Funzione MQ_TERM_EXIT unità 2. Ripulisci ambiente 3. Chiamata MQCONN non riuscita con MQCC_FAILED, MQRC_API_EXIT_ERROR • Per MQXCC_ * 1. Agire come per i valori di MQXCC_ * e MQXR2_ * [1] 2. Se necessario, eseguire la funzione MQ_TERM_EXIT 3. Ripulire l'ambiente, se necessario
e4	*Prima* La funzione MQ_CONNX_EXIT 2 viene completata con: • MQXCC_NON RIUSCITO • MQXCC_ *	• Per MQXCC_FAILED: 1. Unità *dopo la funzione* MQ_CONNX_EXIT 1 2. Funzione MQ_TERM_EXIT unità 3. Ripulisci ambiente 4. Chiamata MQCONN non riuscita con MQCC_FAILED, MQRC_API_EXIT_ERROR • Per MQXCC_ * 1. Agire come per i valori di MQXCC_ * e MQXR2_ * [1] 2. Guidare *dopo la funzione* MQ_CONNX_EXIT 1 se l'uscita non viene eliminata 3. Se necessario, eseguire la funzione MQ_TERM_EXIT 4. Ripulire l'ambiente, se necessario

Tabella 837. Errori di uscita API e azioni appropriate da intraprendere (Continua)

Err Pt	Descrizione	Azioni
e5	La chiamata MQCONN non riesce.	1. Passa MQCONN CompCode e Motivo 2. Drive *dopo la funzione* MQ_CONNX_EXIT 2 se il *precedente a* MQ_CONNX_EXIT 2 ha avuto esito positivo e l'uscita non viene eliminata 3. Guidare *dopo* la funzione MQ_CONNX_EXIT 1 se *prima di* MQ_CONNX_EXIT 1 ha avuto esito positivo e l'uscita non viene eliminata 4. Funzione MQ_TERM_EXIT unità 5. Ripulisci ambiente
e6	*Dopo* la funzione MQ_CONNX_EXIT 2 viene completata con: • MQXCC_NON RIUSCITO • MQXCC_ *	• Per MQXCC_FAILED: 1. Unità *dopo la funzione* MQ_CONNX_EXIT 1 2. Completa chiamata MQCONN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Per MQXCC_ * 1. Agire come per i valori di MQXCC_ * e MQXR2_ * [1] 2. Guidare *dopo* la funzione MQ_CONNX_EXIT 1, se richiesto
e7	*Dopo* MQ_CONNX_EXIT 1 funzione viene completata con: • MQXCC_NON RIUSCITO • MQXCC_ *	• Per MQXCC_FAILED, completare la chiamata MQCONN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Per MQXCC_ *, agire come per i valori MQXCC_ * e MQXR2_ * [1]
e8	*Prima* La funzione MQ_OPEN_EXIT 1 viene completata con: • MQXCC_NON RIUSCITO • MQXCC_ *	• Per MQXCC_FAILED, completare la chiamata MQOPENcon MQCC_FAILED, MQRC_API_EXIT_ERROR • Per MQXCC_ *, agire come per i valori MQXCC_ * e MQXR2_ * [1]
e9	*Prima* La funzione MQ_OPEN_EXIT 2 viene completata con: • MQXCC_NON RIUSCITO • MQXCC_ *	• Per MQXCC_FAILED: 1. Guidare *dopo* la funzione MQ_OPEN_EXIT 1 2. Completa chiamata MQOPEN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Per MQXCC_ *, agire come per i valori MQXCC_ * e MQXR2_ * [1]
e10	Chiamata MQOPEN non riuscita	1. Passa MQOPEN CompCode e il motivo 2. Guidare *dopo* la funzione MQ_OPEN_EXIT 2 se l'uscita non viene eliminata 3. Guidare *dopo* la funzione MQ_OPEN_EXIT 1 se l'uscita non è stata eliminata e se le uscite concatenate non sono state eliminate

Tabella 837. Errori di uscita API e azioni appropriate da intraprendere (Continua)		
Err Pt	**Descrizione**	**Azioni**
e1 1	*Dopo* la funzione MQ_OPEN_EXIT 2 viene completata con: • MQXCC_NON RIUSCITO • MQXCC_ *	• Per MQXCC_FAILED: 1. Guidare *dopo* la funzione MQ_OPEN_EXIT 1 2. Completa chiamata MQOPEN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Per MQXCC_ * 1. Agire come per i valori di MQXCC_ * e MQXR2_ * [1] 2. Guidare *dopo* la funzione MQ_OPEN_EXIT 1 se l'uscita non viene eliminata
e2 5	*Dopo* la funzione MQ_DISC_EXIT 2 viene completata con: • MQXCC_NON RIUSCITO • MQXCC_ *	• Per MQXCC_FAILED: 1. Guidare *dopo la funzione* MQ_DISC_EXIT 1 2. Funzione MQ_TERM_EXIT unità 3. Ripulisci ambiente di esecuzione di uscita 4. Completa chiamata MQDISC con MQCC_FAILED, MQRC_API_EXIT_ERROR • Per MQXCC_ * 1. Agire come per i valori di MQXCC_ * e MQXR2_ * [1] 2. Funzione MQ_TERM_EXIT unità 3. Ripulisci ambiente di esecuzione di uscita

Nota:

1. I valori di MQXCC_ * e MQXR2_* e le azioni corrispondenti sono definiti in Modalità di elaborazione delle funzioni di uscita dei gestori code.

Campi ExitResponse impostati in modo non corretto

Questo argomento fornisce informazioni su cosa accade quando il campo ExitResponse è impostato su valori non supportati.

Se il campo ExitResponse è impostato su un valore diverso da uno dei valori supportati, si applicano le seguenti azioni:

- Per una funzione API exit MQCONN o MQDISC *precedente a* :
  - Il valore ExitResponse2 viene ignorato.
  - Non vengono richiamate ulteriori funzioni di uscita *prima* nella catena di uscita (se presenti); la chiamata API non viene emessa.
  - Per tutte le uscite *prima* che sono state richiamate correttamente, le uscite *dopo* vengono richiamate in ordine inverso.
  - Se registrate, le funzioni di uscita di terminazione per le funzioni di uscita *precedenti* MQCONN o MQDISC nella catena richiamate correttamente vengono ripulite dopo queste funzioni di uscita.
  - La chiamata MQCONN o MQDISC ha esito negativo con MQRC_API_EXIT_ERROR.
- Per una funzione API exit *before* IBM MQ diversa da MQCONN o MQDISC:
  - Il valore ExitResponse2 viene ignorato.
  - Non vengono richiamate ulteriori funzioni di conversione dati *prima di* o *dopo* nella catena di uscita (se presente).

  - Per tutte le uscite *prima* che sono state richiamate correttamente, le uscite *dopo* vengono richiamate in ordine inverso.
  - La chiamata API IBM MQ non viene emessa.
  - La chiamata API IBM MQ ha esito negativo con MQRC_API_EXIT_ERROR.
- Per una funzione di uscita API *dopo* MQCONN o MQDISC:
  - Il valore ExitResponse2 viene ignorato.
  - Le restanti funzioni di uscita che sono state richiamati correttamente prima della chiamata API vengono richiamate in ordine inverso.
  - Se registrato, le funzioni di uscita di terminazione per quelle *prima di* o *dopo* le funzioni di uscita MQCONN o MQDISC nella catena che sono state richiamate correttamente vengono ripulite dopo l'uscita.
  - Un CompCode del più grave di MQCC_WARNING e il CompCode restituito dall'uscita viene restituito all'applicazione.
  - Un motivo di MQRC_API_EXIT_ERROR viene restituito all'applicazione.
  - La chiamata API IBM MQ è stata emessa correttamente.
- Per una funzione di uscita chiamata API *dopo* IBM MQ diversa da MQCONN o MQDISC:
  - Il valore ExitResponse2 viene ignorato.
  - Le restanti funzioni di uscita che sono state richiamati correttamente prima della chiamata API vengono richiamate in ordine inverso.
  - Un CompCode del più grave di MQCC_WARNING e il CompCode restituito dall'uscita viene restituito all'applicazione.
  - Un motivo di MQRC_API_EXIT_ERROR viene restituito all'applicazione.
  - La chiamata API IBM MQ è stata emessa correttamente.
- Per la funzione di conversione dati *prima di* all'uscita get:
  - Il valore ExitResponse2 viene ignorato.
  - Le restanti funzioni di uscita che sono state richiamati correttamente prima della chiamata API vengono richiamate in ordine inverso.
  - Il messaggio non viene convertito e il messaggio non convertito viene restituito all'applicazione.
  - Un CompCode del più grave di MQCC_WARNING e il CompCode restituito dall'uscita viene restituito all'applicazione.
  - Un motivo di MQRC_API_EXIT_ERROR viene restituito all'applicazione.
  - La chiamata API IBM MQ è stata emessa correttamente.

  Nota: Poiché l'errore si verifica con l'uscita, è preferibile restituire MQRC_API_EXIT_ERROR piuttosto che restituire MQRC_NOT_CONVERTED.

Se una funzione exit imposta il campo ExitResponse2 su un valore diverso da uno dei valori supportati, viene utilizzato il valore MQXR2_DEFAULT_CONTINUATION .

Informazioni di riferimento per l'interfaccia dei servizi installabili

Questa raccolta di argomenti fornisce informazioni di riferimento per i servizi installabili.

Le funzioni e i tipi di dati sono elencati in ordine alfabetico all'interno del gruppo per ciascun tipo di servizio.

Concetti correlati

ALW Servizi e componenti installabili per Unix, Linux e Windows

IBM i Componenti e servizi installabili per IBM i

IBM i Informazioni di riferimento dell'interfaccia dei servizi installabili per IBMi

Attività correlate

Estensione delle funzioni del gestore code

ALW Configurazione dei servizi installabili

Modalità di visualizzazione delle funzioni

Come vengono documentate le funzioni dei servizi installabili.

Per ogni funzione è presente una descrizione, incluso l'identificativo della funzione (per MQZEP).

I *parametri* sono elencati nell'ordine in cui devono verificarsi. Devono essere tutti presenti.

Ogni nome parametro è seguito dal relativo tipo di dati. Questi sono i tipi di dati elementari descritti in "Tipi di dati elementari" a pagina 236.

Il richiamo del linguaggio C viene fornito anche dopo la descrizione dei parametri.

MQZ_AUTHENTICATE_USER - Autentica utente

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_5 e viene richiamata da un gestore code per autenticare un utente o per impostare i campi del contesto di identità. Viene richiamato quando viene stabilito il contesto dell'applicazione utente IBM MQ .

Il contesto dell'applicazione viene stabilito durante le chiamate di connessione nel punto in cui viene inizializzato il contesto utente dell'applicazione e in ogni punto in cui viene modificato il contesto utente dell'applicazione. Ogni volta che viene effettuata una chiamata di connessione, le informazioni di contesto utente dell'applicazione vengono riacquisite nel campo *IdentityContext* .

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_AUTHENTICATE_USER.

Sintassi

MQZ_AUTHENTICATE_USER (*QMgrName* , *SecurityParms* , *ApplicationContext* , *IdentityContext* , *CorrelationPtr* , *ComponentData* , *Continuazione* , *CompCode* , *Motivo*)

Parametri

QMgrName

Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

SecurityParms

Tipo: MQCSP - input

Parametri di sicurezza. Dati relativi all'ID utente, alla password e al tipo di autenticazione. Se l'attributo AuthenticationType della struttura MQCSP viene specificato come MQCSP_AUTH_USER_ID_AND_PWD, sia l'ID utente che la password vengono confrontati con i campi equivalenti nel parametro IdentityContext (MQZIC) per determinare se corrispondono. Per ulteriori informazioni, consultare "MQCSP - Parametri di sicurezza" a pagina 340.

Durante una chiamata MQCONN MQI questo parametro contiene valori null o predefiniti.

ApplicationContext

Tipo: MQZAC - input

Contesto dell'applicazione. Dati relativi all'applicazione chiamante. Per i dettagli, consultare MQZAC - Contesto applicazione .

Durante ogni chiamata MQCONN o MQCONNX MQI, le informazioni di contesto utente nella struttura MQZAC vengono riacquisite.

IdentityContext
Tipo: MQZIC - input/output

Contesto identità. Nell'input della funzione di autenticazione utente, identifica il contesto di identità corrente. La funzione di autenticazione utente può modificare questa condizione, a quel punto il gestore code adotta il nuovo contesto di identità. Per ulteriori dettagli sulla struttura MQZIC, consultare MQZIC - Contesto identità .

CorrelationPtr
Tipo: MQPTR - output

Puntatore di correlazione. Specifica l'indirizzo dei dati di correlazione. Questo puntatore viene successivamente passato ad altre chiamate OAM.

ComponentData
Tipo: MQBYTE x ComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una delle funzioni fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una delle funzioni di questo componente.

La lunghezza di questa area di dati viene passata dal gestore code nel parametro di lunghezza ComponentDatadella chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente da altri componenti.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

Per ulteriori informazioni su questi codici di errore, consultare Messaggi e codici di errore.

Richiamo C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                   IdentityContext, &CorrelationPtr, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Dichiarare i parametri passati al servizio nel modo seguente:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQCSP     SecurityParms;      /* Security parameters */
MQZAC     ApplicationContext; /* Application context */
MQZIC     IdentityContext;    /* Identity context */
MQPTR     CorrelationPtr;     /* Correlation pointer */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                                 component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_CHECK_AUTHORITY - Verifica autorizzazione

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_1 e viene avviata dal gestore code per verificare se un'entità dispone dell'autorità per eseguire una o più azioni specifiche su un oggetto specificato.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_CHECK_AUTHORITY.

Sintassi

MQZ_CHECK_AUTHORITY(*QMgrName* , *EntityName* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

EntityName
Tipo: MQCHAR12 - input

Nome entità. Il nome dell'entità la cui autorizzazione all'oggetto deve essere controllata. La lunghezza massima della stringa è di 12 caratteri; se è più corta viene riempita a destra con spazi. Il nome non termina con un carattere null.

Non è essenziale che questa entità sia nota al servizio di sicurezza sottostante. Se non è noto, per il controllo vengono utilizzate le autorizzazioni del gruppo speciale **nobody** (a cui si presume appartengano tutte le entità). Un nome completamente vuoto è valido e può essere utilizzato in questo modo.

EntityType
Tipo: MQLONG - input

Tipo di entità. Il tipo di entità specificata da EntityName. Deve essere uno dei seguenti valori:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

ObjectName
Tipo: MQCHAR48 - input

Il nome dell'oggetto. Il nome dell'oggetto a cui è richiesto l'accesso. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

Se *ObjectType* è MQOT_Q_MGR, questo nome è uguale a *QMgrName*.

ObjectType
Tipo: MQLONG - input

Tipo di oggetto. Il tipo di entità specificata da *ObjectName*. Deve essere uno dei seguenti valori:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

Autorizzazione
Tipo: MQLONG - input

Autorizzazione da controllare. Se viene controllata un'autorizzazione, questo campo è uguale all'operazione di autorizzazione appropriata (costante MQZAO_ *). Se viene controllata più di un'autorizzazione, è l'OR bit per bit delle costanti MQZAO_ * corrispondenti.

Le seguenti autorizzazioni si applicano all'utilizzo delle chiamate MQI:

CONNECT MQZAO_
Possibilità di utilizzare la chiamata MQCONN.

MQZAO_BROWSE
Possibilità di utilizzare la chiamata MQGET con un'opzione di esplorazione.

Ciò consente alle opzioni MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR o MQGMO_BROWSE_NEXT di essere specificate nella chiamata MQGET.

INPUT MQZAO_
Principale. Capacità di utilizzare la chiamata MQGET con un'opzione di input.

Ciò consente all'opzione MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE o MQOO_INPUT_AS_Q_DEF di essere specificata nella chiamata MQOPEN.

OUTPUT MQZAO_
Capacità di utilizzare la chiamata MQPUT.

Ciò consente di specificare l'opzione MQOO_OUTPUT nella chiamata MQOPEN.

INQUIRE MQZAO_
Possibilità di utilizzare la chiamata MQINQ.

Ciò consente di specificare l'opzione MQOO_INQUIRE nella chiamata MQOPEN.

MQZAO_SET
Possibilità di utilizzare la chiamata MQSET.

Ciò consente all'opzione MQOOO_SET di essere specificata nella chiamata MQOPEN.

Contesto MQZAO_PASS_IDENTITY_CONTEXT
Capacità di passare il contesto di identità.

Ciò consente di specificare l'opzione MQOO_PASS_IDENTITY_CONTEXT nella chiamata MQOPEN e l'opzione MQPMO_PASS_IDENTITY_CONTEXT nelle chiamate MQPUT e MQPUT1 .

MQZAO_PASS_ALL_CONTEXT
Capacità di passare tutto il contesto.

Ciò consente di specificare l'opzione MQOO_PASS_ALL_CONTEXT nella chiamata MQOPEN e l'opzione MQPMO_PASS_ALL_CONTEXT nelle chiamate MQPUT e MQPUT1 .

MQZAO_SET_IDENTITY_CONTEXT
Capacità di impostare il contesto di identità.

Ciò consente di specificare l'opzione MQOO_SET_IDENTITY_CONTEXT nella chiamata MQOPEN e l'opzione MQPMO_SET identity_context nelle chiamate MQPUT e MQPUT1 .

MQZAO_SET_ALL_CONTEXT
Possibilità di impostare tutto il contesto.

Ciò consente di specificare l'opzione MQOO_SET_ALL_CONTEXT nella chiamata MQOPEN e l'opzione MQPMO_SET_ALL_CONTEXT nelle chiamate MQPUT e MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY
Possibilità di utilizzare l'autorizzazione utente alternativa.

Ciò consente di specificare l'opzione MQOO_ALTERNATE_USER_AUTHORITY nella chiamata MQOPEN e l'opzione MQPMO_ALTERNATE_USER_AUTHORITY nella chiamata MQPUT1 .

MQZAO_ALL_MQI
Tutte le autorizzazioni MQI.

Ciò abilita tutte le autorizzazioni.

Le seguenti autorizzazioni si applicano alla gestione di un gestore code:

CREA_MQZAO_
Capacità di creare oggetti di un tipo specificato.

MQZAO_DELETE
Possibilità di eliminare un oggetto specificato.

DISPLAY MQZAO_
Possibilità di visualizzare gli attributi di un oggetto specificato.

MODIFICA_MQZO
Possibilità di modificare gli attributi di un oggetto specificato.

CLEAR MQZAO_
Possibilità di eliminare tutti i messaggi da una coda specificata.

MQZAO_AUTHORIZE
Possibilità di autorizzare altri utenti per un oggetto specificato.

CONTROL MQZAO_
Capacità di avviare o arrestare un listener, un servizio o un oggetto canale non client e la possibilità di eseguire il ping di un oggetto canale non client.

MQZAO_CONTROL_XX_ENCODE_CASE_ONE uscita
Possibilità di reimpostare un numero di sequenza o risolvere un messaggio in dubbio su un oggetto canale non client.

MQZAO_ALL_ADMIN
Capacità di impostare il contesto di identità.

Tutte le autorizzazioni di gestione, tranne MQZAO_CREATE.

Le seguenti autorizzazioni si applicano sia all'utilizzo di MQI che alla gestione di un gestore code:

MQZAO_ALL
Tutte le autorizzazioni, diverse da MQZAO_CREATE.

MQZAO_NONE
Nessuna autorizzazione.

ComponentData
Tipo: MQBYTE x ComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_CHECK_AUTHORITY, questo ha lo stesso effetto di MQZCI_STOP.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

Se la chiamata a un componente ha esito negativo (ovvero, *CompCode* restituisce MQCC_FAILED) e il parametro *Continuation* è MQZCI_DEFAULT o MQZCI_CONTINUE, il gestore code continua a richiamare altri componenti, se presenti.

Se la chiamata ha esito positivo (ovvero, *CompCode* restituisce MQCC_OK), nessun altro componente viene richiamato indipendentemente dall'impostazione di *Continuation* .

Se la chiamata ha esito negativo e il parametro *Continuazione* è MQZCI_STOP, non viene richiamato alcun altro componente e l'errore viene restituito al gestore code. I componenti non sono a conoscenza delle chiamate precedenti, quindi il parametro *Continuazione* è sempre impostato su MQZCI_DEFAULT prima della chiamata.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_CHECK_AUTHORITY_2 - Verifica autorizzazione (esteso)

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_2 e viene avviata dal gestore code per verificare se un'entità dispone dell'autorità per eseguire una o più azioni su un oggetto specificato.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_CHECK_AUTHORITY.

MQZ_CHECK_AUTHORITY_2 è come MQZ_CHECK_AUTHORITY, ma con il parametro **EntityName** sostituito dal parametro **EntityData** .

Sintassi

MQZ_CHECK_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

EntityData
Tipo: MQZED - input

Dati entità. I dati relativi all'entità con l'autorizzazione all'oggetto da controllare. Vedi “MQZED - Descrittore entità” a pagina 1724 per i dettagli.

Non è essenziale che questa entità sia nota al servizio di sicurezza sottostante. Se non è noto, per il controllo vengono utilizzate le autorizzazioni del gruppo speciale **nobody** (a cui si presume appartengano tutte le entità). Un nome completamente vuoto è valido e può essere utilizzato in questo modo.

EntityType
Tipo: MQLONG - input

Tipo di entità. Il tipo di entità specificata da *EntityData*. Deve essere uno dei seguenti valori:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

ObjectName
Tipo: MQCHAR48 - input

Il nome dell'oggetto. Il nome dell'oggetto a cui è richiesto l'accesso. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

Se *ObjectType* è MQOT_Q_MGR, questo nome è uguale a *QMgrName*.

ObjectType
Tipo: MQLONG - input

Tipo di oggetto. Il tipo di entità specificata da *ObjectName*. Deve essere uno dei seguenti valori:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

TOPIC MQOT_T
.

Autorizzazione
Tipo: MQLONG - input

Autorizzazione da controllare. Se viene controllata un'autorizzazione, questo campo è uguale all'operazione di autorizzazione appropriata (costante MQZAO_ *). Se viene controllata più di un'autorizzazione, è l'OR bit per bit delle costanti MQZAO_ * corrispondenti.

Le seguenti autorizzazioni si applicano all'utilizzo delle chiamate MQI:

CONNECT MQZAO_
Possibilità di utilizzare la chiamata MQCONN.

MQZAO_BROWSE
Possibilità di utilizzare la chiamata MQGET con un'opzione di esplorazione.

Ciò consente alle opzioni MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR o MQGMO_BROWSE_NEXT di essere specificate nella chiamata MQGET.

INPUT MQZAO_
Principale. Capacità di utilizzare la chiamata MQGET con un'opzione di input.

Ciò consente all'opzione MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE o MQOO_INPUT_AS_Q_DEF di essere specificata nella chiamata MQOPEN.

OUTPUT MQZAO_
Capacità di utilizzare la chiamata MQPUT.

Ciò consente di specificare l'opzione MQOO_OUTPUT nella chiamata MQOPEN.

INQUIRE MQZAO_
Possibilità di utilizzare la chiamata MQINQ.

Ciò consente di specificare l'opzione MQOO_INQUIRE nella chiamata MQOPEN.

MQZAO_SET
Possibilità di utilizzare la chiamata MQSET.

Ciò consente all'opzione MQOOO_SET di essere specificata nella chiamata MQOPEN.

Contesto MQZAO_PASS_IDENTITY_CONTEXT
Capacità di passare il contesto di identità.

Ciò consente di specificare l'opzione MQOO_PASS_IDENTITY_CONTEXT nella chiamata MQOPEN e l'opzione MQPMO_PASS_IDENTITY_CONTEXT nelle chiamate MQPUT e MQPUT1 .

MQZAO_PASS_ALL_CONTEXT
Capacità di passare tutto il contesto.

Ciò consente di specificare l'opzione MQOO_PASS_ALL_CONTEXT nella chiamata MQOPEN e l'opzione MQPMO_PASS_ALL_CONTEXT nelle chiamate MQPUT e MQPUT1 .

MQZAO_SET_IDENTITY_CONTEXT
Capacità di impostare il contesto di identità.

Ciò consente di specificare l'opzione MQOO_SET_IDENTITY_CONTEXT nella chiamata MQOPEN e l'opzione MQPMO_SET identity_context nelle chiamate MQPUT e MQPUT1 .

MQZAO_SET_ALL_CONTEXT
Possibilità di impostare tutto il contesto.

Ciò consente di specificare l'opzione MQOO_SET_ALL_CONTEXT nella chiamata MQOPEN e l'opzione MQPMO_SET_ALL_CONTEXT nelle chiamate MQPUT e MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY
Possibilità di utilizzare l'autorizzazione utente alternativa.

Ciò consente di specificare l'opzione MQOO_ALTERNATE_USER_AUTHORITY nella chiamata MQOPEN e l'opzione MQPMO_ALTERNATE_USER_AUTHORITY nella chiamata MQPUT1 .

MQZAO_ALL_MQI
Tutte le autorizzazioni MQI.

Ciò abilita tutte le autorizzazioni.

Le seguenti autorizzazioni si applicano alla gestione di un gestore code:

CREA_MQZAO_
Capacità di creare oggetti di un tipo specificato.

MQZAO_DELETE
Possibilità di eliminare un oggetto specificato.

DISPLAY MQZAO_
Possibilità di visualizzare gli attributi di un oggetto specificato.

MODIFICA_MQZO
Possibilità di modificare gli attributi di un oggetto specificato.

CLEAR MQZAO_
Possibilità di eliminare tutti i messaggi da una coda specificata.

MQZAO_AUTHORIZE
Possibilità di autorizzare altri utenti per un oggetto specificato.

CONTROL MQZAO_
Capacità di avviare o arrestare un listener, un servizio o un oggetto canale non client e la possibilità di eseguire il ping di un oggetto canale non client.

MQZAO_CONTROL_XX_ENCODE_CASE_ONE uscita
Possibilità di reimpostare un numero di sequenza o risolvere un messaggio in dubbio su un oggetto canale non client.

MQZAO_ALL_ADMIN
Capacità di impostare il contesto di identità.

Tutte le autorizzazioni di gestione, tranne MQZAO_CREATE.

Le seguenti autorizzazioni si applicano sia all'utilizzo di MQI che alla gestione di un gestore code:

MQZAO_ALL
Tutte le autorizzazioni, diverse da MQZAO_CREATE.

MQZAO_NONE
Nessuna autorizzazione.

ComponentData
Tipo: MQBYTE x ComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_CHECK_AUTHORITY, questo ha lo stesso effetto di MQZCI_STOP.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_CHECK_AUTHORITY_2 (QMgrName, &EntityData, EntityType,
                       ObjectName, ObjectType, Authority, ComponentData,
                       &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQZED     EntityData;        /* Entity data */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_CHECK_PRIVILEGED - Verifica se l'utente è privilegiato

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_6 e viene richiamata dal gestore code per stabilire se un utente specificato è un utente privilegiato.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_CHECK_PRIVILEGED.

Sintassi

MQZ_CHECK_PRIVILEGED(*QMgrName* , *EntityData* , *EntityType* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

EntityData
Tipo: MQZED - input

Dati entità. I dati relativi all'entità che deve essere controllata. Per ulteriori informazioni, consultare “MQZED - Descrittore entità” a pagina 1724.

EntityType
Tipo: MQLONG - input

Tipo di entità. Il tipo di entità specificato da EntityData. Deve essere uno dei seguenti valori:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

ComponentData
Tipo: MQBYTExComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_CHECK_AUTHORITY, questo ha lo stesso effetto di MQZCI_STOP.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

Se la chiamata a un componente ha esito negativo (ovvero, *CompCode* restituisce MQCC_FAILED) e il parametro *Continuation* è MQZCI_DEFAULT o MQZCI_CONTINUE, il gestore code continua a richiamare altri componenti, se presenti.

Se la chiamata ha esito positivo (ovvero, *CompCode* restituisce MQCC_OK), nessun altro componente viene richiamato indipendentemente dall'impostazione di *Continuation* .

Se la chiamata ha esito negativo e il parametro *Continuazione* è MQZCI_STOP, non viene richiamato alcun altro componente e l'errore viene restituito al gestore code. I componenti non sono a conoscenza delle chiamate precedenti, quindi il parametro *Continuazione* è sempre impostato su MQZCI_DEFAULT prima della chiamata.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_NOT_PRIVILEGED
(2584, X'A18') Questo non è un ID utente privilegiato.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entità sconosciuta al servizio.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,
                      ComponentData, &Continuation,
                      &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQZED     EntityData;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_COPY_ALL_AUTHORITY - Copia tutte le autorizzazioni

Questa funzione viene fornita da un componente del servizio di autorizzazione. Viene avviato dal gestore code per copiare tutte le autorizzazioni attualmente in vigore per un oggetto di riferimento in un altro oggetto.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_COPY_ALL_AUTHORITY.

Sintassi

MQZ_COPY_ALL_AUTHORITY(*QMgrName* , *RefObjectName* , *ObjectName* , *ObjectType* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

Nome RefObject
Tipo: MQCHAR48 - input

Nome oggetto di riferimento. Il nome dell'oggetto di riferimento, le cui autorizzazioni devono essere copiate. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

ObjectName
Tipo: MQCHAR48 - input

Il nome dell'oggetto. Il nome dell'oggetto per cui devono essere impostati gli accessi. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

ObjectType
Tipo: MQLONG - input

Tipo di oggetto. Il tipo di entità specificato da *RefObjectName* e *ObjectName*. Deve essere uno dei seguenti valori:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

TOPIC MQOT_T
.

ComponentData
Tipo: MQBYTExComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area di dati viene passata dal gestore code nel parametro di lunghezza ComponentDatadella chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_CHECK_AUTHORITY, questo ha lo stesso effetto di MQZCI_STOP.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

MQRC_UNKNOWN_REF_OBJECT
(2294, X'8F6') Oggetto di riferimento sconosciuto.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,
                        ComponentData, &Continuation, &CompCode,
                        &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  RefObjectName;     /* Reference object name */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_DELETE_AUTHORITY - Elimina autorizzazione

Questa funzione viene fornita da un componente del servizio di autorizzazione e viene avviata dal gestore code per eliminare tutte le autorizzazioni associate all'oggetto specificato.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_DELETE_AUTHORITY.

Sintassi

MQZ_DELETE_AUTHORITY(*QMgrName* , *ObjectName* , *ObjectType* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

ObjectName
Tipo: MQCHAR48 - input

Il nome dell'oggetto. Il nome dell'oggetto per il quale è necessario eliminare gli accessi. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

Se *ObjectType* è MQOT_Q_MGR, questo nome è uguale a *QMgrName*.

ObjectType
Tipo: MQLONG - input

Tipo di oggetto. Il tipo di entità specificata da *ObjectName*. Deve essere uno dei seguenti valori:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

TOPIC MQOT_T
.

ComponentData
Tipo: MQBYTE x ComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente

viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area di dati viene passata dal gestore code nel parametro di lunghezza ComponentDatadella chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_CHECK_AUTHORITY, questo ha lo stesso effetto di MQZCI_STOP.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,
                      &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_ENUMERATE_AUTHORITY_DATA - Enumerazione dei dati di autorizzazione

Questa funzione viene fornita da un componente servizio di autorizzazione MQZAS_VERSION_4 e viene avviata ripetutamente dal gestore code per richiamare tutti i dati di autorizzazione che corrispondono ai criteri di selezione specificati al primo richiamo.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_ENUMERATE_AUTHORITY_DATA.

Sintassi

MQZ_ENUMERATE_AUTHORITY_DATA(*QMgrName* , *StartEnumeration* , *Filter* , *AuthorityBufferLength* , *AuthorityBuffer* , *AuthorityDataLength* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

StartEnumeration
Tipo: MQLONG - input

Indicatore che indica se la chiamata può avviare l'enumerazione. Ciò indica se la chiamata può avviare l'enumerazione dei dati di autorizzazione o continuare l'enumerazione dei dati di autorizzazione avviati da una precedente chiamata a MQZ_ENUMERATE_AUTHORITY_DATA. Il valore è uno dei seguenti:

INIZIO_MQZSE
Avvia enumerazione. La chiamata viene avviata con questo valore per avviare l'enumerazione dei dati di autorizzazione. Il parametro **Filter** specifica i criteri di scelta da utilizzare per selezionare i dati di autorizzazione restituiti da questa e successive chiamate.

CONTINUE MQZSE
Continuare l'enumerazione. La chiamata viene avviata con questo valore per continuare l'enumerazione dei dati di autorizzazione. Il parametro **Filter** viene ignorato in questo caso e può essere specificato come puntatore null (i criteri di selezione sono determinati dal parametro **Filter** specificato dalla chiamata che aveva *StartEnumeration* impostato su MQZSE_START).

Filtro
Tipo: MQZAD - input

Filtro. Se *StartEnumeration* è MQZSE_START, *Filter* specifica i criteri di selezione da utilizzare per selezionare i dati di autorizzazione da restituire. Se *Filter* è il puntatore null, non viene utilizzato alcun criterio di selezione, ovvero, vengono restituiti tutti i dati di autorizzazione. Consultare "MQZAD - Dati di autorizzazione" a pagina 1721 per dettagli sui criteri di selezione che possono essere utilizzati.

Se *StartEnumeration* è MQZSE_CONTINUE, *Filter* viene ignorato e può essere specificato come puntatore null.

Lunghezza AuthorityBuffer
Tipo: MQLONG - input

Lunghezza di *AuthorityBuffer*. È la lunghezza in byte del parametro **AuthorityBuffer** . Il buffer di autorizzazione deve essere sufficientemente grande da contenere i dati da restituire.

AuthorityBuffer
Tipo: MQZAD - output

Dati di autorizzazione. Questo è il buffer in cui vengono restituiti i dati di autorizzazione. Il buffer deve essere abbastanza grande da contenere una struttura di MQZAD, una struttura MQZED, più il nome entità più lungo e il nome dominio più lungo definito.

Nota: Nota: questo parametro è definito come MQZAD, poiché MQZAD si verifica sempre all'avvio del buffer. Tuttavia, se il buffer è dichiarato come MQZAD, il buffer sarà troppo piccolo - deve essere più grande di MQZAD in modo che possa contenere i nomi di entità e dominio MQZAD, MQZED.

AuthorityDataLunghezza
Tipo: MQLONG - output

Lunghezza dei dati restituiti in *AuthorityBuffer*. Se il buffer di autorizzazione è troppo piccolo, *AuthorityDataLength* viene impostato sulla lunghezza del buffer richiesto e la chiamata restituisce il codice di completamento MQCC_FAILED e il codice motivo MQRC_BUFFER_LENGTH_ERROR.

ComponentData
Tipo: MQBYTE x ComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area di dati viene passata dal gestore code nel parametro di lunghezza ComponentDatadella chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_ENUMERATE_AUTHORITY_DATA, ha lo stesso effetto di MQZCI_CONTINUE.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE MQRC_BUFFER_LENGTH
(2005, X'7D5') Parametro di lunghezza del buffer non valido.

MQRC_NO_DATA_AVAILABLE
(2379, X'94B') Nessun dato disponibile.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,
                              AuthorityBufferLength,
                              &AuthorityBuffer,
                              &AuthorityDataLength, ComponentData,
                              &Continuation, &CompCode,
                              &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;               /* Queue manager name */
MQLONG    StartEnumeration;       /* Flag indicating whether call should
                                     start enumeration */
MQZAD     Filter;                 /* Filter */
MQLONG    AuthorityBufferLength;  /* Length of AuthorityBuffer */
MQZAD     AuthorityBuffer;        /* Authority data */
MQLONG    AuthorityDataLength;    /* Length of data returned in
                                     AuthorityBuffer */
MQBYTE    ComponentData[n];       /* Component data */
MQLONG    Continuation;           /* Continuation indicator set by
                                     component */
MQLONG    CompCode;               /* Completion code */
MQLONG    Reason;                 /* Reason code qualifying CompCode */
```

MQZ_FREE_USER - Utente libero

Questa funzione viene fornita da un componente del servizio di autorizzazione di MQZAS_VERSION_5 e viene avviata dal gestore code per liberare la risorsa assegnata associata.

Viene avviato quando un'applicazione ha terminato l'esecuzione in tutti i contesti utente, ad esempio durante una chiamata MQI MQDISC.

L'identificativo funzione per questa funzione (per MQZEP) è MQZID_FREE_USER.

Sintassi

MQZ_FREE_USER(*QMgrName* , *FreeParms* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

FreeParms
Tipo: MQZFP - input

Parametri liberi. Una struttura contenente i dati relativi alla risorsa da liberare. Vedi “MQZFP - Parametri liberi” a pagina 1727 per i dettagli.

ComponentData
Tipo: MQBYTE x ComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area di dati viene passata dal gestore code nel parametro di lunghezza ComponentDatadella chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente da altri componenti.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                   IdentityContext, CorrelationPtr, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQZFP     FreeParms;          /* Resource to be freed */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                                 component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_GET_AUTHORITY - Richiama autorizzazione

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_1 e viene avviata dal gestore code per richiamare l'autorizzazione di cui un'entità dispone per accedere all'oggetto specificato, incluse (se l'entità è un principal) le autorizzazioni possedute dai gruppi in cui il principal è un membro. Le autorizzazioni dai profili generici sono incluse nella serie di autorizzazione restituita.

L'identificativo funzione per questa funzione (per MQZEP) è MQZID_GET_AUTHORITY.

Sintassi

MQZ_GET_AUTHORITY(*QMgrName* , *EntityName* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

EntityName
Tipo: MQCHAR12 - input

Nome entità. Il nome dell'entità il cui accesso all'oggetto deve essere richiamato. La lunghezza massima della stringa è di 12 caratteri; se è più corta viene riempita a destra con spazi. Il nome non termina con un carattere null.

EntityType
Tipo: MQLONG - input

Tipo di entità. Il tipo di entità specificata da *EntityName*. Deve essere uno dei seguenti valori:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

ObjectName
Tipo: MQCHAR48 - input

Il nome dell'oggetto. Il nome dell'oggetto a cui deve essere richiamato l'accesso. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

Se *ObjectType* è MQOT_Q_MGR, questo nome è uguale a *QMgrName*.

ObjectType
Tipo: MQLONG - input

Tipo di oggetto. Il tipo di entità specificata da *ObjectName*. Deve essere uno dei seguenti valori:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

TOPIC MQOT_T
.

Autorizzazione
Tipo: MQLONG - input

Autorità dell'entità. Se l'entità dispone di un'autorizzazione, questo campo è uguale all'operazione di autorizzazione appropriata (costante MQZAO_ *). Se ha più di un'autorizzazione, questo campo è l'OR bit per bit delle costanti MQZAO_ * corrispondenti.

ComponentData
Tipo: MQBYTE ×ComponentDataLunghezza - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_GET_AUTHORITY, ha lo stesso effetto di MQZCI_CONTINUE.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entità sconosciuta al servizio.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, &Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_GET_AUTHORITY_2 - Richiama autorizzazione (esteso)

Questa funzione viene fornita da un componente del servizio di autorizzazione di MQZAS_VERSION_2 e viene avviata dal gestore code per richiamare l'autorizzazione di cui dispone un'entità per accedere all'oggetto specificato.

L'identificativo funzione per questa funzione (per MQZEP) è MQZID_GET_AUTHORITY.

MQZ_GET_AUTHORITY_2 è simile a MQZ_GET_AUTHORITY, ma con il parametro **EntityName** sostituito dal parametro **EntityData** .

Sintassi

MQZ_GET_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

EntityData
Tipo: MQZED - input

Dati entità. I dati relativi all'entità per cui deve essere richiamata l'autorizzazione all'oggetto. Vedi “MQZED - Descrittore entità” a pagina 1724 per i dettagli.

EntityType
Tipo: MQLONG - input

Tipo di entità. Il tipo di entità specificata da *EntityData*. Deve essere uno dei seguenti valori:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

ObjectName
Tipo: MQCHAR48 - input

Il nome dell'oggetto. Il nome dell'oggetto per il quale deve essere richiamata l'autorizzazione entità. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

Se *ObjectType* è MQOT_Q_MGR, questo nome è uguale a *QMgrName*.

ObjectType
Tipo: MQLONG - input

Tipo di oggetto. Il tipo di entità specificata da *ObjectName*. Deve essere uno dei seguenti valori:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

TOPIC MQOT_T
.

Autorizzazione
Tipo: MQLONG - input

Autorità dell'entità. Se l'entità dispone di un'autorizzazione, questo campo è uguale all'operazione di autorizzazione appropriata (costante MQZAO_ *). Se ha più di un'autorizzazione, questo campo è l'OR bit per bit delle costanti MQZAO_ * corrispondenti.

ComponentData
Tipo: MQBYTE ×ComponentDataLunghezza - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_CHECK_AUTHORITY, questo ha lo stesso effetto di MQZCI_STOP.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entità sconosciuta al servizio.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_GET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                     ObjectType, &Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQZED     EntityData;        /* Entity data */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_GET_EXPLICIT_AUTHORITY - Richiama autorizzazione esplicita

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_1 e viene avviata dal gestore code per richiamare l'autorizzazione di cui un'entità dispone per accedere all'oggetto specificato, incluse (se l'entità è un principal) le autorizzazioni possedute dai gruppi in cui il principal è un membro. Le autorizzazioni dai profili generici sono incluse nella serie di autorizzazione restituita.

Su AIX and Linux, per l'OAM (object authority manager) IBM MQ integrato, l'autorizzazione restituita è quella posseduta solo dal gruppo principale del principal.

L'identificativo funzione per questa funzione (per MQZEP) è MQZID_GET_EXPLICIT_AUTHORITY.

Sintassi

MQZ_GET_EXPLICIT_AUTHORITY(*QMgrName* , *EntityName* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

EntityName
Tipo: MQCHAR12 - input

Nome entità. Il nome dell'entità per cui deve essere richiamato l'accesso all'oggetto. La lunghezza massima della stringa è di 12 caratteri; se è più corta viene riempita a destra con spazi. Il nome non termina con un carattere null.

EntityType
Tipo: MQLONG - input

Tipo di entità. Il tipo di entità specificata da *EntityName*. Deve essere uno dei seguenti valori:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

ObjectName
Tipo: MQCHAR48 - input

Il nome dell'oggetto. Il nome dell'oggetto per il quale deve essere richiamata l'autorizzazione entità. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

Se *ObjectType* è MQOT_Q_MGR, questo nome è uguale a *QMgrName*.

ObjectType
Tipo: MQLONG - input

Tipo di oggetto. Il tipo di entità specificata da *ObjectName*. Deve essere uno dei seguenti valori:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

TOPIC MQOT_T
.

Autorizzazione
Tipo: MQLONG - input

Autorità dell'entità. Se l'entità dispone di un'autorizzazione, questo campo è uguale all'operazione di autorizzazione appropriata (costante MQZAO_ *). Se ha più di un'autorizzazione, questo campo è l'OR bit per bit delle costanti MQZAO_ * corrispondenti.

ComponentData
Tipo: MQBYTE x ComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_GET_AUTHORITY, ha lo stesso effetto di MQZCI_CONTINUE.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entità sconosciuta al servizio.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,
                            ObjectName, ObjectType, &Authority,
                            ComponentData, &Continuation,
                            &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_GET_EXPLICIT_AUTHORITY_2 - Richiama autorizzazione esplicita (estesa)

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_2 e viene avviata dal gestore code per richiamare l'autorità di cui un gruppo denominato dispone per accedere a un oggetto specificato (ma senza l'autorità aggiuntiva del gruppo **nobody**) o l'autorizzazione che il gruppo primario del principal denominato ha per accedere a un oggetto specifico.

L'identificativo funzione per questa funzione (per MQZEP) è MQZID_GET_EXPLICIT_AUTHORITY.

MQZ_GET_EXPLICIT_AUTHORITY_2 è simile a MQZ_GET_EXPLICIT_AUTHORITY, ma con il parametro **EntityName** sostituito dal parametro **EntityData** .

Sintassi

MQZ_GET_EXPLICIT_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

EntityData
Tipo: MQZED - input

Dati entità. I dati relativi all'entità la cui autorizzazione all'oggetto deve essere richiamata. Vedi “MQZED - Descrittore entità” a pagina 1724 per i dettagli.

EntityType
Tipo: MQLONG - input

Tipo di entità. Il tipo di entità specificata da *EntityData*. Deve essere uno dei seguenti valori:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

ObjectName
Tipo: MQCHAR48 - input

Il nome dell'oggetto. Il nome dell'oggetto per il quale deve essere richiamata l'autorizzazione entità. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

Se *ObjectType* è MQOT_Q_MGR, questo nome è uguale a *QMgrName*.

ObjectType
Tipo: MQLONG - input

Tipo di oggetto. Il tipo di entità specificata da *ObjectName*. Deve essere uno dei seguenti valori:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

TOPIC MQOT_T
.

Autorizzazione
Tipo: MQLONG - input

Autorità dell'entità. Se l'entità dispone di un'autorizzazione, questo campo è uguale all'operazione di autorizzazione appropriata (costante MQZAO_ *). Se ha più di un'autorizzazione, questo campo è l'OR bit per bit delle costanti MQZAO_ * corrispondenti.

ComponentData
Tipo: MQBYTE ×ComponentDataLunghezza - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_CHECK_AUTHORITY, questo ha lo stesso effetto di MQZCI_STOP.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entità sconosciuta al servizio.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_GET_EXPLICIT_AUTHORITY_2 (QMgrName, &EntityData, EntityType,
                              ObjectName, ObjectType, &Authority,
                              ComponentData, &Continuation,
                              &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQZED     EntityData;        /* Entity data */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_INIT_AUTHORITY - Inizializza servizio di autorizzazione

Questa funzionalità viene fornita da un componente del servizio di autorizzazione e viene avviata dal gestore code durante la sua configurazione. È previsto che richiami MQZEP per fornire informazioni al gestore code.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_INIT_AUTHORITY.

Sintassi

MQZ_INIT_AUTHORITY(*Hconfig* , *Options* , *QMgrName* , *ComponentDataLength* , *ComponentData* , *Version* , *CompCode* , *Reason*)

Parametri

Configurazione
Tipo: MQHCONFIG - input

Handle di configurazione. Questo handle rappresenta il particolare componente in fase di inizializzazione. Deve essere utilizzato dal componente quando si richiama il gestore code con la funzione MQZEP.

Opzioni
Tipo: MQLONG - input

Opzioni di inizializzazione. Deve essere uno dei seguenti valori:

MQZIO_PRIMARIO
Inizializzazione primaria.

MQZIO_SECONDARY
Inizializzazione secondaria.

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

ComponentDataLunghezza
Tipo: MQLONG - input

Lunghezza dei dati del componente. Lunghezza in byte dell'area *ComponentData* . Questa lunghezza è definita nei dati di configurazione del componente.

ComponentData
Tipo: MQBYTE x ComponentDataLength - input/output

Dati componente. Viene inizializzato su tutti zeri prima di richiamare la funzione di inizializzazione primaria del componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una qualsiasi delle funzioni (inclusa la funzione di inizializzazione) fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Versione
Tipo: MQLONG - input/output

Numero di versione. In fase di input per la funzione di inizializzazione, identifica il numero di versione più alto supportato dal gestore code. La funzione di inizializzazione deve modificarla, se necessario, nella versione dell'interfaccia che supporta. Se al ritorno il gestore code non supporta la versione restituita dal componente, richiama la funzione MQZ_TERM_AUTHORITY del componente e non utilizza più questo componente.

Sono supportati i seguenti valori:

MQZAS_VERSION_1
Versione 1.

MQZAS_VERSION_2
Versione 2.

MQZAS_VERSION_3
Versione 3.

MQZAS_VERSION_4
Versione 4.

MQZAS_VERSION_5
Versione 5.

MQZAS_VERSION_6
IBM WebSphere MQ 6.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_INITIALIZATION_FAILED
(2286, X'8EE') Inizializzazione non riuscita per un motivo indefinito.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,
                    ComponentData, &Version, &CompCode,
                    &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQHCONFIG  Hconfig;              /* Configuration handle */
MQLONG     Options;              /* Initialization options */
MQCHAR48   QMgrName;             /* Queue manager name */
MQLONG     ComponentDataLength;  /* Length of component data */
MQBYTE     ComponentData[n];     /* Component data */
MQLONG     Version;              /* Version number */
MQLONG     CompCode;             /* Completion code */
MQLONG     Reason;               /* Reason code qualifying CompCode */
```

MQZ_INQUIRE - Richiedi servizio di autorizzazione

Questa funzione è fornita da un componente del servizio di autorizzazione MQZAS_VERSION_5 e viene avviata dal gestore code per interrogare la funzionalità supportata.

Se vengono utilizzati più componenti di servizio, i componenti di servizio vengono richiamati in ordine inverso rispetto all'ordine in cui sono stati installati.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_INQUIRE.

Sintassi

MQZ_INQUIRE(*QMgrName* , *SelectorCount* , *Selectors* , *IntAttrCount* , *IntAttrs* , *CharAttrLength* , *CharAttrs* , *SelectorReturned* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

SelectorCount
Tipo: MQLONG - input

Numero di selettori. Il numero di selettori forniti nel parametro **Selectors** .

Il valore deve essere compreso tra 0 e 256.

Selettori
Tipo: MQLONGxSelectorCount - input

Array di selettori. Ciascun selettore identifica un attributo richiesto e deve essere uno dei seguenti:

- MQIACF_INTERFACE_VERSION (numero intero)
- MQIACF_USER_ID_SUPPORT (intero)
- MQCACF_SERVICE_COMPONENT (carattere)

I selettori possono essere specificati in qualsiasi ordine. Il numero di selettori nell'array è indicato dal parametro **SelectorCount** .

Gli attributi interi identificati dai selettori vengono restituiti nel parametro **IntAttrs** nello stesso ordine in cui vengono visualizzati in *Selectors*.

Gli attributi carattere identificati dai selettori vengono restituiti nel parametro **CharAttrs** nello stesso ordine in cui vengono visualizzati *Selectors*.

IntAttrCount
Tipo: MQLONG - input

Numero di attributi di numeri interi forniti nel parametro IntAttrs .

Il valore deve essere compreso tra 0 e 256.

IntAttrs
Tipo: Conteggio MQLONG x IntAttr- output

Attributi numero intero. Array di attributi integer. Gli attributi integer vengono restituiti nello stesso ordine dei corrispondenti selettori di numeri interi nell'array *Selectors* .

Conteggio CharAttr
Tipo: MQLONG - input

Lunghezza del buffer degli attributi carattere. La lunghezza in byte del parametro **CharAttrs** .

Il valore deve essere almeno la somma delle lunghezze degli attributi carattere richiesti. Se non è richiesto alcun attributo carattere, zero è un valore valido.

CharAttrs
Tipo: Conteggio MQLONG x CharAttr- output

Buffer attributi carattere. Buffer contenente attributi carattere, concatenati. Gli attributi del carattere vengono restituiti nello stesso ordine dei corrispondenti selettori di caratteri nell'array *Selectors* .

La lunghezza del buffer viene fornita dal parametro Conteggio CharAttr.

SelectorReturned
Tipo: MQLONG x SelectorCount - input

Selettore restituito. Array di valori che identificano quali attributi sono stati restituiti dalla serie richiesta dai selettori nel parametro Selettori. Il numero di valori in questo array è indicato dal parametro **SelectorCount** . Ogni valore nell'array si riferisce al selettore dalla posizione corrispondente nell'array Selettori. Ogni valore è uno dei seguenti:

MQZSL_RESTITUITO
L'attributo richiesto dal selettore corrispondente nel parametro **Selectors** è stato restituito.

MQZSL_NON_RESTITUITO
L'attributo richiesto dal selettore corrispondente nel parametro **Selectors** non è stato restituito.

L'array viene inizializzato con tutti i valori come *MQZSL_NOT_RETURNED*. Quando un componente del servizio di autorizzazione restituisce un attributo, imposta il valore appropriato nell'array su *MQZSL_NOT_RETURNED*. Ciò consente a qualsiasi altro componente del servizio di autorizzazione, a cui viene effettuata la chiamata di interrogazione, di identificare quali attributi sono già stati restituiti.

ComponentData
Tipo: MQBYTE x ComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_CHECK_AUTHORITY, questo ha lo stesso effetto di MQZCI_STOP.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT
Spazio insufficiente per gli attributi carattere.

MQRC_INT_COUNT_TOO_SMALL
Spazio insufficiente per gli attributi integer.

Se *CompCode* è MQCC_FAILED:

ERRORE MQRC_SELECTOR_COUNT
Il numero di selettori non è valido.

ERRORE DI MQRC_SELECTOR_ERROR
Selettore attributo non valido.

MQRC_SELECTOR_LIMIT_EXCEEDED
Sono stati specificati troppi selettori.

ERRORE MQRC_INT_ATTR_COUNT_
Il numero di attributi interi non è valido.

ERRORE - MQRC_INT_ATTRS_ARRAY_ERROR
Schiera attributi interi non valida.

MQRC_CHAR_ATTR_LENGTH_ERROR
Numero di attributi carattere non valido.

ERRORE MQRC_CHAR_ATTRS_
La stringa degli attributi carattere non è valida.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
                  &IntAttrs, CharAttrLength, &CharAttrs,
                  SelectorReturned, ComponentData, &Continuation,
                  &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48   QMgrName;              /* Queue manager name */
MQLONG     SelectorCount;         /* Selector count */
MQLONG     Selectors[n];          /* Selectors */
MQLONG     IntAttrCount;          /* IntAttrs count */
MQLONG     IntAttrs[n];           /* Integer attributes */
MQLONG     CharAttrCount;         /* CharAttrs count */
MQLONG     CharAttrs[n];          /* Chatacter attributes */
MQLONG     SelectorReturned[n];   /* Selector returned */
MQBYTE     ComponentData[n];      /* Component data */
MQLONG     Continuation;          /* Continuation indicator set by
                                     component */
MQLONG     CompCode;              /* Completion code */
MQLONG     Reason;                /* Reason code qualifying CompCode */
```

MQZ_REFRESH_CACHE - Aggiorna tutte le autorizzazioni

Questa funzione viene fornita da un componente di servizio di autorizzazione MQZAS_VERSION_3 e viene richiamata dal gestore code per aggiornare l'elenco di autorizzazioni conservate internamente dal componente.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_REFRESH_CACHE (8L).

Sintassi

MQZ_REFRESH_CACHE(*QMgrName* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene trasmesso al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

ComponentData
Tipo: MQBYTE ×ComponentDataLunghezza - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una delle funzioni fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una delle funzioni di questo componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_CHECK_AUTHORITY questo ha lo stesso effetto di MQZCI_STOP.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

Richiamo C

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_SET_AUTHORITY - Imposta autorizzazione

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_1 ed è avviata dal gestore code per impostare l'autorizzazione di cui dispone un'entità per accedere all'oggetto specificato.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_SET_AUTHORITY.

Nota: Questa funzione sostituisce tutte le autorizzazioni esistenti. Per conservare le autorizzazioni esistenti, è necessario impostarle nuovamente con questa funzione.

Sintassi

MQZ_SET_AUTHORITY(*QMgrName* , *EntityName* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

EntityName
Tipo: MQCHAR12 - input

Nome entità. Il nome dell'entità per cui deve essere richiamato l'accesso all'oggetto. La lunghezza massima della stringa è di 12 caratteri; se è più corta viene riempita a destra con spazi. Il nome non termina con un carattere null.

EntityType
Tipo: MQLONG - input

Tipo di entità. Il tipo di entità specificata da *EntityName*. Deve essere uno dei seguenti valori:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

ObjectName
Tipo: MQCHAR48 - input

Il nome dell'oggetto. Il nome dell'oggetto a cui è richiesto l'accesso. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

Se *ObjectType* è MQOT_Q_MGR, questo nome è uguale a *QMgrName*.

ObjectType
Tipo: MQLONG - input

Tipo di oggetto. Il tipo di entità specificata da *ObjectName*. Deve essere uno dei seguenti valori:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

TOPIC MQOT_T
.

Autorizzazione
Tipo: MQLONG - input

Autorità dell'entità. Se è stata impostata un'autorizzazione, questo campo è uguale all'operazione di autorizzazione appropriata (costante MQZAO_ *). Se viene impostata più di un'autorizzazione, questo campo è l'OR bit per bit delle costanti MQZAO_ * corrispondenti.

ComponentDatarname>
Tipo: MQBYTExComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_GET_AUTHORITY, ha lo stesso effetto di MQZCI_CONTINUE.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entità sconosciuta al servizio.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_SET_AUTHORITY_2 - Imposta autorizzazione (esteso)

Questa funzione viene fornita da un componente di servizio di autorizzazione MQZAS_VERSION_2 e viene avviata dal gestore code per impostare l'autorizzazione di cui dispone un'entità per accedere all'oggetto specificato.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_SET_AUTHORITY.

Nota: Questa funzione sostituisce tutte le autorizzazioni esistenti. Per conservare le autorizzazioni esistenti, è necessario impostarle nuovamente con questa funzione.

MQZ_SET_AUTHORITY_2 è come MQZ_SET_AUTHORITY, ma con il parametro **EntityName** sostituito dal parametro **EntityData** .

Sintassi

MQZ_SET_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

EntityData
Tipo: MQZED - input

Dati entità. Dati relativi all'entità la cui autorizzazione all'oggetto deve essere impostata. Vedi "MQZED - Descrittore entità" a pagina 1724 per i dettagli.

EntityType
Tipo: MQLONG - input

Tipo di entità. Il tipo di entità specificata da *EntityData*. Deve essere uno dei seguenti valori:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

ObjectName
Tipo: MQCHAR48 - input

Il nome dell'oggetto. Il nome dell'oggetto su cui deve essere impostata l'autorizzazione entità. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

Se *ObjectType* è MQOT_Q_MGR, questo nome è uguale a *QMgrName*.

ObjectType
Tipo: MQLONG - input

Tipo di oggetto. Il tipo di entità specificata da *ObjectName*. Deve essere uno dei seguenti valori:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

TOPIC MQOT_T
.

Autorizzazione
Tipo: MQLONG - input

Autorità dell'entità. Se è stata impostata un'autorizzazione, questo campo è uguale all'operazione di autorizzazione appropriata (costante MQZAO_ *). Se viene impostata più di un'autorizzazione, questo campo è l'OR bit per bit delle costanti MQZAO_ * corrispondenti.

ComponentData
Tipo: MQBYTE ×ComponentDataLunghezza - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_CHECK_AUTHORITY, questo ha lo stesso effetto di MQZCI_STOP.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entità sconosciuta al servizio.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_SET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQZED     EntityData;        /* Entity data */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_TERM_AUTHORITY - Termina servizio di autorizzazione

Questa funzione viene fornita da un componente del servizio di autorizzazione e viene avviata dal gestore code quando non richiede più i servizi di questo componente. La funzione deve eseguire qualsiasi ripulitura richiesta dal componente.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_TERM_AUTHORITY.

Sintassi

MQZ_TERM_AUTHORITY(*Hconfig* , *Options* , *QMgrName* , *ComponentData* , *CompCode* , *Reason*)

Parametri

Configurazione
Tipo: MQHCONFIG - input

Handle di configurazione. Questo handle rappresenta il componente particolare che viene terminato. Deve essere utilizzato dal componente quando si richiama il gestore code con la funzione MQZEP.

Opzioni
Tipo: MQLONG - input

Opzioni di terminazione. Deve essere uno dei seguenti valori:

MQZTO_PRIMARIO
Terminazione primaria.

MQZTO_SECONDARY
Terminazione secondaria.

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

ComponentData
Tipo: MQBYTE x ComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro di lunghezza ComponentDatasulla chiamata MQZ_INIT_AUTHORITY.

Una volta completata la chiamata MQZ_TERM_AUTHORITY, il gestore code elimina questi dati.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

MQRC_TERMINATION_FAILED
(2287, X'8FF') Terminazione non riuscita per un motivo non definito.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,
                    &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQHCONFIG  Hconfig;           /* Configuration handle */
MQLONG     Options;           /* Termination options */
MQCHAR48   QMgrName;          /* Queue manager name */
MQBYTE     ComponentData[n];  /* Component data */
MQLONG     CompCode;          /* Completion code */
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

MQZ_DELETE_NAME - Elimina nome

Questa funzione viene fornita da un componente servizio nomi e viene avviata dal gestore code per eliminare una voce per la coda specificata.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_DELETE_NAME.

Sintassi

MQZ_DELETE_NAME(*QMgrName* , *QName* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

QName
Tipo: MQCHAR48 - input

Il nome della coda. Il nome della coda per cui deve essere cancellata una voce. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

ComponentData
Tipo: MQBYTE x ComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro di lunghezza ComponentDatanella chiamata MQZ_INIT_NAME.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. Deve essere uno dei seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

STOP MQZCI
Non continuare con il componente successivo.

Per il comando **MQZ_DELETE_NAME** , il gestore code non tenta di avviare un altro componente, indipendentemente da quanto restituito nel parametro **Continuation** .

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Avvertenza (completamento parziale).

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

MQRC_UNKNOWN_NAME
(2288, X'8F0') Nome coda non trovato.

Nota: Potrebbe non essere possibile restituire questo codice se il servizio sottostante risponde con esito positivo per questo caso.

Se *CompCode* è MQCC_FAILED:

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_DELETE_NAME (QMgrName, QName, ComponentData, &Continuation,
                 &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  QName;             /* Queue name */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_INIT_NAME - Inizializza servizio nomi

Questa funzione viene fornita da un componente servizio nomi e viene avviata dal gestore code durante la configurazione del componente. È previsto che richiami MQZEP per fornire informazioni al gestore code.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_INIT_NAME.

Sintassi

MQZ_INIT_NAME(*Hconfig* , *Options* , *QMgrName* , *ComponentDataLength* , *ComponentData* , *Version* , *CompCode* , *Reason*)

Parametri

Configurazione
Tipo: MQHCONFIG - input

Handle di configurazione. Questo handle rappresenta il particolare componente in fase di inizializzazione. Deve essere utilizzato dal componente quando si richiama il gestore code con la funzione MQZEP.

Opzioni
Tipo: MQLONG - input

Opzioni di inizializzazione. Deve essere uno dei seguenti valori:

MQZIO_PRIMARIO
Inizializzazione primaria.

MQZIO_SECONDARY
Inizializzazione secondaria.

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

ComponentDataLunghezza
Tipo: MQLONG - input

Lunghezza dei dati del componente. Lunghezza in byte dell'area *ComponentData* . Questa lunghezza è definita nei dati di configurazione del componente.

ComponentData
Tipo: MQBYTE x ComponentDataLength - input/output

Dati componente. Viene inizializzato su tutti zeri prima di richiamare la funzione di inizializzazione primaria del componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una qualsiasi delle funzioni (inclusa la funzione di inizializzazione) fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Versione
Tipo: MQLONG - input/output

Numero di versione. In fase di input per la funzione di inizializzazione, identifica il numero di versione più alto supportato dal gestore code. La funzione di inizializzazione deve modificarla, se necessario, nella versione dell'interfaccia che supporta. Se al ritorno il gestore code non supporta la versione restituita dal componente, richiama la funzione MQZ_TERM_NAME del componente e non utilizza più questo componente.

Sono supportati i seguenti valori:

MQZAS_VERSION_1
Versione 1.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_INITIALIZATION_FAILED
(2286, X'8EE') Inizializzazione non riuscita per un motivo indefinito.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_INIT_NAME (Hconfig, Options, QMgrName, ComponentDataLength,
               ComponentData, &Version, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQHCONFIG  Hconfig;              /* Configuration handle */
MQLONG     Options;              /* Initialization options */
MQCHAR48   QMgrName;             /* Queue manager name */
MQLONG     ComponentDataLength;  /* Length of component data */
MQBYTE     ComponentData[n];     /* Component data */
MQLONG     Version;              /* Version number */
MQLONG     CompCode;             /* Completion code */
MQLONG     Reason;               /* Reason code qualifying CompCode */
```

MQZ_INSERT_NAME - Inserisci nome

Questa funzione viene fornita da un componente del servizio nomi e viene avviata dal gestore code per inserire una voce per la coda specificata, contenente il nome del gestore code proprietario della coda. Se la coda è già definita nel servizio, la chiamata ha esito negativo.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_INSERT_NAME.

Sintassi

MQZ_INSERT_NAME(*QMgrName* , *QName* , *ResolvedQMgrName* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName

Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

QName

Tipo: MQCHAR48 - input

Il nome della coda. Il nome della coda per cui è necessario inserire una voce. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Nome ResolvedQMgr

Tipo: MQCHAR48 - input

Nome gestore code risolto. Il nome del gestore code in cui viene risolta la coda. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

ComponentData

Tipo: MQBYTE ×ComponentDataLunghezza - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una qualsiasi delle funzioni (inclusa la funzione di inizializzazione) fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_NAME.

Continuazione
Tipo: MQLONG - input/output

Indicatore di continuazione impostato per componente. Per MQZ_INSERT_NAME, il gestore code non tenta di avviare un altro componente, qualsiasi cosa venga restituita nel parametro **Continuation** .

Sono supportati i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_Q_ALREADY_EXISTS
(2290, X'8F2') L'oggetto coda esiste già.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_INSERT_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,
                 &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  QName;             /* Queue name */
MQCHAR48  ResolvedQMgrName;  /* Resolved queue manager name */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_LOOKUP_NAME - Nome ricerca

Questa funzione viene fornita da un componente servizio nomi e viene avviata dal gestore code per recuperare il nome del gestore code proprietario per una coda specificata.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_LOOKUP_NAME.

Sintassi

MQZ_LOOKUP_NAME(*QMgrName* , *QName* , *ResolvedQMgrName* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName

Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

QName

Tipo: MQCHAR48 - input

Il nome della coda. Il nome della coda per cui deve essere risolta una voce. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Nome ResolvedQMgr

Tipo: MQCHAR48 - output

Nome gestore code risolto. Se la funzione viene completata correttamente, questo è il nome del gestore code proprietario della coda.

Il nome restituito dal componente del servizio deve essere riempito a destra con spazi vuoti fino alla lunghezza completa del parametro; il nome non deve terminare con un carattere null o deve contenere spazi vuoti iniziali o incorporati.

ComponentData

Tipo: MQBYTExComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una qualsiasi delle funzioni (inclusa la funzione di inizializzazione) fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_NAME.

Continuazione

Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. Per MQZ_LOOKUP_NAME, il gestore code specifica se avviare un altro componente del servizio nomi, come segue:

- Se *CompCode* è MQCC_OK, non vengono avviati ulteriori componenti, indipendentemente dal valore restituito in *Continuazione*.
- Se *CompCode* non è MQCC_OK, viene avviato un ulteriore componente, a meno che *Continuation* non sia MQZCI_STOP.

Sono supportati i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

MQRC_UNKNOWN_Q_NAME
(2288, X'8F0') Nome coda non trovato.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_LOOKUP_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,
                 &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  QName;             /* Queue name */
MQCHAR48  ResolvedQMgrName;  /* Resolved queue manager name */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_TERM_NAME - Termina servizio nomi

Questa funzione viene fornita da un componente servizio nomi e viene avviata dal gestore code quando non richiede più i servizi di questo componente. La funzione deve eseguire qualsiasi ripulitura richiesta dal componente.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_TERM_NAME.

Sintassi

MQZ_TERM_NAME(*Hconfig* , *Options* , *QMgrName* , *ComponentData* , *CompCode* , *Reason*)

Parametri

Configurazione
Tipo: MQHCONFIG - input

Handle di configurazione. Questo handle rappresenta il componente particolare che viene terminato. Viene utilizzato dal componente durante la chiamata al gestore code con la funzione MQZEP.

Opzioni
Tipo: MQLONG - input

Opzioni di terminazione. Deve essere uno dei seguenti valori:

MQZTO_PRIMARIO
Terminazione primaria.

MQZTO_SECONDARY
Terminazione secondaria.

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

ComponentData
Tipo: MQBYTE x ComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una qualsiasi delle funzioni (inclusa la funzione di inizializzazione) fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una di queste funzioni del componente.

I dati del componente sono nella memoria condivisa accessibile a tutti i processi.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_NAME.

Una volta completata la chiamata MQZ_TERM_NAME, il gestore code elimina questi dati.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_TERMINATION_FAILED
(2287, X'8FF') Terminazione non riuscita per un motivo non definito.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_TERM_NAME (Hconfig, Options, QMgrName, ComponentData, &CompCode,
               &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQHCONFIG  Hconfig;           /* Configuration handle */
MQLONG     Options;           /* Termination options */
MQCHAR48   QMgrName;          /* Queue manager name */
MQBYTE     ComponentData[n];  /* Component data */
MQLONG     CompCode;          /* Completion code */
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

MQZAC - Contesto applicazione

La struttura MQZAC viene utilizzata nella chiamata MQZ_AUTHENTICATE_USER per il parametro *ApplicationContext* . Questo parametro specifica i dati relativi all'applicazione chiamante.

Tabella 1 riepiloga i campi nella struttura.

Tabella 838. Campi in MQZAC

Campo	Descrizione
StrucId	Identificativo struttura
Versione	Numero di versione della struttura
ProcessId	Identificativo del processo
ThreadId	Identificativo thread
ApplName	Nome applicazione
UserID	Identificativo utente
EffectiveUserID	Identificativo utente effettivo
Ambiente	Ambiente
CallerType	Tipo chiamante
AuthenticationType	Tipo di autenticazione
BindType	Tipo di collegamento

Campi

StrucId
Tipo: MQCHAR4 - input

Identificatore struttura. Il valore è il seguente:

ID_STRUC_MQZAC
Identificativo per la struttura del contesto dell'applicazione.

Per il linguaggi di programmazione C, viene definita anche la costante MQZAC_STRUC_ID_ARRAY, che ha lo stesso valore di MQZAC_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione
Tipo: MQLONG - input

Numero di versione della struttura. Il valore è il seguente:

MQZAC_VERSION_1
Struttura del contesto dell'applicazione Version-1 . La costante MQZAC_CURRENT_VERSION specifica il numero di versione della versione corrente.

ProcessId
Tipo: MQPID - input

Identificativo processo dell'applicazione.

ThreadId
Tipo: MQTID - input

Identificativo thread dell'applicazione.

ApplName
Tipo: MQCHAR28 - input

Nome dell'applicazione

UserID
Tipo: MQCHAR12 - input

Identificativo utente. Su AIX and Linux questo campo specifica l'ID utente reale dell'applicazione. Su Windows questo campo specifica l'ID utente dell'applicazione.

ID EffectiveUser
Tipo: MQCHAR12 - input

Identificativo utente effettivo. Su AIX and Linux questo campo specifica l'ID utente effettivo dell'applicazione. Su Windows questo campo è vuoto.

Ambiente
Tipo: MQLONG - input

Ambiente. Questo campo specifica l'ambiente da cui è stata effettuata la chiamata. Il campo è uno dei seguenti valori:

SERVER MQXE_COMMAND_
Server dei comandi

MQXE_MQSC
Interprete dei comandi **runmqsc**

MQXE_MCA
Agent canale dei messaggi MQXE_OTHER

MQXE_ALTRO
Ambiente non definito

CallerType
Tipo: MQLONG - input

Tipo di chiamante. Questo campo specifica il tipo di programma che ha effettuato la chiamata. Il campo è uno dei seguenti valori:

MQXACT_EXTERNAL
La chiamata è esterna al gestore code.

MQXACT_INTERNO
La chiamata è interna al gestore code.

AuthenticationType
Tipo: MQLONG - input

Tipo di autenticazione. Questo campo specifica il tipo di autenticazione da eseguire. Il campo è uno dei seguenti valori:

MQZAT_CONTESTO_INIZIALE
La chiamata di autenticazione è dovuta al contesto utente in fase di inizializzazione. Questo valore viene utilizzato durante una chiamata MQCONN o MQCONNX.

MQZAT_CONTESTO_MODIFICA
La chiamata di autenticazione è dovuta al fatto che il contesto utente è stato modificato. Questo valore viene utilizzato quando l'MCA modifica il contesto utente. Argomento principale: MQZAC -

BindType
Tipo: MQLONG - input

Tipo di bind. Questo campo specifica il tipo di collegamento in uso. Il campo è uno dei seguenti valori:

MQCNO_FASTPATH_BINDING
Collegamento Fastpath.

MQCNO_SHARED_BINDING
Binding condiviso.

MQCNO_ISOLATED_BINDING
Collegamento isolato.

Dichiarazione C

Dichiarare i campi della struttura come segue:

```
typedef struct tagMQZAC MQZAC;
struct tagMQZAC {
  MQCHAR4   StrucId;            /* Structure identifier */
  MQLONG    Version;            /* Structure version number */
  MQPID     ProcessId;          /* Process identifier */
  MQTID     ThreadId;           /* Thread identifier */
  MQCHAR28  ApplName;           /* Application name */
  MQCHAR12  UserID;             /* User identifier */
  MQCHAR12  EffectiveUserID;    /* Effective user identifier */
  MQLONG    Environment;        /* Environment */
  MQLONG    CallerType;         /* Caller type */
  MQLONG    AuthenticationType; /* Authentication type */
  MQLONG    BindType;           /* Bind type */
};
```

MQZAD - Dati di autorizzazione

La struttura di MQZAD viene utilizzata nella chiamata MQZ_ENUMERATE_AUTHORITY_DATA per due parametri, un input e un output.

Consultare “MQZ_ENUMERATE_AUTHORITY_DATA - Enumerazione dei dati di autorizzazione” a pagina 1683 per ulteriori informazioni sui parametri **Filter** e **AuthorityBuffer** :

- MQZAD viene utilizzato per il parametro **Filter** che viene immesso nella chiamata. Questo parametro specifica i criteri di scelta da utilizzare per selezionare i dati di autorizzazione restituiti dalla chiamata.
- MQZAD viene utilizzato anche per il parametro **AuthorityBuffer** che è l'output della chiamata. Questo parametro specifica le autorizzazioni per una combinazione di nome profilo, tipo di oggetto ed entità.

Tabella 1 riepiloga i campi nella struttura.

Tabella 839. Campi in MQZAD

Campo	Descrizione
StrucId	Identificativo struttura

Tabella 839. Campi in MQZAD (Continua)	
Campo	**Descrizione**
Versione	Numero di versione della struttura
ProfileName	Nome profilo
ObjectType	Tipo oggetto
Autorità	Autorizzazione
EntityDataptr	Puntatore ai dati di entità
EntityType	Tipo di entità
Opzioni	Opzioni

Campi

StrucId
Tipo: MQCHAR4 - input

Identificatore struttura. Il valore è il seguente:

ID_STRUC_MQZAD
Identificativo per la struttura dati di autorizzazione.

Per il linguaggio di programmazione C, viene definita anche la costante MQZAD_STRUC_ID_ARRAY; ha lo stesso valore di MQZAD_STRUC_ID, ma è una schiera di caratteri invece di una stringa.

Versione
Tipo: MQLONG - input

Numero di versione della struttura. Il valore è il seguente:

MQZAD_VERSION_1
Struttura del contesto dell'applicazione Version-1 . La costante MQZAD_CURRENT_VERSION specifica il numero di versione della versione corrente.

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQZAD_CURRENT_
La versione corrente della struttura dati di autorizzazione.

ProfileName
Tipo: MQCHAR48 - input

Nome del profilo.

Per il parametro **Filter** , questo campo è il nome profilo per cui sono necessari i dati di autorizzazione. Se il nome è completamente vuoto fino alla fine del campo o al primo carattere null, vengono restituiti i dati di autorizzazione per tutti i nomi profilo.

Per il parametro **AuthorityBuffer** , questo campo è il nome di un profilo che corrisponde ai criteri di selezione specificati.

ObjectType
Tipo: MQLONG - input

Tipo di oggetto.

Per il parametro **Filter** , questo campo è il tipo di oggetto per cui sono richiesti i dati di autorizzazione. Se il valore è MQOT_ALL, vengono restituiti i dati di autorizzazione per tutti i tipi di oggetto.

Per il parametro **AuthorityBuffer** , questo campo è il tipo di oggetto a cui si applica il profilo identificato dal parametro **ProfileName** .

Il valore è uno dei seguenti; per il parametro **Filter** , è valido anche il valore MQOT_ALL:

INFO MQOT_AUTH_O
Informazioni di autenticazione

CANALIZZATA MQOT_
Canale

MQOT_CLNTCONN_CHALLEGATO
Canale connessione client

LISTENER MQOT_
Listener

ELENCO NOMI MQOTT
Elenco nomi

PROCESSO MQOT_
Definizione di processo

MQOT_Q
Coda

Gestore code MQOT_GR
Gestore code

SERVIZIO_MQT
Servizio

Autorizzazione
Tipo: MQLONG - input

Autorizzazione.

Per il parametro **Filter** , questo campo viene ignorato.

Per il parametro **AuthorityBuffer** , questo campo rappresenta le autorizzazioni che l'entità ha sugli oggetti identificati da **ProfileName** e **ObjectType**. Se l'entità dispone di una sola autorizzazione, il campo è uguale al valore di autorizzazione appropriato (costante MQZAO_ *). Se l'entità ha più di un'autorizzazione, il campo è l'OR bit per bit delle costanti MQZAO_ * corrispondenti.

Ptr EntityData
Tipo: PMQZED - input

Indirizzo della struttura MQZED che identifica un'entità.

Per il parametro **Filter** , questo campo indica una struttura MQZED che identifica l'entità per cui sono necessari i dati di autorizzazione. Se **EntityDataPtr** è il puntatore null, vengono restituiti i dati di autorizzazione per tutte le entità.

Per il parametro **AuthorityBuffer** , questo campo indica una struttura MQZED che identifica l'entità per cui sono stati restituiti i dati di autorizzazione.

EntityType
Tipo: MQLONG - input

Tipo di entità.

Per il parametro **Filter** , questo campo specifica il tipo di entità per cui sono necessari i dati di autorizzazione. Se il valore è MQZAET_NONE, vengono restituiti i dati di autorizzazione per tutti i tipi di entità.

Per il parametro **AuthorityBuffer** , questo campo specifica il tipo di entità identificato dalla struttura MQZED indicata dal parametro **EntityDataPtr** .

Il valore è uno dei seguenti; per il parametro **Filter** , è valido anche il valore MQZAET_NONE:

PRINCIPALE_MQZAET
Preside

GRUPPO_MQZ
Gruppo

Opzioni
Tipo: MQAUTHOPT - input

Opzioni. Questo campo specifica le opzioni che forniscono il controllo sui profili visualizzati. È necessario specificare uno dei seguenti valori:

MQAUTHOPT_NAME_ALL_MATCHING
Visualizza tutti i profili

MQAUTHOPT_NAME_EXPLICIT
Visualizza i profili che hanno esattamente lo stesso nome specificato nel campo **ProfileName** .

Inoltre, deve essere specificato anche uno dei seguenti:

MQAUTOPT_ENTITA_SET
Visualizzare tutti i profili utilizzati per calcolare l'autorizzazione cumulativa che l'entità ha sull'oggetto specificato dal parametro **ProfileName** . Il parametro **ProfileName** non deve contenere caratteri jolly.

- Se l'entità specificata è un principal, per ciascun membro della serie {entity, groups} viene visualizzato il profilo più applicabile che si applica all'oggetto.
- Se l'entità specificata è un gruppo, viene visualizzato il profilo più applicabile dal gruppo che si applica all'oggetto.
- Se questo valore viene specificato, i valori di **ProfileName**, **ObjectType**, **EntityType**e il nome entità specificato nella struttura MQZED **EntityDataPtr** , devono essere tutti non vuoti.

Se è stato specificato MQAUTHOPT_NAME_ALL_MATCHING, è possibile specificare anche il seguente valore:

MQAUTHOPT_ENTITY_EXPLICIT
Visualizza i profili che hanno esattamente lo stesso nome entità del nome entità specificato nella struttura **EntityDataPtr** MQZED.

Dichiarazione C

```
typedef struct tagMQZAD MQZAD;
struct tagMQZAD {
  MQCHAR4   StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQCHAR48  ProfileName;    /* Profile name */
  MQLONG    ObjectType;     /* Object type */
  MQLONG    Authority;      /* Authority */
  PMQZED    EntityDataPtr;  /* Address of MQZED structure identifying an
                               entity */
  MQLONG    EntityType;     /* Entity type */
  MQAUTHOPT Options;        /* Options */
};
```

MQZED - Descrittore entità

La struttura MQZED viene utilizzata in un numero di chiamate al servizio di autorizzazione per specificare l'entità per cui deve essere controllata l'autorizzazione.

Tabella 1 riepiloga i campi nella struttura.

Tabella 840. Campi in MQZED

Campo	Descrizione
StrucId	Identificativo struttura
Versione	Versione

Tabella 840. Campi in MQZED (Continua)	
Campo	**Descrizione**
EntityName Ptr	Nome entità
EntityDomainPtr	Puntatore dominio entità
SecurityId	Identificativo di sicurezza
CorrelationPtr	Puntatore di correlazione

Campi

StrucId
Tipo: MQCHAR4 - input

Identificatore struttura. Il valore è il seguente:

ID_STRUC_MQZ_
Identificativo per la struttura descrittore entità.

Per il linguaggio di programmazione C, è definita anche la costante MQZED_STRUC_ID_ARRAY, che ha lo stesso valore di MQZED_STRUC_ID, ma è un array di caratteri invece di una stringa.

Versione
Tipo: MQLONG - input

Numero di versione della struttura. Il valore è il seguente:

MQZED_VERSION_1
Struttura descrittore entità Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQZED_CURRENT_
La versione corrente della struttura del descrittore entità.

EntityNameptr
Tipo: PMQCHAR - input

Nome del profilo.

Indirizzo del nome entità. Questo è un puntatore al nome dell'entità la cui autorizzazione deve essere controllata.

Ptr EntityDomain
Tipo: PMQCHAR - input

Indirizzo del nome dominio entità. Questo è un puntatore al nome del dominio contenente la definizione dell'entità la cui autorizzazione deve essere controllata.

SecurityId
Tipo: MQBYTE40 - input

Autorizzazione.

Identificativo di sicurezza. Questo è l'identificativo di sicurezza la cui autorizzazione deve essere controllata.

CorrelationPtr
Tipo: MQPTR - input

Puntatore di correlazione. Ciò facilita il trasferimento dei dati di correlazione tra la funzione utente di autenticazione e altre funzioni OAM appropriate.

Dichiarazione C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  PMQCHAR   EntityNamePtr;    /* Address of entity name */
  PMQCHAR   EntityDomainPtr;  /* Address of entity domain name */
  MQBYTE40  SecurityId;       /* Security identifier */
  MQPTR     CorrelationPtr;   /* Address of correlation data */
```

MQZEP - Aggiungi punto di ingresso componente

Un componente di servizio avvia questa funzione, durante l'inizializzazione, per aggiungere un punto di ingresso al vettore del punto di ingresso per tale componente di servizio.

Sintassi

MQZEP (*Hconfig* , *Funzione* , *EntryPoint* , *CompCode* , *Motivo*)

Parametri

Configurazione
Tipo: MQHCONFIG - input

Handle di configurazione. Questo handle rappresenta il componente che viene configurato per questo particolare servizio installabile. Deve essere uguale al componente passato alla funzione di configurazione del componente dal gestore code sulla chiamata di inizializzazione del componente.

Funzione
Tipo: MQLONG - input

Identificativo funzione. I relativi valori validi sono definiti per ciascun servizio installabile.

Se MQZEP viene richiamato più di una volta per la stessa funzione, l'ultima chiamata effettuata fornisce il punto di ingresso utilizzato.

EntryPoint
Tipo: PMQFUNC - input

Punto di ingresso della funzione. Questo è l'indirizzo del punto di entrata fornito dal componente per eseguire la funzione.

Il valore NULL è valido e indica che la funzione non è fornita da questo componente. Viene assunto NULL per i punti di ingresso che non sono definiti utilizzando MQZEP.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE MQRC_FUNCTION_
(2281, X'8E9') Identificatore funzione non valido.

ERRORE MQRC_HCONFIG
(2280, X'8E8') Handle di configurazione non valido.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONFIG  Hconfig;     /* Configuration handle */
MQLONG     Function;    /* Function identifier */
PMQFUNC    EntryPoint;  /* Function entry point */
MQLONG     CompCode;    /* Completion code */
MQLONG     Reason;      /* Reason code qualifying CompCode */
```

MQZFP - Parametri liberi

La struttura MQZFP viene utilizzata nella chiamata MQZ_FREE_USER per il parametro *FreeParms* . Questo parametro specifica i dati relativi alla risorsa da liberare.

Tabella 1 riepiloga i campi nella struttura.

Tabella 841. Campi in MQZFP

Campo	Descrizione
StrucId	Identificativo struttura
Versione	Versione
riservato	Campo riservato
CorrelationPtr	Puntatore di correlazione

Campi

StrucId
Tipo: MQCHAR4 - input

Identificatore struttura. Il valore è il seguente:

ID_STRUC_MQZIC
Identificativo per la struttura del contesto identità. Per il linguaggio di programmazione C, viene definita anche la costante MQZIC_STRUC_ID_ARRAY, che ha lo stesso valore di MQZIC_STRUC_ID, ma è un array di caratteri anziché una stringa.

Versione
Tipo: MQLONG - input

Numero di versione della struttura. Il valore è il seguente:

MQZFP_VERSION_1
Struttura di parametri liberi Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQZFP_CURRENT_
La versione corrente della struttura dei parametri liberi.

Riservato
Tipo: MQBYTE8 - input

Campo riservato. Il valore iniziale è null.

CorrelationPtr
Tipo: MQPTR - input

Puntatore di correlazione. Indirizzo dei dati di correlazione relativi alla risorsa da liberare.

Dichiarazione C

```
typedef struct tagMQZFP MQZFP;
struct tagMQZFP {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  MQBYTE8   Reserved;         /* Reserved field */
  MQPTR     CorrelationPtr;   /* Address of correlation data */
};
```

MQZIC - Contesto identità

La struttura MQZIC viene utilizzata nella chiamata MQZ_AUTHENTICATE_USER per il parametro *IdentityContext* .

La struttura MQZIC contiene informazioni sul contesto identità, che identificano l'utente dell'applicazione che per primo ha inserito il messaggio su una coda:

- Il gestore code riempie il campo *UserIdentifier* con un nome che identifica l'utente, il modo in cui il gestore code può eseguire questa operazione dipende dall'ambiente in cui l'applicazione è in esecuzione.
- Il gestore code riempie il campo *AccountingToken* con un token o un numero determinato dall'applicazione che ha inserito il messaggio.
- Le applicazioni possono utilizzare il campo *ApplIdentityData* per qualsiasi informazione aggiuntiva che desiderano includere sull'utente (ad esempio, una password codificata).

Le applicazioni opportunamente autorizzate possono impostare il contesto di identità utilizzando la funzione MQZ_AUTHENTICATE_USER.

Un SID (systems security identifier) Windows viene memorizzato nel campo *AccountingToken* quando un messaggio viene creato in IBM MQ for Windows. Il SID può essere utilizzato per integrare il campo *UserIdentifier* e per stabilire le credenziali di un utente.

Tabella 1 riepiloga i campi nella struttura.

Tabella 842. Campi in MQZIC

Campo	Descrizione
StrucId	Identificativo struttura
Versione	Versione
UserIdentifier	Identificativo utente
AccountingToken	Token account
ApplIdentityData	Dati identità applicazione

Campi

StrucId
Tipo: MQCHAR4 - input

Identificatore struttura. Il valore è il seguente:

ID_STRUC_MQZIC
Identificativo per la struttura del contesto identità. Per il linguaggio di programmazione C, viene definita anche la costante MQZIC_STRUC_ID_ARRAY, che ha lo stesso valore di MQZIC_STRUC_ID, ma è un array di caratteri anziché una stringa.

Versione
Tipo: MQLONG - input

Numero di versione della struttura. Il valore è il seguente:

MQZIC_VERSION_1
Struttura del contesto di identità Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQZIC_CURRENT_
La versione corrente della struttura del contesto di identità.

UserIdentifier
Tipo: MQCHAR12 - input

Identificativo utente. Questa è una parte del contesto di identità del messaggio. *UserIdentifier* specifica l'ID utente dell'applicazione che ha creato il messaggio. Il gestore code considera queste informazioni come dati carattere, ma non ne definisce il formato. Per ulteriori informazioni sul campo *UserIdentifier* , consultare “UserIdentifier (MQCHAR12) per MQMD” a pagina 469.

AccountingToken
Tipo: MQBYTE32 - input

Token di account. Questa è una parte del contesto di identità del messaggio. *AccountingToken* consente a un'applicazione di far sì che il lavoro eseguito come risultato del messaggio venga addebitato in maniera appropriata. Il gestore code tratta queste informazioni come una stringa di bit e non ne controlla il contenuto. Per ulteriori informazioni sul campo *AccountingToken* , consultare “AccountingToken (MQBYTE32) per MQMD” a pagina 471.

ApplIdentityData
Tipo: MQCHAR32 - input

Dati dell'applicazione relativi all'identità. Questa è una parte del contesto di identità del messaggio. ApplIdentityI dati sono informazioni definite dalla suite di applicazioni che è possibile utilizzare per fornire ulteriori informazioni sull'origine del messaggio. Ad esempio, potrebbe essere impostato dalle applicazioni in esecuzione con l'autorizzazione utente appropriata per indicare se i dati di identità sono attendibili. Per ulteriori informazioni sul campo ApplIdentityData, consultare “Dati ApplIdentity(MQCHAR32) per MQMD” a pagina 472.

Dichiarazione C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  MQCHAR12  UserIdentifier;   /* User identifier */
  MQBYTE32  AccountingToken;  /* Accounting token */
  MQCHAR32  ApplIdentityData; /* Application data relating to identity */
};
```

IBM i Informazioni di riferimento sull'interfaccia dei servizi installabili su IBM i

Utilizzare queste informazioni per comprendere le informazioni di riferimento per i servizi installabili per IBM i.

Per ogni funzione è presente una descrizione, incluso l'identificativo della funzione (per MQZEP).

I *parametri* sono elencati nell'ordine in cui devono verificarsi. Devono essere tutti presenti.

Ogni nome parametro è seguito dal relativo tipo di dati tra parentesi. Questi sono i tipi di dati elementari descritti in “Tipi di dati elementari” a pagina 1024.

Il richiamo del linguaggio C viene fornito anche dopo la descrizione dei parametri.

Concetti correlati

IBM i Componenti e servizi installabili per IBM i

ALW Servizi e componenti installabili per Unix, Linux e Windows

Riferimenti correlati

“Informazioni di riferimento per l'interfaccia dei servizi installabili” a pagina 1665
Questa raccolta di argomenti fornisce informazioni di riferimento per i servizi installabili.

IBM i MQZEP (Add component entry point) su IBM i

Questa funzionalità viene richiamata da un componente di servizio, durante l'inizializzazione, per aggiungere un punto di ingresso al vettore del punto di ingresso per quel componente di servizio.

Sintassi

```
MQZEP (Hconfig, Function, EntryPoint, CompCode, Reason)
```

Parametri

La chiamata MQZEP ha i parametri seguenti.

Hconfig (MQHCONFIG) - input
Handle di configurazione.

Questo handle rappresenta il componente che viene configurato per questo particolare servizio installabile. Deve essere uguale a quello passato alla funzione di configurazione del componente dal gestore code sulla chiamata di inizializzazione del componente.

Funzione (MQLONG) - input
Identificativo funzione.

I relativi valori validi sono definiti per ciascun servizio installabile. Se MQZEP viene richiamato più di una volta per la stessa funzione, l'ultima chiamata effettuata fornisce il punto di ingresso utilizzato.

EntryPoint (PMQFUNC) - input
Punto di ingresso della funzione.

Questo è l'indirizzo del punto di entrata fornito dal componente per eseguire la funzione. Il valore NULL è valido e indica che la funzione non è fornita da questo componente. NULL viene assunto per i punti di ingresso che non sono definiti utilizzando MQZEP.

CompCode (MQLONG) - output
Codice di completamento.

Il valore è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo (MQLONG) - output
Codice di errore *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE MQRC_FUNCTION_
(2281, X'8E9') Identificatore funzione non valido.

ERRORE MQRC_HCONFIG
(2280, X'8E8') Handle di configurazione non valido.

Per ulteriori informazioni su questi codici di errore, consultare Messaggi e codici di errore.

Richiamo C

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQHCONFIG  Hconfig;     /* Configuration handle */
MQLONG     Function;    /* Function identifier */
PMQFUNC    EntryPoint;  /* Function entry point */
MQLONG     CompCode;    /* Completion code */
MQLONG     Reason;      /* Reason code qualifying CompCode */
```

IBM i MQHCONFIG (handle di configurazione) su IBM i

Il tipo di dati MQHCONFIG rappresenta un handle di configurazione, ossia il componente che viene configurato per un determinato servizio installabile. Un handle di configurazione deve essere allineato sul suo limite naturale.

Le applicazioni devono testare le variabili di questo tipo solo per l'uguaglianza.

Dichiarazione C

```
typedef void MQPOINTER MQHCONFIG;
```

IBM i PMQFUNC (Puntatore alla funzione) in IBM i

Puntatore a una funzione.

Dichiarazione C

```
typedef void MQPOINTER PMQFUNC;
```

IBM i MQZ_AUTHENTICATE_USER (Autenticazione utente) su IBM i

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_5 . Viene richiamato dal gestore code per autenticare un utente o per impostare i campi del contesto di identità.

Viene richiamato quando viene stabilito un contesto dell'applicazione utente IBM MQ . Ciò si verifica durante le chiamate di connessione nel punto in cui viene inizializzato il contesto utente dell'applicazione e in ogni punto in cui viene modificato il contesto utente dell'applicazione. Ogni volta che viene effettuata una chiamata di connessione, le informazioni di contesto utente dell'applicazione vengono riacquisite nel campo *IdentityContext* .

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_AUTHENTICATE_USER.

Sintassi

UTENTE_AUTENTICAZIONE_MQZ *(QMgrName, SecurityParms, ApplicationContext, IdentityContext, CorrelationPtr, ComponentData, Continuation, CompCode, Reason)*

Parametri

La chiamata MQZ_AUTHENTICATE_USER ha i seguenti parametri.

QMgrName (MQCHAR48) - input
È il nome del gestore code.

Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null. Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

SecurityParms (MQCSP) - input
Parametri di sicurezza.

Dati relativi all'ID utente, alla password e al tipo di autenticazione.

Durante una chiamata MQCONN MQI questo parametro contiene valori null o predefiniti.

ApplicationContext (MQZAC) - input
Contesto dell'applicazione.

Dati relativi all'applicazione chiamante. Vedi “MQZAC (Contesto applicazione) su IBM i” a pagina 1761 per i dettagli. Durante ogni chiamata MQCONN o MQCONNX MQI, le informazioni di contesto utente nella struttura MQZAC vengono riacquisite.

IdentityContext (MQZIC) - input/output
Contesto identità.

Nell'input della funzione di autenticazione utente, identifica il contesto di identità corrente. La funzione di autenticazione utente può modificare questa condizione, a quel punto il gestore code adotta il nuovo contesto di identità. Consultare “MQZIC (contesto di identità) su IBM i” a pagina 1768 per ulteriori dettagli sulla struttura MQZIC.

CorrelationPtr (MQPTR) - output
Puntatore di correlazione.

Specifica l'indirizzo dei dati di correlazione. Questo puntatore viene quindi passato ad altre chiamate OAM.

ComponentData (MQBYTE x ComponentDataLength) - input/output
Dati componente.

Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una delle funzioni fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una di queste funzioni del componente. La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione (MQLONG) - output
Indicatore di continuazione.

È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente da altri componenti.

STOP MQZCI
Non continuare con il componente successivo.

CompCode (MQLONG) - output
Codice di completamento.

Il valore è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo (MQLONG) - output
Codice di errore *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

Per ulteriori informazioni su questi codici di errore, consultare Messaggi e codici di errore.

Richiamo C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                   IdentityContext, &CorrelationPtr, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQCSP     SecurityParms;      /* Security parameters */
MQZAC     ApplicationContext; /* Application context */
MQZIC     IdentityContext;    /* Identity context */
MQPTR     CorrelationPtr;     /* Correlation pointer */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                                 component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

IBM i MQZ_CHECK_AUTHORITY (Verifica autorizzazione) in IBM i

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_1 e viene richiamata dal gestore code per controllare se un'entità dispone dell'autorizzazione per eseguire una o più azioni particolari su un oggetto specificato.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_CHECK_AUTHORITY.

Sintassi

MQZ_CHECK_AUTHORITY ***(QMgrName, EntityName, EntityType,***
ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

Parametri

La chiamata MQZ_CHECK_AUTHORITY ha i seguenti parametri.

QMgrName (MQCHAR48) - input
È il nome del gestore code.

Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null. Il nome del gestore

code viene trasmesso al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

EntityName (MQCHAR12) - input
Nome entità.

Il nome dell'entità la cui autorizzazione all'oggetto deve essere controllata. La lunghezza massima della stringa è di 12 caratteri; se è più corta viene riempita a destra con spazi. Il nome non termina con un carattere null.

Non è essenziale che questa entità sia nota al servizio di sicurezza sottostante. Se non è noto, per il controllo vengono utilizzate le autorizzazioni del gruppo speciale **nobody** (a cui si presume appartengano tutte le entità). Un nome completamente vuoto è valido e può essere utilizzato in questo modo.

EntityType (MQLONG) - input
Tipo di entità.

Il tipo di entità specificata da *EntityName*. Il valore è uno dei seguenti:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

ObjectName (MQCHAR48) - input
Il nome dell'oggetto.

Il nome dell'oggetto a cui è richiesto l'accesso. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

Se *ObjectType* è MQOT_Q_MGR, questo nome è uguale a *QMgrName*.

ObjectType (MQLONG) - input
Tipo di oggetto.

Il tipo di entità specificata da *ObjectName*. Il valore è uno dei seguenti:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

Autorizzazione (MQLONG) - input
Autorizzazione da controllare.

Se viene controllata un'autorizzazione, questo campo è uguale all'operazione di autorizzazione appropriata (costante MQZAO_ *). Se viene controllata più di un'autorizzazione, è l'OR bit per bit delle costanti MQZAO_ * corrispondenti.

Le seguenti autorizzazioni si applicano all'utilizzo delle chiamate MQI:

CONNECT MQZAO_
Possibilità di utilizzare la chiamata MQCONN.

MQZAO_BROWSE
Possibilità di utilizzare la chiamata MQGET con un'opzione di esplorazione.

Ciò consente alle opzioni MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR o MQGMO_BROWSE_NEXT di essere specificate nella chiamata MQGET.

INPUT MQZAO_
Capacità di utilizzare la chiamata MQGET con un'opzione di input.

Ciò consente all'opzione MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE o MQOO_INPUT_AS_Q_DEF di essere specificata nella chiamata MQOPEN.

OUTPUT MQZAO_
Capacità di utilizzare la chiamata MQPUT.

Ciò consente di specificare l'opzione MQOO_OUTPUT nella chiamata MQOPEN.

INQUIRE MQZAO_
Possibilità di utilizzare la chiamata MQINQ.

Ciò consente di specificare l'opzione MQOO_INQUIRE nella chiamata MQOPEN.

MQZAO_SET
Possibilità di utilizzare la chiamata MQSET.

Ciò consente all'opzione MQOOO_SET di essere specificata nella chiamata MQOPEN.

Contesto MQZAO_PASS_IDENTITY_CONTEXT
Capacità di passare il contesto di identità.

Ciò consente di specificare l'opzione MQOO_PASS_IDENTITY_CONTEXT nella chiamata MQOPEN e l'opzione MQPMO_PASS_IDENTITY_CONTEXT nelle chiamate MQPUT e MQPUT1 .

MQZAO_PASS_ALL_CONTEXT
Capacità di passare tutto il contesto.

Ciò consente di specificare l'opzione MQOO_PASS_ALL_CONTEXT nella chiamata MQOPEN e l'opzione MQPMO_PASS_ALL_CONTEXT nelle chiamate MQPUT e MQPUT1 .

MQZAO_SET_IDENTITY_CONTEXT
Capacità di impostare il contesto di identità.

Ciò consente di specificare l'opzione MQOO_SET_IDENTITY_CONTEXT nella chiamata MQOPEN e l'opzione MQPMO_SET identity_context nelle chiamate MQPUT e MQPUT1 .

MQZAO_SET_ALL_CONTEXT
Possibilità di impostare tutto il contesto.

Ciò consente di specificare l'opzione MQOO_SET_ALL_CONTEXT nella chiamata MQOPEN e l'opzione MQPMO_SET_ALL_CONTEXT nelle chiamate MQPUT e MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY
Possibilità di utilizzare l'autorizzazione utente alternativa.

Ciò consente di specificare l'opzione MQOO_ALTERNATE_USER_AUTHORITY nella chiamata MQOPEN e l'opzione MQPMO_ALTERNATE_USER_AUTHORITY nella chiamata MQPUT1 .

MQZAO_ALL_MQI
Tutte le autorizzazioni MQI.

Ciò abilita tutte le autorizzazioni descritte in precedenza.

Le seguenti autorizzazioni si applicano alla gestione di un gestore code:

CREA_MQZAO_
Capacità di creare oggetti di un tipo specificato.

MQZAO_DELETE
Possibilità di eliminare un oggetto specificato.

DISPLAY MQZAO_
Possibilità di visualizzare gli attributi di un oggetto specificato.

MODIFICA_MQZO
Possibilità di modificare gli attributi di un oggetto specificato.

CLEAR MQZAO_
Possibilità di eliminare tutti i messaggi da una coda specificata.

MQZAO_AUTHORIZE
Possibilità di autorizzare altri utenti per un oggetto specificato.

CONTROL MQZAO_
Possibilità di avviare, arrestare o eseguire il ping di un oggetto canale non client.

MQZAO_CONTROL_XX_ENCODE_CASE_ONE uscita
Possibilità di reimpostare un numero di sequenza o risolvere un messaggio in dubbio su un oggetto canale non client.

MQZAO_ALL_ADMIN
Tutte le autorizzazioni di gestione, tranne MQZAO_CREATE.

Le seguenti autorizzazioni si applicano sia all'utilizzo di MQI che alla gestione di un gestore code:

MQZAO_ALL
Tutte le autorizzazioni, diverse da MQZAO_CREATE.

MQZAO_NONE
Nessuna autorizzazione.

ComponentData (MQBYTE x ComponentDataLength) - input/output
Dati componente.

Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una delle funzioni fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una delle funzioni di questo componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione (MQLONG) - output
Indicatore di continuazione impostato per componente.

È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_CHECK_AUTHORITY questo ha lo stesso effetto di MQZCI_STOP.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode (MQLONG) - output
Codice di completamento.

Il valore è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo (MQLONG) - output
Codice di errore *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

Per ulteriori informazioni su questi codici di errore, consultare Messaggi e codici di errore.

Richiamo C

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_CHECK_PRIVILEGED - Verifica se l'utente è privilegiato

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_6 e viene richiamata dal gestore code per stabilire se un utente specificato è un utente privilegiato.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_CHECK_PRIVILEGED.

Sintassi

MQZ_CHECK_PRIVILEGED(*QMgrName* , *EntityData* , *EntityType* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametri

QMgrName
Tipo: MQCHAR48 - input

È il nome del gestore code. Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

EntityData
Tipo: MQZED - input

Dati entità. I dati relativi all'entità che deve essere controllata. Per ulteriori informazioni, consultare “MQZED - Descrittore entità” a pagina 1724.

EntityType
Tipo: MQLONG - input

Tipo di entità. Il tipo di entità specificato da EntityData. Deve essere uno dei seguenti valori:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

ComponentData
Tipo: MQBYTExComponentDataLength - input/output

Dati componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una di queste funzioni del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione
Tipo: MQLONG - output

Indicatore di continuazione impostato per componente. È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_CHECK_AUTHORITY, questo ha lo stesso effetto di MQZCI_STOP.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

Se la chiamata a un componente ha esito negativo (ovvero, *CompCode* restituisce MQCC_FAILED) e il parametro *Continuation* è MQZCI_DEFAULT o MQZCI_CONTINUE, il gestore code continua a richiamare altri componenti, se presenti.

Se la chiamata ha esito positivo (ovvero, *CompCode* restituisce MQCC_OK), nessun altro componente viene richiamato indipendentemente dall'impostazione di *Continuation* .

Se la chiamata ha esito negativo e il parametro *Continuazione* è MQZCI_STOP, non viene richiamato alcun altro componente e l'errore viene restituito al gestore code. I componenti non sono a conoscenza delle chiamate precedenti, quindi il parametro *Continuazione* è sempre impostato su MQZCI_DEFAULT prima della chiamata.

CompCode
Tipo: MQLONG - output

Codice di completamento. Deve essere uno dei seguenti valori:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo
Tipo: MQLONG - output

Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_NOT_PRIVILEGED
(2584, X'A18') Questo non è un ID utente privilegiato.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entità sconosciuta al servizio.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

Per ulteriori informazioni su questi codici di errore, consultare Completamento API e codici di errore.

Richiamo C

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,
                      ComponentData, &Continuation,
                      &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQZED     EntityData;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_COPY_ALL_AUTHORITY (Copia tutte le autorizzazioni) su IBM i

Questa funzione viene fornita da un componente del servizio di autorizzazione. Viene richiamato dal gestore code per copiare tutte le autorizzazioni attualmente in vigore per un oggetto di riferimento in un altro oggetto.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_COPY_ALL_AUTHORITY.

Sintassi

MQZ_COPY_ALL_AUTHORITY ***(QMgrName, RefObjectName, ObjectName,***
ObjectType, ComponentData, Continuation, CompCode, Reason)

Parametri

La chiamata MQZ_COPY_ALL_AUTHORITY ha i seguenti parametri.

QMgrName (MQCHAR48) - input
È il nome del gestore code.

Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene trasmesso al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

Nome RefObject(MQCHAR48) - input
Nome oggetto di riferimento.

Il nome dell'oggetto di riferimento, le cui autorizzazioni devono essere copiate. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

ObjectName (MQCHAR48) - input
Il nome dell'oggetto.

Il nome dell'oggetto per cui devono essere impostati gli accessi. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

ObjectType (MQLONG) - input
Tipo di oggetto.

Il tipo di oggetto specificato da *RefObjectName* e *ObjectName*. Il valore è uno dei seguenti:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

ComponentData (MQBYTE x ComponentDataLength) - input/output
Dati componente.

Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una delle funzioni fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una delle funzioni di questo componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione (MQLONG) - output
Indicatore di continuazione impostato per componente.

È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_COPY_ALL_AUTHORITY questo ha lo stesso effetto di MQZCI_STOP.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode (MQLONG) - output
Codice di completamento.

Il valore è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo (MQLONG) - output
Codice di errore *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

MQRC_UNKNOWN_REF_OBJECT
(2294, X'8F6') Oggetto di riferimento sconosciuto.

Per ulteriori informazioni su questi codici di errore, consultare Messaggi e codici di errore.

Richiamo C

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,
                        ComponentData, &Continuation, &CompCode,
                        &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  RefObjectName;     /* Reference object name */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_DELETE_AUTHORITY (Elimina autorizzazione) su IBM i

Questa funzione viene fornita da un componente del servizio di autorizzazione e viene richiamata dal gestore code per eliminare tutte le autorizzazioni associate all'oggetto specificato.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_DELETE_AUTHORITY.

Sintassi

MQZ_DELETE_AUTHORITY ***(QMgrName, ObjectName, ObjectType,***
ComponentData, Continuation, CompCode, Reason)

Parametri

La chiamata MQZ_DELETE_AUTHORITY ha i seguenti parametri.

QMgrName (MQCHAR48) - input
È il nome del gestore code.

Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene trasmesso al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

ObjectName (MQCHAR48) - input
Il nome dell'oggetto.

Il nome dell'oggetto per il quale è necessario eliminare gli accessi. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

Se *ObjectType* è MQOT_Q_MGR, questo nome è uguale a *QMgrName*.

ObjectType (MQLONG) - input
Tipo di oggetto.

Il tipo di entità specificata da *ObjectName*. Il valore è uno dei seguenti:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

ComponentData (MQBYTE x ComponentDataLength) - input/output
Dati componente.

Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una delle funzioni fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una delle funzioni di questo componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione (MQLONG) - output
Indicatore di continuazione impostato per componente.

È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_DELETE_AUTHORITY questo ha lo stesso effetto di MQZCI_STOP.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode (MQLONG) - output
Codice di completamento.

Il valore è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo (MQLONG) - output
Codice di errore *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

Per ulteriori informazioni su questi codici di errore, consultare Messaggi e codici di errore.

Richiamo C

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,
                      &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_ENUMERATE_AUTHORITY_DATA (Enumerate authority data) su IBM i

Questa funzione viene fornita da un componente del servizio di autenticazione MQZAS_VERSION_4 e viene richiamata ripetutamente dal gestore code per richiamare tutti i dati di autorizzazione che corrispondono ai criteri di selezione specificati al primo richiamo.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_ENUMERATE_AUTHORITY_DATA.

Sintassi

MQZ_ENUMERATE_AUTHORITY_DATA *(QMgrName, StartEnumeration,*
Filter, AuthorityBufferLength, AuthorityBuffer, AuthorityDataLength,
ComponentData, Continuation, CompCode, Reason)

Parametri

La chiamata MQZ_ENUMERATE_AUTHORITY_DATA ha i seguenti parametri.

QMgrName (MQCHAR48) - input
È il nome del gestore code.

Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene trasmesso al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

StartEnumeration (MQLONG) - input
Indicatore che indica se la chiamata deve avviare l'enumerazione.

Ciò indica se la chiamata deve avviare l'enumerazione dei dati di autorizzazione o continuare l'enumerazione dei dati di autorizzazione avviata da una precedente chiamata a MQZ_ENUMERATE_AUTHORITY_DATA. Il valore è uno dei seguenti:

INIZIO_MQZSE
Avvia enumerazione.

La chiamata viene richiamata con questo valore per avviare l'enumerazione dei dati di autorizzazione. Il parametro **Filter** specifica i criteri di scelta da utilizzare per selezionare i dati di autorizzazione restituiti da questa e successive chiamate.

CONTINUE MQZSE
Continuare l'enumerazione.

La chiamata viene richiamata con questo valore per continuare l'enumerazione dei dati di autorizzazione. Il parametro **Filter** viene ignorato in questo caso e può essere specificato come puntatore null (i criteri di selezione sono determinati dal parametro **Filter** specificato dalla chiamata che aveva *StartEnumeration* impostato su MQZSE_START).

Filtro (MQZAD) - input
Filtro.

Se *StartEnumeration* è MQZSE_START, *Filter* specifica i criteri di selezione da utilizzare per selezionare i dati di autorizzazione da restituire. Se *Filter* è il puntatore null, non viene utilizzato alcun criterio di selezione, ovvero, vengono restituiti tutti i dati di autorizzazione. Consultare “MQZAD (dati di autorizzazione) su IBM i” a pagina 1763 per dettagli sui criteri di selezione che possono essere utilizzati.

Se *StartEnumeration* è MQZSE_CONTINUE, *Filter* viene ignorato e può essere specificato come puntatore null.

AuthorityBufferLength (MQLONG) - input
Lunghezza di *AuthorityBuffer*.

È la lunghezza in byte del parametro **AuthorityBuffer** . Il buffer di autorizzazioni deve essere abbastanza grande da contenere i dati da restituire.

AuthorityBuffer (MQZAD) - output
Dati di autorizzazione.

Questo è il buffer in cui vengono restituiti i dati di autorizzazione. Il buffer deve essere abbastanza grande da contenere una struttura MQZAD, una struttura MQZED, più il nome entità più lungo e il nome dominio più lungo definito.

Nota: Questo parametro è definito come MQZAD, poiché MQZAD si verifica sempre all'inizio del buffer. Tuttavia, se il buffer è effettivamente dichiarato come un MQZAD, il buffer sarà troppo piccolo - deve

essere più grande di un MQZAD in modo che possa accogliere i nomi di entità e dominio MQZAD, MQZED.

AuthorityDataLunghezza (MQLONG) - output
Lunghezza dei dati restituiti in *AuthorityBuffer*.

Questa è la lunghezza dei dati restituiti in *AuthorityBuffer*. Se il buffer di autorizzazione è troppo piccolo, *AuthorityDataLength* viene impostato sulla lunghezza del buffer richiesto e la chiamata restituisce il codice di completamento MQCC_FAILED e il codice motivo MQRC_BUFFER_LENGTH_ERROR.

ComponentData (MQBYTE x ComponentDataLength) - input/output
Dati componente.

Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una delle funzioni fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una delle funzioni di questo componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione (MQLONG) - output
Indicatore di continuazione impostato per componente.

È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_ENUMERATE_AUTHORITY_DATA questo ha lo stesso effetto di MQZCI_CONTINUE.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode (MQLONG) - output
Codice di completamento.

Il valore è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo (MQLONG) - output
Codice di errore *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE MQRC_BUFFER_LENGTH
(2005, X'7D5') Parametro di lunghezza del buffer non valido.

MQRC_NO_DATA_AVAILABLE
(2379, X'94B') Nessun dato disponibile.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

Per ulteriori informazioni su questi codici di errore, consultare Messaggi e codici di errore.

Richiamo C

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,
                             AuthorityBufferLength,
                             &AuthorityBuffer,
                             &AuthorityDataLength, ComponentData,
                             &Continuation, &CompCode,
                             &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;               /* Queue manager name */
MQLONG    StartEnumeration;       /* Flag indicating whether call should
                                     start enumeration */
MQZAD     Filter;                 /* Filter */
MQLONG    AuthorityBufferLength;  /* Length of AuthorityBuffer */
MQZAD     AuthorityBuffer;        /* Authority data */
MQLONG    AuthorityDataLength;    /* Length of data returned in
                                     AuthorityBuffer */
MQBYTE    ComponentData[n];       /* Component data */
MQLONG    Continuation;           /* Continuation indicator set by
                                     component */
MQLONG    CompCode;               /* Completion code */
MQLONG    Reason;                 /* Reason code qualifying CompCode */
```

MQZ_FREE_USER - Utente libero

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_5 e viene richiamata dal gestore code per liberare la risorsa assegnata associata. Viene richiamato quando un'applicazione ha terminato l'esecuzione in tutti i contesti utente, ad esempio durante una chiamata MQI MQDISC.

L'identificativo funzione per questa funzione (per MQZEP) è MQZID_FREE_USER.

IBM i MQZ_GET_AUTHORITY (Richiama autorizzazione) in IBM i

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_1 e viene richiamata dal gestore code per richiamare l'autorizzazione di cui dispone un'entità per accedere all'oggetto specificato.

L'identificativo funzione per questa funzione (per MQZEP) è MQZID_GET_AUTHORITY.

Sintassi

MQZ_GET_AUTHORITY ***(QMgrName, EntityName, EntityType, ObjectName,***
ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

Parametri

La chiamata MQZ_GET_AUTHORITY ha i seguenti parametri.

QMgrName (MQCHAR48) - input
È il nome del gestore code.

Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene trasmesso al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

EntityName (MQCHAR12) - input
Nome entità.

Il nome dell'entità il cui accesso all'oggetto deve essere richiamato. La lunghezza massima della stringa è di 12 caratteri; se è più corta viene riempita a destra con spazi. Il nome non termina con un carattere null.

EntityType (MQLONG) - input
Tipo di entità.

Il tipo di entità specificata da *EntityName*. È possibile specificare il seguente valore:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

ObjectName (MQCHAR48) - input
Il nome dell'oggetto.

Il nome dell'oggetto per cui deve essere richiamata l'autorità dell'entità. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

Se *ObjectType* è MQOT_Q_MGR, questo nome è uguale a *QMgrName*.

ObjectType (MQLONG) - input
Tipo di oggetto.

Il tipo di entità specificata da *ObjectName*. Il valore è uno dei seguenti:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

Autorizzazione (MQLONG) - output
Autorità dell'entità.

Se l'entità dispone di un'autorizzazione, questo campo è uguale all'operazione di autorizzazione appropriata (costante MQZAO_ *). Se ha più di un'autorizzazione, questo campo è l'OR bit per bit delle costanti MQZAO_ * corrispondenti.

ComponentData (MQBYTE x ComponentDataLength) - input/output
Dati componente.

Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una delle funzioni fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una delle funzioni di questo componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione (MQLONG) - output
Indicatore di continuazione impostato per componente.

È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_GET_AUTHORITY questo ha lo stesso effetto di MQZCI_CONTINUE.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode (MQLONG) - output
Codice di completamento.

Il valore è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo (MQLONG) - output
Codice di errore *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entità sconosciuta al servizio.

Per ulteriori informazioni su questi codici di errore, consultare Messaggi e codici di errore.

Richiamo C

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, &Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_GET_EXPLICIT_AUTHORITY (Richiama autorizzazione esplicita) su IBM i

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_1 e viene richiamata dal gestore code per richiamare l'autorizzazione che un gruppo denominato ha per accedere a un oggetto specificato (ma senza l'autorizzazione aggiuntiva del gruppo **nobody**) o l'autorizzazione che il gruppo principale del principal denominato ha per accedere a un oggetto specificato.

L'identificativo funzione per questa funzione (per MQZEP) è MQZID_GET_EXPLICIT_AUTHORITY.

Sintassi

MQZ_GET_EXPLICIT_AUTHORITY (***QMgrName, EntityName, EntityType,***
ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

Parametri

La chiamata MQZ_GET_EXPLICIT_AUTHORITY ha i seguenti parametri.

QMgrName (MQCHAR48) - input
È il nome del gestore code.

Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene trasmesso al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

EntityName (MQCHAR12) - input
Nome entità.

Il nome dell'entità da cui richiamare l'accesso all'oggetto. La lunghezza massima della stringa è di 12 caratteri; se è più corta viene riempita a destra con spazi. Il nome non termina con un carattere null.

EntityType (MQLONG) - input
Tipo di entità.

Il tipo di entità specificata da *EntityName*. È possibile specificare il seguente valore:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

ObjectName (MQCHAR48) - input
Il nome dell'oggetto.

Il nome dell'oggetto per cui deve essere richiamata l'autorità dell'entità. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

Se *ObjectType* è MQOT_Q_MGR, questo nome è uguale a *QMgrName*.

ObjectType (MQLONG) - input
Tipo di oggetto.

Il tipo di entità specificata da *ObjectName*. Il valore è uno dei seguenti:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

Autorizzazione (MQLONG) - output
Autorità dell'entità.

Se l'entità dispone di un'autorizzazione, questo campo è uguale all'operazione di autorizzazione appropriata (costante MQZAO_ *). Se ha più di un'autorizzazione, questo campo è l'OR bit per bit delle costanti MQZAO_ * corrispondenti.

ComponentData (MQBYTE x ComponentDataLength) - input/output
Dati componente.

Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una delle funzioni fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una delle funzioni di questo componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione (MQLONG) - output
Indicatore di continuazione impostato per componente.

È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_GET_EXPLICIT_AUTHORITY ha lo stesso effetto di MQZCI_CONTINUE.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode (MQLONG) - output
Codice di completamento.

Il valore è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo (MQLONG) - output
Codice di errore *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entità sconosciuta al servizio.

Per ulteriori informazioni su questi codici di errore, consultare Messaggi e codici di errore.

Richiamo C

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,
                            ObjectName, ObjectType, &Authority,
                            ComponentData, &Continuation,
                            &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_INIT_AUTHORITY (Inizializza servizio di autorizzazione) su IBM i

Questa funzione viene fornita da un componente del servizio di autorizzazione e viene richiamato dal gestore code durante la configurazione del componente. È previsto che richiami MQZEP per fornire informazioni al gestore code.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_INIT_AUTHORITY.

Sintassi

MQZ_INIT_AUTHORITY ***(Hconfig, Options, QMgrName, ComponentDataLength,***
ComponentData, Version, CompCode, Reason)

Parametri

La chiamata MQZ_INIT_AUTHORITY ha i seguenti parametri.

Hconfig (MQHCONFIG) - input
Handle di configurazione.

Questo handle rappresenta il particolare componente in fase di inizializzazione. Deve essere utilizzato dal componente quando si richiama il gestore code con la funzione MQZEP.

Opzioni (MQLONG) - input
Opzioni di inizializzazione.

Il valore è uno dei seguenti:

MQZIO_PRIMARIO
Inizializzazione primaria.

MQZIO_SECONDARY
Inizializzazione secondaria.

QMgrName (MQCHAR48) - input
È il nome del gestore code.

Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene trasmesso al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

ComponentDataLength (MQLONG) - input
Lunghezza dei dati del componente.

Lunghezza in byte dell'area *ComponentData* . Questa lunghezza è definita nei dati di configurazione del componente.

ComponentData (MQBYTE x ComponentDataLength) - input/output
Dati componente.

Viene inizializzato su tutti gli zeri prima di richiamare la funzione di inizializzazione principale del componente. Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una delle funzioni (inclusa la funzione di inizializzazione) fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una delle funzioni di questo componente.

Versione (MQLONG) - input/output
Numero di versione.

In fase di input per la funzione di inizializzazione, identifica il numero di versione *più alto* supportato dal gestore code. La funzione di inizializzazione deve modificarla, se necessario, nella versione dell'interfaccia che *supporta* . Se alla restituzione il gestore code non supporta la versione restituita dal componente, richiama la funzione MQZ_TERM_AUTHORITY del componente e non utilizza più questo componente.

Sono supportati i seguenti valori:

MQZAS_VERSION_1
Versione 1.

MQZAS_VERSION_2
Versione 2.

MQZAS_VERSION_3
Versione 3.

MQZAS_VERSION_4
Versione 4.

MQZAS_VERSION_5
Versione 5.

MQZAS_VERSION_6
IBM WebSphere MQ 6.

CompCode (MQLONG) - output
Codice di completamento.

Il valore è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo (MQLONG) - output
Codice di errore *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_INITIALIZATION_FAILED
(2286, X'8EE') Inizializzazione non riuscita per un motivo indefinito.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

Per ulteriori informazioni su questi codici di errore, consultare Messaggi e codici di errore.

Richiamo C

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,
                    ComponentData, &Version, &CompCode,
                    &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQHCONFIG  Hconfig;              /* Configuration handle */
MQLONG     Options;              /* Initialization options */
MQCHAR48   QMgrName;             /* Queue manager name */
MQLONG     ComponentDataLength;  /* Length of component data */
MQBYTE     ComponentData[n];     /* Component data */
MQLONG     Version;              /* Version number */
MQLONG     CompCode;             /* Completion code */
MQLONG     Reason;               /* Reason code qualifying CompCode */
```

IBM i MQZ_INQUIRE (Interroga servizio di autorizzazione) su IBM i

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_5 e viene richiamata dal gestore code per interrogare la funzionalità supportata. Se vengono utilizzati più componenti di servizio, i componenti di servizio vengono richiamati in ordine inverso rispetto all'ordine in cui sono stati installati.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_INQUIRE.

Sintassi

INQUIRE MQZ
(QMgrName, SelectorCount, Selectors, IntAttrCount, IntAttrs, CharAttrLength, CharAttrs, SelectorReturned, ComponentData, Continuation, CompCode, Reason)

Parametri

La chiamata MQZ_INQUIRE presenta i seguenti parametri.

QMgrName (MQCHAR48) - input
È il nome del gestore code.

Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene trasmesso al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

SelectorCount (MQLONG) - input
Numero di selettori.

Il numero di selettori forniti nel parametro Selettori.

Il valore deve essere compreso tra zero e 256.

Selettori (MQLONG x SelectorCount) - input
Selettori.

Array di selettori. Ogni selettore identifica un attributo richiesto e deve essere di uno dei seguenti tipi:

- MQIACF_ * (intero)
- MQCACF_ * (carattere)

I selettori possono essere specificati in qualsiasi ordine. Il numero di selettori nell'array è indicato dal parametro SelectorCount .

Gli attributi interi identificati dai selettori vengono restituiti nel parametro IntAttrs nello stesso ordine in cui vengono visualizzati nei selettori.

Gli attributi carattere identificati dai selettori vengono restituiti nel parametro CharAttrs nello stesso ordine in cui vengono visualizzati nei selettori.

Conteggio IntAttr(MQLONG) - input
Numero di attributi interi.

Numero di attributi interi forniti nel parametro IntAttrs .

Il valore deve essere compreso tra 0 e 256.

IntAttrs (MQLONG IntAttrConteggio) - output
Attributi numero intero.

Array di attributi integer. Gli attributi integer vengono restituiti nello stesso ordine dei selettori integer corrispondenti nell'array di selettori.

CharAttrCount (MQLONG) - input
Lunghezza del buffer degli attributi carattere.

La lunghezza in byte del parametro CharAttrs .

Il valore deve essere almeno la somma delle lunghezze degli attributi carattere richiesti. Se non è richiesto alcun attributo carattere, zero è un valore valido.

CharAttrs (MQLONG x CharAttrCount) - output
Buffer attributi carattere.

Buffer contenente attributi carattere, concatenati. Gli attributi del carattere vengono restituiti nello stesso ordine dei corrispondenti selettori di caratteri nell'array Selettori.

La lunghezza del buffer viene fornita dal parametro Conteggio CharAttr.

SelectorReturned (ConteggioMQLONGxSelector) - input
Selettore restituito.

Array di valori che identificano quali attributi sono stati restituiti dalla serie richiesta dai selettori nel parametro Selettori. Il numero di valori in questo array viene indicato dal parametro SelectorCount . Ogni valore nell'array si riferisce al selettore dalla posizione corrispondente nell'array Selettori. Ogni valore è uno dei seguenti:

MQZSL_RESTITUITO
L'attributo richiesto dal selettore corrispondente nel parametro Selettori è stato restituito.

MQZSL_NON_RESTITUITO
L'attributo richiesto dal selettore corrispondente nel parametro Selettori non è stato restituito.

L'array viene inizializzato con tutti i valori come *MQZSL_NOT_RESI*. Quando un componente del servizio di autorizzazione restituisce un attributo, imposta il valore appropriato nell'array su *MQZSL_RETURNS*. Ciò consente a qualsiasi altro componente del servizio di autorizzazione, a cui viene effettuata la chiamata di interrogazione, di identificare quali attributi sono già stati restituiti.

ComponentData (MQBYTE x ComponentDataLength) - input/output
Dati componente.

Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una delle funzioni fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una delle funzioni di questo componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione (MQLONG) - output
Indicatore di continuazione.

È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente da altri componenti.

STOP MQZCI
Non continuare con il componente successivo.

CompCode (MQLONG) - output
Codice di completamento.

Il valore è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_AVVERTENZA
Completamento parziale.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo (MQLONG) - output
Codice di errore *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT
Spazio insufficiente per gli attributi carattere.

MQRC_INT_COUNT_TOO_SMALL
Spazio insufficiente per gli attributi integer.

Se *CompCode* è MQCC_FAILED:

ERRORE MQRC_SELECTOR_COUNT
Il numero di selettori non è valido.

ERRORE DI MQRC_SELECTOR_ERROR
Selettore attributo non valido.

MQRC_SELECTOR_LIMIT_EXCEEDED
Sono stati specificati troppi selettori.

ERRORE MQRC_INT_ATTR_COUNT_
Il numero di attributi interi non è valido.

ERRORE - MQRC_INT_ATTRS_ARRAY_ERROR
Schiera attributi interi non valida.

MQRC_CHAR_ATTR_LENGTH_ERROR
Numero di attributi carattere non valido.

ERRORE MQRC_CHAR_ATTRS_
La stringa degli attributi carattere non è valida.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

Richiamo C

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
                  &IntAttrs, CharAttrLength, &CharAttrs,
                  SelectorReturned, ComponentData, &Continuation,
                  &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48   QMgrName;              /* Queue manager name */
MQLONG     SelectorCount;         /* Selector count */
MQLONG     Selectors[n];          /* Selectors */
MQLONG     IntAttrCount;          /* IntAttrs count */
MQLONG     IntAttrs[n];           /* Integer attributes */
MQLONG     CharAttrCount;         /* CharAttrs count */
MQLONG     CharAttrs[n];          /* Chatacter attributes */
MQLONG     SelectorReturned[n];   /* Selector returned */
MQBYTE     ComponentData[n];      /* Component data */
MQLONG     Continuation;          /* Continuation indicator set by
                                     component */
MQLONG     CompCode;              /* Completion code */
MQLONG     Reason;                /* Reason code qualifying CompCode */
```

IBM i MQZ_REFRESH_CACHE (Aggiorna tutte le autorizzazioni) su IBM i

Questa funzione è fornita da un componente del Servizio di autorizzazione MQZAS_VERSION_3 . Viene richiamata dal gestore code per aggiornare l'elenco di autorizzazioni conservate internamente dal componente.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_REFRESH_CACHE (8L).

Sintassi

MQZ_REFRESH_CACHE
(QMgrName, ComponentData, Continuazione, CompCode, Motivo)

Parametri

***QMgrName* (MQCHAR48) - input**
È il nome del gestore code.

Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene passato al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

***ComponentData* (MQBYTE x *ComponentDataLunghezza*) - input/output**
Dati componente.

Questi dati vengono conservati dal gestore code per conto di questo particolare componente. Qualsiasi modifica apportata da una delle funzioni fornite da questo componente viene conservata e presentata la volta successiva che viene richiamata una funzione del componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro *ComponentDataLength* della chiamata MQZ_INIT_AUTHORITY.

***Continuazione* (MQLONG) - output**
Indicatore di continuazione impostato per componente.

È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_REFRESH_CACHE, ha lo stesso effetto di MQZCI_CONTINUE.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

***CompCode* (MQLONG) - output**
Codice di completamento.

Il valore è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

***Motivo* (MQLONG) - output**
Codice motivo che qualifica *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

Richiamo C

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Dichiarare i parametri come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_SET_AUTHORITY (Imposta autorizzazione) su IBM i

Questa funzione viene fornita da un componente del servizio di autorizzazione MQZAS_VERSION_1 e viene richiamata dal gestore code per impostare l'autorizzazione che un'entità ha per accedere all'oggetto specificato.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_SET_AUTHORITY.

Nota: Questa funzione sostituisce tutte le autorizzazioni esistenti. Per conservare le autorizzazioni esistenti, è necessario impostarle nuovamente con questa funzione.

Sintassi

MQZ_SET_AUTHORITY ***(QMgrName, EntityName, EntityType, ObjectName,***
ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

Parametri

La chiamata MQZ_SET_AUTHORITY ha i seguenti parametri.

QMgrName (MQCHAR48) - input
È il nome del gestore code.

Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene trasmesso al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

EntityName (MQCHAR12) - input
Nome entità.

Il nome dell'entità per cui deve essere impostato l'accesso all'oggetto. La lunghezza massima della stringa è di 12 caratteri; se è più corta viene riempita a destra con spazi. Il nome non termina con un carattere null.

EntityType (MQLONG) - input
Tipo di entità.

Il tipo di entità specificata da *EntityName*. È possibile specificare il seguente valore:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

ObjectName (MQCHAR48) - input
Il nome dell'oggetto.

Il nome dell'oggetto a cui è richiesto l'accesso. La lunghezza massima della stringa è di 48 caratteri; se è più breve viene riempita a destra con spazi. Il nome non termina con un carattere null.

Se *ObjectType* è MQOT_Q_MGR, questo nome è uguale a *QMgrName*.

ObjectType (MQLONG) - input
Tipo di oggetto.

Il tipo di entità specificata da *ObjectName*. Il valore è uno dei seguenti:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

Autorizzazione (MQLONG) - input
Autorizzazione da controllare.

Se viene impostata un'autorizzazione, questo campo è uguale all'operazione di autorizzazione appropriata (costante MQZAO_ *). Se viene impostata più di un'autorizzazione, è l'OR bit per bit delle costanti MQZAO_ * corrispondenti.

ComponentData (MQBYTE x ComponentDataLength) - input/output
Dati componente.

Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una delle funzioni fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una delle funzioni di questo componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** della chiamata MQZ_INIT_AUTHORITY.

Continuazione (MQLONG) - output
Indicatore di continuazione impostato per componente.

È possibile specificare i seguenti valori:

MQZCI_PREDEFINITO
Continuazione dipendente dal gestore code.

Per MQZ_SET_AUTHORITY questo ha lo stesso effetto di MQZCI_STOP.

CONTINUE MQZCI
Continuare con il componente successivo.

STOP MQZCI
Non continuare con il componente successivo.

CompCode (MQLONG) - output
Codice di completamento.

Il valore è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo (MQLONG) - output
Codice di errore *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Non autorizzato per l'accesso.

ERRORE_SERVIZIO_MQRC
(2289, X'8F1') Si è verificato un errore non previsto durante l'accesso al servizio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entità sconosciuta al servizio.

Richiamo C

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_TERM_AUTHORITY - Termina servizio di autorizzazione

Questa funzionalità è fornita da un componente del servizio di autorizzazione e viene richiamata dal gestore code quando non richiede più i servizi di questo componente. La funzione deve eseguire qualsiasi ripulitura richiesta dal componente.

L'identificativo della funzione per questa funzione (per MQZEP) è MQZID_TERM_AUTHORITY.

Sintassi

MQZ_TERM_AUTHORITY ***(Hconfig, Options, QMgrName, ComponentData,***
CompCode, Reason)

Parametri

La chiamata MQZ_TERM_AUTHORITY ha i seguenti parametri.

Hconfig (MQHCONFIG) - input
Handle di configurazione.

Questo handle rappresenta il componente particolare che viene terminato.

Opzioni (MQLONG) - input
Opzioni di terminazione.

Il valore è uno dei seguenti:

MQZTO_PRIMARIO
Terminazione primaria.

MQZTO_SECONDARY
Terminazione secondaria.

QMgrName (MQCHAR48) - input
È il nome del gestore code.

Il nome del gestore code che richiama il componente. Questo nome viene riempito con spazi vuoti fino alla lunghezza completa del parametro; il nome non termina con un carattere null.

Il nome del gestore code viene trasmesso al componente per informazioni; l'interfaccia del servizio di autorizzazione non richiede che il componente lo utilizzi in modo definito.

ComponentData (MQBYTE x ComponentDataLength) - input/output
Dati componente.

Questi dati vengono conservati dal gestore code per conto di questo particolare componente; tutte le modifiche apportate ad esso da una delle funzioni fornite da questo componente vengono conservate e presentate la volta successiva che viene richiamata una delle funzioni di questo componente.

La lunghezza di questa area dati viene passata dal gestore code nel parametro **ComponentDataLength** sulla chiamata MQZ_INIT_AUTHORITY.

Una volta completata la chiamata MQZ_TERM_AUTHORITY, il gestore code elimina questi dati.

CompCode (MQLONG) - output
Codice di completamento.

Il valore è uno dei seguenti:

MQCC_OK
Completamento con esito positivo.

MQCC_NON RIUSCITO
Chiamata fallita.

Motivo (MQLONG) - output
Codice di errore *CompCode*.

Se *CompCode* è MQCC_OK:

MQRC_NONE
(0, X'000 ') Nessun motivo per segnalare.

Se *CompCode* è MQCC_FAILED:

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Servizio di supporto non disponibile.

MQRC_TERMINATION_FAILED
(2287, X'8FF') Terminazione non riuscita per un motivo non definito.

Per ulteriori informazioni su questi codici di errore, consultare Messaggi e codici di errore.

Richiamo C

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,
                    &CompCode, &Reason);
```

I parametri passati al servizio vengono dichiarati come segue:

```
MQHCONFIG  Hconfig;           /* Configuration handle */
MQLONG     Options;           /* Termination options */
MQCHAR48   QMgrName;          /* Queue manager name */
MQBYTE     ComponentData[n];  /* Component data */
MQLONG     CompCode;          /* Completion code */
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZAC (Contesto applicazione) su IBM i

Questo parametro specifica i dati relativi all'applicazione chiamante.

La struttura MQZAC viene utilizzata nella chiamata MQZ_AUTHENTICATE_USER per il parametro **ApplicationContext** .

Campi

StrucId (MQCHAR4)
Identificatore struttura.

Il valore è:

ID_STRUC_MQZAC
Identificativo per la struttura del contesto dell'applicazione.

Per il linguaggi di programmazione C, viene definita anche la costante MQZAC_STRUC_ID_ARRAY, che ha lo stesso valore di MQZAC_STRUC_ID, ma è un array di caratteri invece di una stringa.

Questo è un campo di input per il servizio.

Versione (MQLONG
Numero di versione della struttura.

Il valore è:

MQZAC_VERSION_1
Struttura del contesto dell'applicazione Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQZAC_CURRENT_
La versione corrente della struttura del contesto dell'applicazione.

Questo è un campo di input per il servizio.

ProcessId (MQPID)
Identificativo processo.

L'identificativo del processo dell'applicazione.

ThreadId (MQTID)
Identificativo sottoprocesso.

L'identificativo del thread dell'applicazione.

ApplName (MQCHAR28)
Nome dell'applicazione

Il nome dell'applicazione.

UserID (MQCHAR12)
Identificativo utente.

Per sistemi IBM i , il profilo utente con cui è stato creato il lavoro dell'applicazione. (Su IBM i, quando viene eseguito uno scambio di profilo con la API QWTSETP nel lavoro dell'applicazione, viene restituito il profilo utente corrente).

ID EffectiveUser(MQCHAR12)
Identificativo utente effettivo.

Per i sistemi IBM i il profilo utente corrente del lavoro dell'applicazione.

Ambiente (MQLONG)
Ambiente.

Questo campo specifica l'ambiente da cui è stata effettuata la chiamata.

Può avere uno dei seguenti valori:

SERVER MQXE_COMMAND_
Server dei comandi.

MQXE_MQSC
Interprete del comando `runmqsc` .

MQXE_MCA
Agent del canale messaggi

MQXE_ALTRO
Ambiente non definito

CallerType (MQLONG)
Tipo di chiamante.

Questo campo specifica il tipo di programma che ha effettuato la chiamata.

Può avere uno dei seguenti valori:

MQXACT_EXTERNAL
La chiamata è esterna al gestore code.

MQXACT_INTERNO
La chiamata è interna al gestore code.

AuthenticationType (MQLONG)
Tipo di autenticazione.

Questo campo specifica il tipo di autenticazione da eseguire.

Può avere uno dei seguenti valori:

MQZAT_CONTESTO_INIZIALE
La chiamata di autenticazione è dovuta al contesto utente in fase di inizializzazione. Questo valore viene utilizzato durante una chiamata MQCONN o MQCONNX .

MQZAT_CONTESTO_MODIFICA
La chiamata di autenticazione è dovuta al fatto che il contesto utente è stato modificato. Questo valore viene utilizzato quando l'MCA modifica il contesto utente.

v

BindType (MQLONG
Tipo di bind.

Questo campo specifica il tipo di collegamento in uso.

Può avere uno dei seguenti valori:

MQCNO_FASTPATH_BINDING
Collegamento Fastpath.

MQCNO_SHARED_BINDING
Binding condiviso.

MQCNO_ISOLATED_BINDING
Collegamento isolato.

Dichiarazione C

```
typedef struct tagMQZAC MQZAC;
struct tagMQZAC {
  MQCHAR4   StrucId;            /* Structure identifier */
  MQLONG    Version;            /* Structure version number */
  MQPID     ProcessId;          /* Process identifier */
  MQTID     ThreadId;           /* Thread identifier */
  MQCHAR28  ApplName;           /* Application name */
  MQCHAR12  UserID;             /* User identifier */
  MQCHAR12  EffectiveUserID;    /* Effective user identifier */
  MQLONG    Environment;        /* Environment */
  MQLONG    CallerType;         /* Caller type */
  MQLONG    AuthenticationType; /* Authentication type */
  MQLONG    BindType;           /* Bind type */
};
```

IBM i MQZAD (dati di autorizzazione) su IBM i

La struttura MQZAD viene utilizzata sulla chiamata MQZ_ENUMERATE_AUTHORITY_DATA per due parametri.

Consultare “MQZ_ENUMERATE_AUTHORITY_DATA (Enumerate authority data) su IBM i” a pagina 1743 per ulteriori informazioni sui parametri **Filter** e **AuthorityBuffer** :

- MQZAD viene utilizzato per il parametro **Filter** che viene immesso nella chiamata. Questo parametro specifica i criteri di scelta da utilizzare per selezionare i dati di autorizzazione restituiti dalla chiamata.
- MQZAD viene utilizzato anche per il parametro **AuthorityBuffer** che è l'output della chiamata. Questo parametro specifica le autorizzazioni per una combinazione di nome profilo, tipo di oggetto ed entità.

Campi

StrucId (MQCHAR4)
Identificatore struttura.

Il valore è:

ID_STRUC_MQZAD
Identificativo per la struttura dati di autorizzazione.

Per il linguaggio di programmazione C, viene definita anche la costante MQZAD_STRUC_ID_ARRAY; ha lo stesso valore di MQZAD_STRUC_ID, ma è una schiera di caratteri invece di una stringa.

Questo è un campo di input per il servizio.

Version (MQLONG)
Numero di versione della struttura.

Il valore è:

MQZAD_VERSION_1
Struttura dati di autorizzazione Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQZAD_CURRENT_
La versione corrente della struttura dati di autorizzazione.

Questo è un campo di input per il servizio.

ProfileName (MQCHAR48)
Nome del profilo.

Per il parametro **Filter** , questo campo è il nome del profilo da cui sono necessari i dati di autorizzazione. Se il nome è completamente vuoto fino alla fine del campo o al primo carattere null, vengono restituiti i dati di autorizzazione per tutti i nomi profilo.

Per il parametro **AuthorityBuffer** , questo campo è il nome di un profilo che corrisponde ai criteri di selezione specificati.

ObjectType (MQLONG)
Tipo di oggetto.

Per il parametro **Filter** , questo campo è il tipo di oggetto per cui sono richiesti i dati di autorizzazione. Se il valore è MQOT_ALL, vengono restituiti i dati di autorizzazione per tutti i tipi di oggetto.

Per il parametro **AuthorityBuffer** , questo campo è il tipo di oggetto a cui si applica il profilo identificato da **ProfileName** .

Il valore è uno dei seguenti; per il parametro **Filter** , è valido anche il valore MQOT_ALL:

INFO MQOT_AUTH_O
Informazioni di autenticazione.

CANALIZZATA MQOT_
Canale.

MQOT_CLNTCONN_CHALLEGATO
Canale di connessione client.

LISTENER MQOT_
Listener.

ELENCO NOMI MQOTT
Elenco nomi.

PROCESSO MQOT_
Definizione processo.

MQOT_Q
Coda.

Gestore code MQOT_GR
Gestore code.

SERVIZIO_MQT
Servizio.

Autorizzazione (MQLONG)
Autorizzazione.

Per il parametro **Filter** , questo campo viene ignorato.

Per il parametro **AuthorityBuffer** , questo campo rappresenta le autorizzazioni che l'entità ha sugli oggetti identificati da **ProfileName** e **ObjectType**. Se l'entità dispone di una sola autorizzazione, il campo è uguale al valore di autorizzazione appropriato (costante MQZAO_ *). Se l'entità ha più di un'autorizzazione, il campo è l'OR bit per bit delle costanti MQZAO_ * corrispondenti.

Ptr EntityData(PMQZED)
Indirizzo della struttura MQZED che identifica un'entità.

Per il parametro **Filter** , questo campo indica una struttura MQZED che identifica l'entità da cui sono richiesti i dati di autorizzazione. Se **EntityDataPtr** è il puntatore null, vengono restituiti i dati di autorizzazione per tutte le entità.

Per il parametro **AuthorityBuffer** , questo campo punta a una struttura MQZED che identifica l'entità da cui provengono i dati di autorizzazione restituiti.

EntityType (MQLONG)
Tipo di entità.

Per il parametro **Filter** , questo campo specifica il tipo di entità per cui sono necessari i dati di autorizzazione. Se il valore è MQZAET_NONE, vengono restituiti i dati di autorizzazione per tutti i tipi di entità.

Per il parametro **AuthorityBuffer** , questo campo specifica il tipo di entità identificato dalla struttura MQZED indicata da **EntityDataPtr**.

Il valore è uno dei seguenti; per il parametro **Filter** , è valido anche il valore MQZAET_NONE:

PRINCIPALE_MQZAET
Principale.

GRUPPO_MQZ
Gruppo.

Opzioni (MQAUTHOPT)
Opzioni.

Questo campo specifica le opzioni che forniscono il controllo sui profili visualizzati.

È necessario specificare uno dei seguenti valori:

MQAUTHOPT_NAME_ALL_MATCHING
Visualizza tutti i profili

MQAUTHOPT_NAME_EXPLICIT
Visualizza i profili che hanno esattamente lo stesso nome specificato nel campo **ProfileName** .

Inoltre, deve essere specificato anche uno dei seguenti:

MQAUTOPT_ENTITA_SET
Visualizzare tutti i profili utilizzati per calcolare l'autorità cumulativa di cui l'entità dispone per l'oggetto specificato da **ProfileName**. Il campo **ProfileName** non deve contenere caratteri jolly.

- Se l'entità specificata è un principal, per ciascun membro della serie {entity, groups} viene visualizzato il profilo più applicabile che si applica all'oggetto.
- Se l'entità specificata è un gruppo, viene visualizzato il profilo più applicabile dal gruppo che si applica all'oggetto.
- Se questo valore viene specificato, i valori di **ProfileName**, **ObjectType**, **EntityType**e il nome entità specificato nella struttura MQZED **EntityDataPtr** , devono essere tutti non vuoti.

Se è stato specificato *MQAUTHOPT_NAME_ALL_MATCHING*, è anche possibile specificare quanto segue:

MQAUTHOPT_ENTITY_EXPLICIT
Visualizza i profili che hanno esattamente lo stesso nome entità del nome entità specificato nella struttura **EntityDataPtr** MQZED.

Dichiarazione C

```
typedef struct tagMQZAD MQZAD;
struct tagMQZAD {
  MQCHAR4   StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQCHAR48  ProfileName;    /* Profile name */
  MQLONG    ObjectType;     /* Object type */
  MQLONG    Authority;      /* Authority */
  PMQZED    EntityDataPtr;  /* Address of MQZED structure identifying an
                               entity */
  MQLONG    EntityType;     /* Entity type */
  MQAUTHOPT Options;        /* Options */
};
```

IBM i MQZED (descrittore entità) su IBM i

La struttura MQZED viene utilizzata in un numero di chiamate al servizio di autorizzazione per specificare l'entità per cui deve essere controllata l'autorizzazione.

Campi

StrucId (MQCHAR4)
Identificatore struttura.

Il valore è:

ID_STRUC_MQZ_
Identificativo per la struttura descrittore entità.

Per il linguaggio di programmazione C, è definita anche la costante MQZED_STRUC_ID_ARRAY, che ha lo stesso valore di MQZED_STRUC_ID, ma è un array di caratteri invece di una stringa.

Questo è un campo di input per il servizio.

Version (MQLONG)
Numero di versione della struttura.

Il valore è:

MQZED_VERSION_1
Struttura descrittore entità Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQZED_CURRENT_
La versione corrente della struttura del descrittore entità.

Questo è un campo di input per il servizio.

EntityNameptr (PMQCHAR)
Indirizzo del nome entità.

Questo è un puntatore al nome dell'entità la cui autorizzazione deve essere controllata.

Ptr EntityDomain(PMQCHAR)
Indirizzo del nome dominio entità.

Questo è un puntatore al nome del dominio contenente la definizione dell'entità la cui autorizzazione deve essere controllata.

SecurityId (MQBYTE40)
Identificativo di sicurezza.

Questo è l'identificativo di sicurezza la cui autorizzazione deve essere controllata.

CorrelationPtr (MQPTR)
Puntatore di correlazione.

Ciò facilita il trasferimento dei dati di correlazione tra la funzione utente di autenticazione e altre funzioni OAM appropriate.

Dichiarazione C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  PMQCHAR   EntityNamePtr;    /* Address of entity name */
  PMQCHAR   EntityDomainPtr;  /* Address of entity domain name */
  MQBYTE40  SecurityId;       /* Security identifier */
  MQPTR     CorrelationPtr;   /* Address of correlation data */
```

IBM i MQZFP (parametri liberi) su IBM i

Questo parametro specifica i dati relativi alla risorsa da liberare.

La struttura di MQZFP viene utilizzata nella chiamata MQZ_FREE_USER per il parametro **FreeParms** .

Campi

StrucId (MQCHAR4)
Identificatore struttura.

Il valore è:

ID_STRUC_MQZFP
Identificativo per la struttura dei parametri liberi.

Per il linguaggio di programmazione C, viene definita anche la costante MQZFP_STRUC_ID_ARRAY; ha lo stesso valore di MQZFP_STRUC_ID, ma è un array di caratteri invece di una stringa.

Questo è un campo di input per il servizio.

Version (MQLONG)
Numero di versione della struttura.

Il valore è:

MQZFP_VERSION_1
Struttura di parametri liberi Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQZFP_CURRENT_
La versione corrente della struttura dei parametri liberi.

Questo è un campo di input per il servizio.

Riservato (MQBYTE8)
Campo riservato.

Il valore iniziale è null.

CorrelationPtr (MQPTR)
Puntatore di correlazione.

Indirizzo dei dati di correlazione relativi alla risorsa da liberare.

Dichiarazione C

```
typedef struct tagMQZFP MQZFP;
struct tagMQZFP {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  MQBYTE8   Reserved;         /* Reserved field */
  MQPTR     CorrelationPtr;   /* Address of correlation data */
};
```

IBM i MQZIC (contesto di identità) su IBM i

La struttura MQZIC viene utilizzata nella chiamata MQZ_AUTHENTICATE_USER per il parametro **IdentityContext** .

La struttura MQZIC contiene informazioni sul contesto di identità, che identificano l'utente dell'applicazione che per primo ha inserito il messaggio su una coda:

- Il gestore code riempie il campo UserIdentifier con un nome che identifica l'utente, il modo in cui il gestore code può eseguire questa operazione dipende dall'ambiente in cui è in esecuzione l'applicazione.
- Il gestore code compila il campo AccountingToken con un token o un numero determinato dall'applicazione che ha inserito il messaggio.
- Le applicazioni possono utilizzare il campo Dati ApplIdentityper qualsiasi informazione aggiuntiva che si desidera includere sull'utente (ad esempio, una parola d'ordine codificata).

Le applicazioni debitamente autorizzate possono impostare il contesto di identità utilizzando la funzione MQZ_AUTHENTICATE_USER.

Un SID (systems security identifier) Windows viene memorizzato nel campo AccountingToken quando viene creato un messaggio in IBM MQ for Windows. Il SID può essere utilizzato per integrare il campo UserIdentifier e per stabilire le credenziali di un utente.

Campi

StrucId (MQCHAR4)
Identificatore struttura.

Il valore è:

ID_STRUC_MQZIC
Identificativo per la struttura del contesto identità.

Per il linguaggio di programmazione C, viene definita anche la costante MQZIC_STRUC_ID_ARRAY, che ha lo stesso valore di MQZIC_STRUC_ID, ma è un array di caratteri anziché una stringa.

Questo è un campo di input per il servizio.

Version (MQLONG)
Numero di versione della struttura.

Il valore è:

MQZIC_VERSION_1
Struttura del contesto di identità Version-1 .

La seguente costante specifica il numero di versione della versione corrente:

VERSIONE MQZIC_CURRENT_
La versione corrente della struttura del contesto di identità.

Questo è un campo di input per il servizio.

UserIdentifier (MQCHAR12)
Identificativo utente.

Fa parte del **contesto di identità** del messaggio.

UserIdentifier specifica l'ID utente dell'applicazione che ha originato il messaggio. Il gestore code considera queste informazioni come dati carattere, ma non ne definisce il formato. Per ulteriori informazioni sul campo *UserIdentifier* , consultare “UserIdentifier (MQCHAR12) per MQMD” a pagina 469.

AccountingToken (MQBYTE32)
Token di account.

Fa parte del **contesto di identità** del messaggio.

AccountingToken consente a una applicazione di far sì che il lavoro eseguito come risultato del messaggio venga addebitato in maniera appropriata. Il gestore code tratta queste informazioni come una stringa di bit e non ne controlla il contenuto. Per ulteriori informazioni sul campo *AccountingToken* , consultare “AccountingToken (MQBYTE32) per MQMD” a pagina 471.

Dati ApplIdentity(MQCHAR32)
Dati dell'applicazione relativi all'identità.

Fa parte del **contesto di identità** del messaggio.

ApplIdentityData è un'informazione definita dalla suite di applicazioni che può essere utilizzata per fornire ulteriori informazioni sull'origine del messaggio. Ad esempio, potrebbe essere impostato dalle applicazioni in esecuzione con l'autorizzazione utente appropriata per indicare se i dati di identità sono attendibili. Per ulteriori informazioni sul campo *ApplIdentityData* , consultare “Dati ApplIdentity(MQCHAR32) per MQMD” a pagina 472.

Dichiarazione C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  MQCHAR12  UserIdentifier;   /* User identifier */
  MQBYTE32  AccountingToken;  /* Accounting token */
  MQCHAR32  ApplIdentityData; /* Application data relating to identity */
};
```

Classi e interfacce di IBM MQ .NET

Le classi e interfacce IBM MQ .NET sono elencate in ordine alfabetico. Vengono descritti le proprietà, i metodi e i costruttori.

Classe MQAsyncStatus.NET

Utilizzare MQAsyncStatus per analizzare lo stato di un'attività MQI precedente; ad esempio, per analizzare la riuscita di operazioni di immissione asincrone precedenti. MQAsyncStatus incapsula le funzioni della struttura dati MQSTS .

Classe

```
System.Object
    │
    └─ IBM.WMQ.MQBase
        │
        └─ IBM.WMQ.MQBaseObject
            │
            └─ IBM.WMQ.MQAsyncStatus
```

```
public class IBM.WMQ.MQAsyncStatus extends IBM.WMQ.MQBaseObject;
```


- "Proprietà" a pagina 1770
- "Costruttori" a pagina 1771

Proprietà

Test per `MQException` generato durante il richiamo delle proprietà.

`public static int CompCode {get;}`

Il codice di completamento dal primo errore o avvertenza.

`public static int Reason {get;}`

Il codice di errore dal primo errore o avvertenza.

`public static int PutSuccessCount {get;}`

Il numero di chiamate di inserimento MQI asincrone riuscite.

`public static int PutWarningCount {get;}`

Il numero di chiamate di inserimento MQI asincrone riuscite con un'avvertenza.

`public static int PutFailureCount {get;}`

Il numero di chiamate di inserimento MQI asincrone non riuscite.

`public static int ObjectType {get;}`

Il tipo di oggetto per il primo errore. Sono possibili i seguenti valori:

- `MQC.MQOT_ALIAS_Q`
- `MQC.MQOT_LOCAL_Q`
- `MQC.MQOT_MODEL_Q`
- `MQC.MQOT_Q`
- `MQC.MQOT_REMOTE_Q`
- `MQC.MQOT_TOPIC`
- `0`, il che significa che non viene restituito alcun oggetto

`public static string ObjectName {get;}`

Il nome dell'oggetto.

`public static string ObjectQMgrName {get;}`

Il nome del gestore code oggetti.

`public static string ResolvedObjectName {get;}`

Il nome dell'oggetto risolto.

`public static string ResolvedObjectQMgrName {get;}`

Il nome del gestore code dell'oggetto risolto.

Costruttori

public MQAsyncStatus() throws MQException;

Metodo costruttore, crea un oggetto con campi inizializzati su zero o vuoti come appropriato.

Classe MQAuthenticationInformationRecord.NET

Utilizzare MQAuthenticationInformationRecord per specificare le informazioni su un programma di autenticazione da utilizzare in una connessione client TLS IBM MQ . MQAuthenticationInformationRecord incapsula un record delle informazioni di autenticazione, MQAIR.

Classe

```
System.Object
    |
    └ IBM.WMQ.MQAuthenticationInformationRecord
```

```
public class IBM.WMQ.MQAuthenticationInformationRecord extends System.Object;
```


- "Proprietà" a pagina 1771
- "Costruttori" a pagina 1772

Proprietà

Test per MQException generato durante il richiamo delle proprietà.

public long Version {get; set;}

Numero di versione della struttura.

public long AuthInfoType {get; set;}

Il tipo di informazioni di autenticazione. Questo attributo deve essere impostato su uno dei seguenti valori:

- OCSP - Il controllo dello stato di revoca del certificato viene eseguito utilizzando OCSP.
- CRLLDAP - Il controllo dello stato di revoca del certificato viene effettuato utilizzando i CRL (Certificate Revocation List) sui server LDAP.

public string AuthInfoConnName {get; set;}

Il nome DNS o l'indirizzo IP dell'host su cui è in esecuzione il server LDAP, con un numero di porta facoltativo. Questa parola chiave è obbligatoria.

public string LDAPPassword {get; set;}

La password associata al DN (distinguished name) dell'utente che accede al server LDAP. Questa proprietà si applica solo quando **AuthInfoType** è impostato su CRLLDAP.

public string LDAPUserName {get; set;}

Il DN dell'utente che accede al server LDAP. Quando si imposta questa proprietà, LDAPUserNameLength e LDAPUserNamePtr vengono impostate automaticamente correttamente. Questa proprietà si applica solo quando AuthInfoType è impostato su CRLLDAP.

public string OCSPResponderURL {get; set;}

L'URL al quale può essere contattato il responder OCSP. Questa proprietà si applica solo quando il tipo AuthInfo è impostato su OCSP

Questo campo è sensibile al maiuscolo/minuscolo. Deve iniziare con la stringa http:// in minuscolo. Il resto dell'URL potrebbe essere sensibile al maiuscolo / minuscolo, a seconda dell'implementazione del server OCSP.

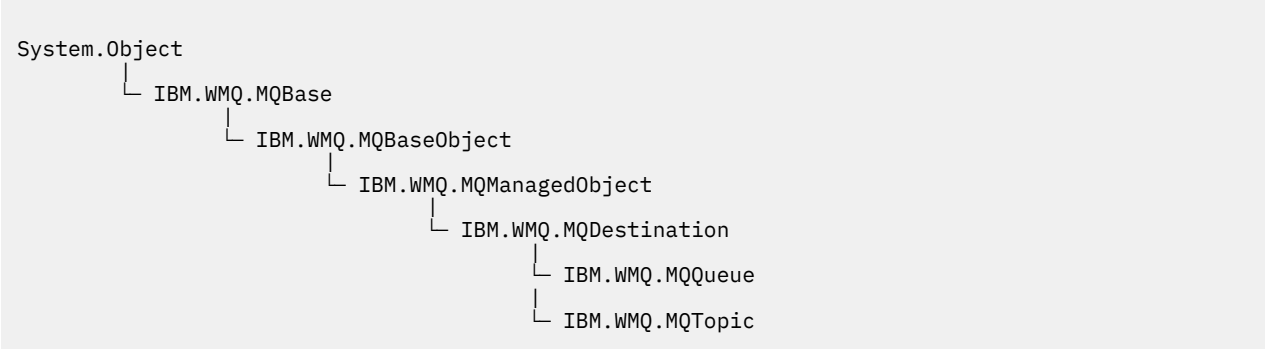
Costruttori

MQAuthenticationInformationRecord();

Classe MQDestination.NET

Utilizzare MQDestination per accedere ai metodi comuni a MQQueue e MQTopic. MQDestination è una classe di base astratta e non può essere istanziata.

Classe



public class IBM.WMQ.MQDestination extends IBM.WMQ.MQManagedObject;

- “Proprietà” a pagina 1772
- “Metodi” a pagina 1772
- “Costruttori” a pagina 1774

Proprietà

Test per MQException generato durante il richiamo delle proprietà.

public DateTime CreationDateTime {get;}

La data e l'ora di creazione della coda o dell'argomento. Originariamente contenuta in MQQueue, questa proprietà è stata spostata nella classe MQDestination di base.

Non è impostato alcun valore predefinito.

public int DestinationType {get;}

Valore intero che descrive il tipo di destinazione utilizzato. Inizializzato dal costruttore delle sottoclassi, MQQueue o MQTopic, questo valore può assumere uno dei seguenti valori:

- MQOT_Q
- MQOT_TOPIC

Non è impostato alcun valore predefinito.

Metodi

public void Get(MQMessage *message*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*, int *MaxMsgSize*);

Genera MQException.

Ottiene un messaggio da una coda se la destinazione è un oggetto MQQueue o da un argomento se la destinazione è un oggetto MQTopic , utilizzando un'istanza predefinita di MQGetMessageOptions per eseguire il richiamo.

Se il richiamo ha esito negativo, l'oggetto MQMessage non viene modificato. Se ha esito positivo, il descrittore del messaggio e le parti di dati del messaggio di MQMessage vengono sostituiti con il descrittore del messaggio e i dati del messaggio dal messaggio in arrivo.

Tutte le chiamate a IBM MQ da un particolare MQQueueManager sono sincrone. Pertanto, se si esegue un richiamo con attesa, a tutti gli altri thread che utilizzano lo stesso MQQueueManager viene impedito di effettuare ulteriori chiamate IBM MQ fino a quando non viene eseguita la chiamata Get. Se sono necessari più thread per accedere simultaneamente a IBM MQ , ogni thread deve creare il proprio oggetto MQQueueManager .

messaggio
Contiene il descrittore del messaggio e i dati del messaggio restituiti. Alcuni dei campi nel descrittore del messaggio sono parametri di input. È importante assicurarsi che i parametri di input MessageId e CorrelationId vengano impostati come richiesto.

Un client ricollegabile restituisce il codice di errore MQRC_BACKED_OUT dopo una riconnessione riuscita, per i messaggi ricevuti in MQGM_SYNCPOINT.

Opzioni getMessage
Opzioni che controllano l'azione del get.

L'utilizzo dell'opzione MQC.MQGMO_CONVERT potrebbe causare un'eccezione con codice di errore MQC.MQRC_CONVERTED_STRING_TOO_BIG durante la conversione da codici di caratteri a byte singolo in codici a byte doppio. In tal caso, il messaggio viene copiato nel buffer senza conversione.

Se *getMessageOptions* non è specificato, l'opzione del messaggio utilizzata è MQGMO_NOWAIT.

Se si utilizza l'opzione MQGMO_LOGICAL_ORDER in un client ricollegabile, viene restituito il codice di errore MQRC_RECONNECT_INCOMPATIBLE .

Dimensione MaxMsg
Il messaggio più grande che questo oggetto messaggio deve ricevere. Se il messaggio sulla coda è più grande di questa dimensione, si verifica una delle due seguenti situazioni:

- Se l'indicatore MQGMO_ACCEPT_TRUNCATED_MSG è impostato nell'oggetto MQGetMessageOptions , il messaggio viene riempito con il maggior numero possibile di dati del messaggio. Viene generata un'eccezione con il codice di completamento MQCC_WARNING e il codice motivo MQRC_TRUNCATED_MSG_ACCEPTED .
- Se l'indicatore MQGMO_ACCEPT_TRUNCATED_MSG non è impostato, il messaggio rimane nella coda. Viene generata un'eccezione con il codice di completamento MQCC_WARNING e il codice motivo MQRC_TRUNCATED_MSG_FAILED .

Se *MaxMsgSize* non è specificato, viene richiamato l'intero messaggio.

```
public void Put(MQMessage message);
public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);
```

Genera MQException.

Inserisce un messaggio in una coda se la destinazione è un oggetto MQQueue o pubblica un messaggio in un argomento se la destinazione è un oggetto MQTopic .

Le modifiche all'oggetto MQMessage dopo che la chiamata Put è stata effettuata non influiscono sul messaggio effettivo sulla coda IBM MQ o sull'argomento di pubblicazione.

Put aggiorna le proprietà MessageId e CorrelationId dell'oggetto MQMessage e non cancella i dati del messaggio. Ulteriori chiamate Put o Get fanno riferimento alle informazioni aggiornate nell'oggetto MQMessage . Ad esempio, nel seguente frammento di codice, il primo messaggio contiene a e il secondo ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

messaggio
Un MQMessage oggetto contenente i dati del descrittore del messaggio e il messaggio da inviare. Il descrittore del messaggio può essere modificato come conseguenza di questo metodo. I valori nel descrittore del messaggio immediatamente dopo il completamento di questo metodo sono i valori inseriti nella coda o pubblicati nell'argomento.

I seguenti codici di errore vengono restituiti ad un client ricollegabile:

- MQRC_CALL_INTERRUPTED se la connessione viene interrotta durante l'esecuzione di una chiamata Put su un messaggio persistente e la riconnessione ha esito positivo.
- MQRC_NONE se la connessione ha esito positivo durante l'esecuzione di una chiamata Put su un messaggio non persistente (consultare Ripristino applicazione).

Opzioni putMessage
Opzioni che controllano l'azione dell'inserimento.

Se *putMessageOptions* non è specificato, viene utilizzata l'istanza predefinita di MQPutMessageOptons .

Se si utilizza l'opzione MQPMO_LOGICAL_ORDER in un client ricollegabile, viene restituito il codice di errore MQRC_RECONNECT_INCOMPATIBLE .

Nota: Per semplicità e prestazioni, se si desidera inserire un singolo messaggio in una coda, utilizzare l'oggetto MQQueueManager.Put . Si dovrebbe avere un oggetto MQQueue per questo.

Costruttori
MQDestination è una classe di base astratta e non può essere istanziata. Accedere alle destinazioni utilizzando i costruttori MQQueue e MQTopic oppure utilizzando MQQueueManager.AccessQueue e MQQueueManager.AccessTopic methods.

Classe MQEnvironment.NET

Utilizzare MQEnvironment per controllare come viene richiamato il costruttore MQQueueManager e per selezionare una connessione IBM MQ MQI client . La classe MQEnvironment contiene proprietà che controllano il funzionamento di IBM MQ.

Classe

```
System.Object
        |
        └─ IBM.WMQ.MQEnvironment
```

```
public class IBM.WMQ.MQEnvironment extends System.Object;
```

- “Proprietà - solo client” a pagina 1774
- “Proprietà” a pagina 1775
- “Costruttori” a pagina 1777

Proprietà - solo client

Test per MQException generato durante il richiamo delle proprietà.

public static int CertificateValPolicy {get; set;}

Impostare la politica di convalida del certificato TLS utilizzata per convalidare i certificati digitali ricevuti dai sistemi partner remoti. I valori validi sono:

- MQC.CERTIFICATE_VALIDATION_POLICY_ANY
- MQC.CERTIFICATE_VALIDATION_POLICY_RFC5280

public static ArrayList EncryptionPolicySuiteB {get; set;}

Impostare il livello di crittografia conforme a Suite B. I valori validi sono:

- MQC.MQ_SUITE_B_NONE - Questo è il valore predefinito.
- MQC.MQ_SUITE_B_128_BIT
- MQC.MQ_SUITE_B_192_BIT

public static string Channel {get; set;}

Il nome del canale per la connessione al gestore code di destinazione. Devi impostare la proprietà del canale prima di creare un'istanza `MQQueueManager` in modalità client.

public static int FipsRequired {get; set;}

Specificare `MQC.MQSSL_FIPS_YES` per utilizzare solo algoritmi certificati FIPS se la crittografia viene eseguita in IBM MQ. Il valore predefinito è `MQC.MQSSL_FIPS_NO`.

Se l'hardware di crittografia è configurato, i moduli di crittografia utilizzati sono quelli forniti dal prodotto hardware. A seconda dell'hardware in uso, questi potrebbero non essere certificati FIPS ad un particolare livello.

public static string Hostname {get; set;}

Il nome host TCP/IP del computer su cui risiede il server IBM MQ . Se il nome host non è impostato e non è impostata alcuna proprietà di sovrascrittura, viene utilizzata la modalità di bind del server per la connessione al gestore code locale.

public static int Port {get; set;}

La porta a cui connettersi. Questa è la porta su cui il server IBM MQ è in ascolto delle richieste di connessione in entrata. Il valore predefinito è 1414.

public static string SSLCipherSpec {get; set;}

Impostare `SSLCipherSpec` sul valore di CipherSpec impostato sul canale SVRCONN per abilitare TLS per la connessione. Il valore predefinito è Null e TLS non è abilitato per la connessione.

public static string sslPeerName {get; set;}

Un modello DN (distinguished name). Se `sslCipherSpec` è impostato, questa variabile può essere utilizzata per garantire che venga utilizzato il gestore code corretto. Se impostato su null (valore predefinito), il DN del gestore code non viene eseguito. `sslPeerName` viene ignorato se `sslCipherSpec` è null.

public static string SSLKeyRepository {get; set;}

Il testo semplice o la passphrase codificata per accedere al repository delle chiavi. Le passphrase del repository di chiavi sono codificate per l'utilizzo da parte delle applicazioni client utilizzando il programma di utilità **runmqicred** .

Se `SSLKeyRepositoryPassword` è impostato su null (valore predefinito), viene utilizzato il valore della variabile di ambiente *MQKEYRPWD* o l'attributo **SSLKeyRepositoryPassword** nel file di configurazione del client.

public static string InitialKey {get; set;}

La chiave iniziale utilizzata per codificare la passphrase del repository chiavi specificata in `SSLKeyRepositoryPassword`.

La chiave iniziale deve essere specificata se è stato specificato un file di chiavi iniziale quando la passphrase del repository chiavi è stata codificata utilizzando il programma di utilità **runmqicred** .

Proprietà

Test per `MQException` generato durante il richiamo delle proprietà.

public static ArrayList HdrCompList {get; set;}

Elenco di compressione dati di intestazione

public static int KeyResetCount {get; set;}

Indica il numero di byte non codificati inviati e ricevuti all'interno di una conversazione TLS prima che la chiave segreta venga rinegoziata.

public static ArrayList MQAIRArray {get; set;}
Un array di oggetti MQAuthenticationInformationRecord .

public static ArrayList MsgCompList {get; set;}
Elenco di compressione dati messaggio

public static string Password {get; set;}
La password da autenticare. La password a cui si fa riferimento dalla struttura MQCSP viene popolata impostando questa proprietà Password.

public static string ReceiveExit {get; set;}
Un'uscita di ricezione consente di esaminare e modificare i dati ricevuti da un gestore code. Viene normalmente utilizzato con un'uscita di invio corrispondente sul gestore code. Se ReceiveExit è impostato su null, non viene richiamata alcuna uscita di ricezione.

public static string ReceiveUserData {get; set;}
I dati utente associati a un'uscita di ricezione. Limitato a 32 caratteri.

public static string SecurityExit {get; set;}
Un'uscita di sicurezza consente di personalizzare i flussi di protezione che si verificano quando viene effettuato un tentativo di connettersi a un gestore code. Se SecurityExit è impostato su null, non viene richiamata alcuna uscita di sicurezza.

public static string SecurityUserData {get; set;}
I dati utente associati a un'uscita di sicurezza. Limitato a 32 caratteri.

public static string SendExit {get; set;}
Un'uscita di invio consente di esaminare o modificare i dati inviati a un gestore code. Viene normalmente utilizzato con un'uscita di ricezione corrispondente sul gestore code. Se SendExit è impostato su null, non viene richiamata alcuna uscita di invio.

public static string SendUserData {get; set;}
I dati utente associati a un'uscita di invio. Limitato a 32 caratteri.

public static string SharingConversations {get; set;}

Il campo SharingConversations viene utilizzato sulle connessioni da applicazioni .NET , quando queste applicazioni non utilizzano una CCDT (client channel definition table).

SharingConversations determina il numero massimo di conversazioni che è possibile condividere su un socket associato a questa connessione.

Il valore 0 indica che il canale funziona come prima di IBM WebSphere MQ 7.0, per quanto riguarda la condivisione della conversazione, la lettura anticipata e l'heartbeat.

Il campo viene inoltrato nella tabella hash delle proprietà come SHARING_CONVERSATIONS_PROPERTY, quando si crea un'istanza di un gestore code IBM MQ .

Se non si specifica SharingConversations, viene utilizzato il valore predefinito 10.

public static string SSLCryptoHardware {get; set;}

Imposta il nome della stringa di parametro richiesta per configurare l'hardware crittografico presente sul sistema. SSLCryptoHardware viene ignorato se sslCipherSpec è null.

public static string SSLKeyRepository {get; set;}

Impostare il nome file completo del repository chiavi.

Se l'estensione file non viene fornita, si presume che sia .kdb

Se SSLKeyRepository è impostato su null (impostazione predefinita), la variabile di ambiente MQSSLKEYR del certificato viene utilizzata per individuare il repository delle chiavi.

public static string UserId {get; set;}

L'ID utente da autenticare. L'ID utente a cui si fa riferimento dalla struttura MQCSP viene popolato impostando UserId. Autenticare UserId utilizzando un'API o un'uscita di sicurezza.

Costruttori

public MQEnvironment()

Classe MQException.NET

Utilizzare MQException per individuare il codice di completamento e motivo di una funzione IBM MQ non riuscita. Un MQException viene generato ogni volta che si verifica un errore IBM MQ .

Classe

```
System.Object
        |
        └─ System.Exception
                |
                └─ System.ApplicationException
                        |
                        └─ IBM.WMQ.MQException
```

public class IBM.WMQ.MQException extends System.ApplicationException;

- “Proprietà” a pagina 1777
- “Costruttori” a pagina 1777

Proprietà

public int CompletionCode {get; set;}
Il codice di completamento IBM MQ associato all'errore. I valori possibili sono:

- MQException.MQCC_OK
- MQException.MQCC_WARNING
- MQException.MQCC_FAILED

public int ReasonCode {get; set;}
Codice di errore IBM MQ che descrive l'errore.

Costruttori

public MQException(int *completionCode*, int *reasonCode*)

completionCode
Il codice di completamento IBM MQ .

reasonCode
Il codice di completamento IBM MQ .

Classe MQGetMessageOptions.NET

Utilizzare MQGetMessageOptions per specificare in che modo vengono richiamati i messaggi. Modifica il funzionamento di MQDestination.Get.

Classe

```
System.Object
        |
        └─ IBM.WMQ.MQBase
                |
                └─ IBM.WMQ.MQBaseObject
                        |
                        └─ IBM.WMQ.MQGetMessageOptions
```

```
public class IBM.WMQ.MQGetMessageOptions extends IBM.WMQ.MQBaseObject;
```


Proprietà

Nota: Il comportamento di alcune delle opzioni disponibili in questa classe dipende dall'ambiente in cui vengono utilizzate. Questi elementi sono contrassegnati con un asterisco *.

Test per `MQException` generato durante il richiamo delle proprietà.

public int GroupStatus {get;}*
`GroupStatus` indica se il messaggio richiamato si trova in un gruppo e se è l'ultimo nel gruppo. I possibili valori sono:

MQC.MQGS_LAST_MSG_IN_GROUP
Il messaggio è l'ultimo o l'unico messaggio nel gruppo.

MQC.MQGS_MSG_IN_GROUP
Il messaggio si trova in un gruppo, ma non è l'ultimo nel gruppo.

MQC.MQGS_NOT_IN_GROUP
Il messaggio non è in un gruppo.

public int MatchOptions {get; set;}*
`MatchOptions` determina come viene selezionato un messaggio. È possibile impostare le seguenti opzioni di corrispondenza:

MQC.MQMO_MATCH_CORREL_ID
ID correlazione da mettere in corrispondenza.

MQC.MQMO_MATCH_GROUP_ID
ID gruppo da mettere in corrispondenza.

MQC.MQMO_MATCH_MSG_ID
ID messaggio da mettere in corrispondenza.

MQC.MQMO_MATCH_MSG_SEQ_NUMBER
Corrisponde al numero di sequenza del messaggio.

MQC.MQMO_NONE
Nessuna corrispondenza richiesta.

public int Options {get; set;}
Le `Opzioni` controllano l'azione di `MQQueue.get`. È possibile specificare uno dei seguenti valori. Se è richiesta più di un'opzione, i valori possono essere aggiunti o combinati utilizzando l'operatore OR bit per bit.

MQC.MQGMO_ACCEPT_TRUNCATED_MSG
Consente il troncamento dei dati del messaggio.

MQC.MQGMO_ALL_MSGS_AVAILABLE*
Richiamare i messaggi da un gruppo solo quando tutti i messaggi nel gruppo sono disponibili.

MQC.MQGMO_ALL_SEGMENTS_AVAILABLE*
Richiamare i segmenti di un messaggio logico solo quando tutti i segmenti nel gruppo sono disponibili.

MQC.MQGMO_BROWSE_FIRST
Sfoglia dall'inizio della coda.

MQC.MQGMO_BROWSE_MSG_UNDER_CURSOR*
Sfoglia messaggio sotto il cursore di ricerca.

MQC.MQGMO_BROWSE_NEXT
Sfoglia dalla posizione corrente nella coda.

MQC.MQGMO_COMPLETE_MSG*
Richiamare solo i messaggi logici completi.

MQC.MQGMO_CONVERT
Richiedere i dati dell'applicazione da convertire, per essere conformi agli attributi `CharacterSet` e `Codifica` di `MQMessage`, prima che i dati vengano copiati nel buffer di messaggio. Poiché la conversione dei dati viene applicata anche quando i dati vengono richiamati dal buffer dei messaggi, le applicazioni non impostano questa opzione.

L'uso di questa opzione può causare problemi durante la conversione da serie di caratteri a byte singolo a serie di caratteri a doppio byte. Eseguire invece la conversione utilizzando i metodi `readString`, `readLinee` `writeString` dopo che il messaggio è stato consegnato.

MQC.MQGMO_FAIL_IF_QUIESCING
Errore se il gestore code è in attesa.

MQC.MQGMO_LOCK*
Bloccare il messaggio visualizzato.

MQC.MQGMO_LOGICAL_ORDER*
Restituisce i messaggi in gruppi e segmenti di messaggi logici, in ordine logico.

Se si utilizza l'opzione `MQGMO_LOGICAL_ORDER` in un client ricollegabile, il codice di errore `MQRC_RECONNECT_INCOMPATIBLE` viene restituito all'applicazione.

MQC.MQGMO_MARK_SKIP_BACKOUT*
Consentire il backout di un'unità di lavoro senza ripristinare il messaggio sulla coda.

MQC.MQGMO_MSG_UNDER_CURSOR
Richiamare il messaggio sotto il cursore di ricerca.

MQC.MQGMO_NONE
Non sono state specificate altre opzioni; tutte le opzioni assumono i valori predefiniti.

MQC.MQGMO_NO_PROPERTIES
Non viene richiamata alcuna proprietà del messaggio, tranne le proprietà contenute nel descrittore del messaggio (o estensione).

MQC.MQGMO_NO_SYNCPOINT
Richiamare il messaggio senza il controllo del punto di sincronizzazione.

MQC.MQGMO_NO_WAIT
Restituire immediatamente se non è presente alcun messaggio adatto.

MQC.MQGMO_PROPERTIES_AS_Q_DEF
Richiamare le proprietà del messaggio come definito dall'attributo `PropertyControl` di `MQQueue`. L'accesso alle proprietà del messaggio nel descrittore del messaggio o nell'estensione non è influenzato dall'attributo `PropertyControl` .

MQC.MQGMO_PROPERTIES_COMPATIBILITY
Richiamare le proprietà del messaggio con un prefisso `mcd`, `jms`, `usro` `mqext`, nelle intestazioni `MQRFH2` . Le altre proprietà del messaggio, ad eccezione delle proprietà contenute nel descrittore del messaggio o nell'estensione, vengono eliminate.

MQC.MQGMO_PROPERTIES_FORCE_MQRFH2
Richiamare le proprietà del messaggio, tranne le proprietà contenute nel descrittore del messaggio o nell'estensione, nelle intestazioni `MQRFH2` . Utilizzare `MQC.MQGMO_PROPERTIES_FORCE_MQRFH2` nelle applicazioni che prevedono di richiamare le proprietà ma che non possono essere modificate per utilizzare gli handle del messaggio.

MQC.MQGMO_PROPERTIES_IN_HANDLE
Richiamare le proprietà del messaggio utilizzando `MsgHandle`.

MQC.MQGMO_SYNCPOINT
Richiamare il messaggio sotto il controllo del punto di sincronizzazione. Il messaggio è contrassegnato come non disponibile per altre applicazioni, ma viene eliminato dalla coda solo quando viene eseguito il commit dell'unità di lavoro. Il messaggio viene reso nuovamente disponibile se viene eseguito il backout dell'unità di lavoro.

MQC.MQGMO_SYNCPOINT_IF_PERSISTENT*
Richiamare il messaggio con il controllo del punto di sincronizzazione se il messaggio è persistente.

MQC.MQGMO_UNLOCK*
Sbloccare un messaggio precedentemente bloccato.

MQC.MQGMO_WAIT
Attendere l'arrivo di un messaggio.

public string ResolvedQueueName {get;}
Il gestore code imposta `ResolvedQueueNome` sul nome locale della coda da cui è stato richiamato il messaggio. `ResolvedQueueNome` è diverso dal nome utilizzato per aprire la coda se è stata aperta una coda alias o una coda modello.

public char Segmentation {get;}*
`Segmentazione` indica se è possibile consentire la segmentazione per il messaggio richiamato. I possibili valori sono:

MQC.MQSEG_INHIBITED
Non consentire la segmentazione.

MQC.MQSEG_ALLOWED
Consenti segmentazione

public byte SegmentStatus {get;}*
`SegmentStatus` è un campo di output che indica se il messaggio richiamato è un segmento di un messaggio logico. Se il messaggio è un segmento, l'indicatore indica se si tratta dell'ultimo segmento. I possibili valori sono:

MQC.MQSS_LAST_SEGMENT
Il messaggio è l'ultimo o l'unico segmento del messaggio logico.

MQC.MQSS_NOT_A_SEGMENT
Il messaggio non è un segmento.

MQC.MQSS_SEGMENT
Il messaggio è un segmento, ma non è l'ultimo segmento del messaggio logico.

public int WaitInterval {get; set;}
`WaitInterval` è il periodo di tempo massimo in millisecondi durante il quale una chiamata `MQQueue.get` attende l'arrivo di un messaggio adatto. Utilizzare `WaitInterval` con `MQC.MQGMO_WAIT`. Impostare il valore `MQC.MQWI_UNLIMITED` per attendere un tempo illimitato per un messaggio.

Costruttori

public MQGetMessageOptions()
Costruire un nuovo oggetto `MQGetMessageOptions` con `Opzioni` impostato su `MQC.MQGMO_NO_WAIT`, `WaitInterval` impostato a zero e `ResolvedQueueName` impostato su vuoto.

Classe MQManagedObject.NET

Utilizzare `MQManagedObject` per analizzare e impostare attributi di `MQDestination`, `MQProcess`, `MQQueueManagere` `MQSubscription`. `MQManagedObject` è una superclasse di queste classi.

Classi

```
System.Object
        │
        └─ IBM.WMQ.MQBase
                │
                └─ IBM.WMQ.MQBaseObject
                        │
                        └─ IBM.WMQ.MQManagedObject
                                │
                                └─ IBM.WMQ.MQProcess
                                │
                                └─ IBM.WMQ.MQQueueManager
                                │
                                └─ IBM.WMQ.MQSubscription
                                │
                                └─ IBM.WMQ.MQDestination
```

```
public class IBM.WMQ.MQManagedObject extends IBM.WMQ.MQBaseObject;
```


- “Proprietà” a pagina 1781
- “Metodi” a pagina 1782
- “Costruttori” a pagina 1783

Proprietà

Test per `MQException` generato durante il richiamo delle proprietà.

public string AlternateUserId {get; set;}
L'ID utente alternativo, se presente, impostato quando è stata aperta la risorsa. `AlternateUserID.set` viene ignorato quando viene emesso per un oggetto aperto. `AlternateUserId` non è valido per le sottoscrizioni.

public int CloseOptions {get; set;}

Impostare questo attributo per controllare la modalità di chiusura della risorsa. Il valore predefinito è `MQC.MQCO_NONE`. `MQC.MQCO_NONE` è l'unico valore consentito per tutte le risorse diverse dalle code dinamiche permanenti, dalle code dinamiche temporanee, dalle sottoscrizioni e dagli argomenti a cui accedono gli oggetti che le hanno create.

Per code e argomenti, sono consentiti i seguenti valori aggiuntivi:

MQC.MQCO_DELETE
Eliminare la coda se non sono presenti messaggi.

MQC.MQCO_DELETE_PURGE
Eliminare la coda, eliminando tutti i messaggi su di essa.

MQC.MQCO_QUIESCE
Richiedere la chiusura della coda, ricevendo un avviso se rimangono dei messaggi (consentendo loro di essere richiamati prima della chiusura finale).

Per le sottoscrizioni, sono consentiti i seguenti valori aggiuntivi:

MQC.MQCO_KEEP_SUB
La sottoscrizione non viene eliminata. Questa opzione è valida solo se la sottoscrizione originale è durevole. `MQC.MQCO_KEEP_SUB` è il valore predefinito per un argomento durevole.

MQC.MQCO_REMOVE_SUB
La sottoscrizione viene eliminata. `MQC.MQCO_REMOVE_SUB` è il valore predefinito per un argomento non durevole e non gestito.

MQC.MQCO_PURGE_SUB
La sottoscrizione viene eliminata. `MQC.MQCO_PURGE_SUB` è il valore predefinito per un argomento gestito non durevole.

public MQQueueManager ConnectionReference {get;}
Il gestore code a cui appartiene questa risorsa.

public string MQDescription {get;}
La descrizione della risorsa conservata dal gestore code. `MQDescription` restituisce una stringa vuota per le sottoscrizioni e gli argomenti.

public boolean IsOpen {get;}
Indica se la risorsa è attualmente aperta.

public string Name {get;}
Il nome della risorsa. Il nome è quello fornito sul metodo di accesso o quello assegnato dal gestore code per una coda dinamica.

public int OpenOptions {get; set;}
OpenOptions vengono impostati quando un oggetto IBM MQ viene aperto. Il metodo `OpenOptions.set` viene ignorato e non causa un errore. Le sottoscrizioni non hanno `OpenOptions`.

Metodi

public virtual void Close();
Genera `MQException`.

Chiude l'oggetto. Non sono consentite ulteriori operazioni su questa risorsa dopo aver richiamato `Close`. Per modificare il comportamento del metodo `Close` , impostare l'attributo `closeOptions` .

public string GetAttributeString(int *selector*, int *length*);
Genera `MQException`.

Richiama una stringa attributo.

selettore
Numero intero che indica quale attributo viene interrogato.

lunghezza
Numero intero che indica la lunghezza della stringa richiesta.

public void Inquire(int[] *selectors*, int[] *intAttrs*, byte[] *charAttrs*);
Genera `MQException`.

Restituisce un array di numeri interi e una serie di stringhe di caratteri che contengono gli attributi di una coda, di un processo o di un gestore code. Gli attributi da interrogare sono specificati nell'array dei selettori.

Nota: Molti degli attributi più comuni possono essere interrogati utilizzando i metodi `Get` definiti in `MQManagedObject`, `MQQueue` e `MQQueueManager`.

Selettori
Array di numeri interi che identificano gli attributi con i valori da interrogare.

intAttrs
L'array in cui vengono restituiti i valori dell'attributo integer. I valori di attributo integer vengono restituiti nello stesso ordine dei selettori di attributo integer nell'array di selettori.

charAttrs
Il buffer in cui vengono restituiti gli attributi carattere, concatenati. Gli attributi carattere vengono restituiti nello stesso ordine dei selettori di attributi carattere nell'array di selettori. La lunghezza di ciascuna stringa attributo è fissa per ciascun attributo.

public void Set(int[] *selectors*, int[] *intAttrs*, byte[] *charAttrs*);
Genera `MQException`.

Imposta gli attributi definiti nel vettore dei selettori. Gli attributi da impostare sono specificati nell'array selettori.

Selettori
Array di numeri interi che identificano gli attributi con valori da impostare.

intAttrs
L'array di valori attributo interi da impostare. Questi valori devono essere nello stesso ordine dei selettori di attributi interi nell'array di selettori.

charAttrs
Il buffer in cui sono concatenati gli attributi carattere da impostare. Questi valori devono essere nello stesso ordine dei selettori di attributi di caratteri nell'array di selettori. La lunghezza di ciascun attributo carattere è fissa.

public void SetAttributeString(int *selector*, string *value*, int *length*);
Genera MQException.

Imposta una stringa attributo.

selettore
Numero intero che indica quale attributo viene impostato.

valore
La stringa da impostare come valore dell'attributo.

lunghezza
Numero intero che indica la lunghezza della stringa richiesta.

Costruttori

protected MQManagedObject()
Metodo costruttore. Questo oggetto è una classe di base astratta che non può essere istanziata da sola.

Classe MQMessage.NET

Utilizzare MQMessage per accedere al descrittore del messaggio e ai dati per un messaggio IBM MQ . MQMessage incapsula un messaggio IBM MQ .

Classe

```
System.Object
        |
        └ IBM.WMQ.MQBase
                |
                └ IBM.WMQ.MQBaseObject
                        |
                        └ IBM.WMQ.MQMessage
```

```
public class IBM.WMQ.MQMessage extends IBM.WMQ.MQBaseObject;
```

Creare un oggetto MQMessage , quindi utilizzare i metodi Read e Write per trasferire i dati tra il messaggio e altri oggetti nell'applicazione. Inviare e ricevere oggetti MQMessage utilizzando i metodi Put e Get delle classi MQDestination, MQQueue e MQTopic .

Richiamare e impostare le proprietà del descrittore del messaggio utilizzando le proprietà di MQMessage. Impostare e richiamare le proprietà dei messaggi estesi utilizzando i metodi SetProperty e GetProperty .

- “Proprietà” a pagina 1783
- “Metodi del messaggio Read e Write” a pagina 1789
- “Metodi buffer” a pagina 1792
- “Metodi proprietà” a pagina 1793
- “Costruttori” a pagina 1795

Proprietà

Test per MQException generato durante il richiamo delle proprietà.

public string AccountingToken {get; set;}
Parte del contesto di identità del messaggio; aiuta un'applicazione ad addebitare il lavoro eseguito come risultato del messaggio. Il valore predefinito è `MQC.MQACT_NONE`.

public string ApplicationIdData {get; set;}
Parte del contesto di identità del messaggio. `ApplicationIdI dati` sono informazioni definite dalla suite di applicazioni e possono essere utilizzate per fornire ulteriori informazioni sul messaggio o sul suo creatore. Il valore predefinito è `""`.

public string ApplicationOriginData {get; set;}
Informazioni definite dall'applicazione che possono essere utilizzate per fornire ulteriori informazioni sull'origine del messaggio. Il valore predefinito è `""`.

public int BackoutCount {get;}
Un conteggio del numero di volte in cui il messaggio è stato precedentemente restituito e ripristinato da una chiamata `MQQueue.Get` come parte di un'unità di lavoro. Il valore predefinito è zero.

public int CharacterSet {get; set;}

Il CCSID (coded character set identifier) dei dati carattere nel messaggio.

Impostare `CharacterSet` per identificare la serie di caratteri dei dati del messaggio. Richiamare `CharacterSet` per individuare la serie di caratteri utilizzata per codificare i dati dei caratteri nel messaggio.

Le applicazioni .NET vengono sempre eseguite in Unicode, mentre in altri ambienti le applicazioni vengono eseguite nella stessa serie di caratteri con cui viene eseguito il gestore code.

I metodi `ReadString` e `ReadLine` convertono i dati carattere nel messaggio in Unicode.

Il metodo `WriteString` esegue la conversione da Unicode alla serie di caratteri codificata in `CharacterSet`. Se `CharacterSet` è impostato sul valore predefinito, `MQC.MQCCSI_Q_MGR`, che è 0, non viene eseguita alcuna conversione e `CharacterSet` viene impostato su 1200. Se si imposta `CharacterSet` su un altro valore, `WriteString` converte da Unicode al valore alternativo.

Nota: Altri metodi di lettura e scrittura non utilizzano `CharacterSet`.

- `ReadChar` e `WriteChar` leggono e scrivono un carattere Unicode nel e dal buffer di messaggi senza conversione.
- `ReadUTF` e `WriteUTF` convertono tra una stringa Unicode nell'applicazione e una stringa UTF-8 , preceduta da un campo di lunghezza di 2 byte, nel buffer di messaggi.
- I metodi byte trasferiscono i byte tra l'applicazione e il buffer di messaggi senza alcuna modifica.

public byte[] CorrelationId {get; set;}

- Per una chiamata `MQQueue.Get` , l'identificativo di correlazione del messaggio da richiamare. Il gestore code restituisce il primo messaggio con un identificativo del messaggio e un identificativo di correlazione che corrispondono ai campi del descrittore del messaggio. Il valore predefinito, `MQC.MQCI_NONE`, consente la corrispondenza di qualsiasi identificativo di correlazione.
- Per una chiamata `MQQueue.Put` , l'identificativo di correlazione da impostare.

public int DataLength {get;}
Il numero di byte dei dati del messaggio rimanenti da leggere.

public int DataOffset {get; set;}
La posizione corrente del cursore all'interno dei dati del messaggio. Le letture e le scritture hanno effetto nella posizione corrente.

public int Encoding {get; set;}
La rappresentazione utilizzata per i valori numerici nei dati del messaggio dell'applicazione. La `codifica` si applica ai dati binari, decimali compressi e a virgola mobile. Il comportamento dei metodi di lettura e scrittura per questi formati numerici viene modificato di conseguenza. Costruire un valore per il campo di codifica aggiungendo un valore da ognuna di queste tre sezioni. In alternativa, creare il valore che combina i valori di ciascuna delle tre sezioni utilizzando l'operatore OR bitwise.

1. Numero intero binario

MQC.MQENC_INTEGER_NORMAL
Numeri interi Big - endian.

MQC.MQENC_INTEGER_REVERSED
Numeri interi little - endian, utilizzati nell'architettura Intel .

2. Decimale compresso

MQC.MQENC_DECIMAL_NORMAL
Decimale compresso Big - endian, come utilizzato da z/OS.

MQC.MQENC_DECIMAL_REVERSED
Decimale compresso Little - endian.

3. Virgola mobile

MQC.MQENC_FLOAT_IEEE_NORMAL
Galleggianti IEEE Big - endian.

MQC.MQENC_FLOAT_IEEE_REVERSED
Little - endian IEEE è mobile, come architettura Intel utilizzata.

MQC.MQENC_FLOAT_S390
z/OS formattare le virgola mobile.

Il valore predefinito è:

```
MQC.MQENC_INTEGER_REVERSED |
MQC.MQENC_DECIMAL_REVERSED |
MQC.MQENC_FLOAT_IEEE_REVERSED
```

L'impostazione predefinita fa sì che `WriteInt` scriva un numero intero little - endian e `ReadInt` legga un numero intero little - endian. Se invece si imposta l'indicatore `MQC.MQENC_INTEGER_NORMAL` , `WriteInt` scrive un numero intero big - endian e `ReadInt` legge un numero intero big - endian.

Nota: Una perdita di precisione può verificarsi durante la conversione da punti mobili in formato IEEE a punti mobili in formato zSeries .

public int Expiry {get; set;}
Un tempo di scadenza espresso in decimi di secondo, impostato dall'applicazione che inserisce il messaggio. Una volta trascorso il tempo di scadenza di un messaggio, è idoneo per essere eliminato dal gestore code. Se il messaggio ha specificato uno degli indicatori `MQC.MQRO_EXPIRATION` , viene generato un report quando il messaggio viene eliminato. Il valore predefinito è `MQC.MQEI_UNLIMITED`, il che significa che il messaggio non scade mai.

public int Feedback {get; set;}
Utilizzare `Feedback` con un messaggio di tipo `MQC.MQMT_REPORT` per indicare la natura del report. I seguenti codici di feedback sono definiti dal sistema:

- `MQC.MQFB_EXPIRATION`
- `MQC.MQFB_COA`
- `MQC.MQFB_COD`
- `MQC.MQFB_QUIT`
- `MQC.MQFB_PAN`
- `MQC.MQFB_NAN`
- `MQC.MQFB_DATA_LENGTH_ZERO`
- `MQC.MQFB_DATA_LENGTH_NEGATIVE`
- `MQC.MQFB_DATA_LENGTH_TOO_BIG`
- `MQC.MQFB_BUFFER_OVERFLOW`
- `MQC.MQFB_LENGTH_OFF_BY_ONE`
- `MQC.MQFB_IIH_ERROR`

È possibile utilizzare anche i valori di feedback definiti dall'applicazione compresi nell'intervallo tra MQC.MQFB_APPL_FIRST e MQC.MQFB_APPL_LAST . Il valore predefinito di questo campo è MQC.MQFB_NONE, che indica che non viene fornito alcun feedback.

public string Format {get; set;}
Un nome formato utilizzato dal mittente del messaggio per indicare la natura dei dati nel messaggio al destinatario. È possibile utilizzare i propri nomi formato, ma i nomi che iniziano con le lettere MQ hanno significati definiti dal gestore code. I formati integrati del gestore code sono:

MQC.MQFMT_ADMIN
Messaggio del server di comando di richiesta/risposta.

MQC.MQFMT_COMMAND_1
Messaggio di replica di comando tipo 1.

MQC.MQFMT_COMMAND_2
Messaggio di risposta del comando di tipo 2.

MQC.MQFMT_DEAD_LETTER_HEADER
Intestazione non instradabile.

MQC.MQFMT_EVENT
Messaggio evento.

MQC.MQFMT_NONE
Nessun nome formato.

MQC.MQFMT_PCF
Messaggio definito dall'utente in formato di comando programmabile.

MQC.MQFMT_STRING
Messaggio composto interamente da caratteri.

MQC.MQFMT_TRIGGER
messaggio di trigger

MQC.MQFMT_XMIT_Q_HEADER
Intestazione della coda di trasmissione.

Il valore predefinito è MQC.MQFMT_NONE.

public byte[] GroupId {get; set;}
Una stringa di byte che identifica il gruppo di messaggi a cui appartiene il messaggio fisico. Il valore predefinito è MQC.MQGI_NONE.

public int MessageFlags {get; set;}
Indicatori che controllano la segmentazione e lo stato di un messaggio.

public byte[] MessageId {get; set;}

Per una chiamata MQQueue.Get , questo campo specifica l'identificativo del messaggio da richiamare. Normalmente, il gestore code restituisce il primo messaggio con un identificativo del messaggio e un identificativo di correlazione che corrispondono ai campi del descrittore del messaggio. Consentire la corrispondenza di qualsiasi identificativo di messaggio utilizzando il valore speciale MQC.MQMI_NONE.

Per una chiamata MQQueue.Put , questo campo specifica l'identificativo del messaggio da utilizzare. Se MQC.MQMI_NONE sono specificati, il gestore code genera un identificativo di messaggio univoco quando il messaggio viene inserito. Il valore di questa variabile membro viene aggiornato dopo l'inserimento per indicare l'identificativo del messaggio che è stato utilizzato. Il valore predefinito è MQC.MQMI_NONE.

public int MessageLength {get;}
Il numero di byte dei dati del messaggio nell'oggetto MQMessage .

public int MessageSequenceNumber {get; set;}
Il numero di sequenza di un messaggio logico all'interno di un gruppo.

public int MessageType {get; set;}
Indica il tipo di messaggio. I seguenti valori sono attualmente definiti dal sistema:

- MQC.MQMT_DATAGRAM
- MQC.MQMT_REPLY
- MQC.MQMT_REPORT
- MQC.MQMT_REQUEST

I valori definiti dall'applicazione possono essere utilizzati anche nell'intervallo compreso tra MQC.MQMT_APPL_FIRST e MQC.MQMT_APPL_LAST. Il valore predefinito di questo campo è MQC.MQMT_DATAGRAM.

public int Offset {get; set;}
In un messaggio segmentato, l'offset dei dati in un messaggio fisico dall'avvio di un messaggio logico.

public int OriginalLength {get; set;}
La lunghezza originale di un messaggio segmentato.

public int Persistence {get; set;}

Durata del messaggio. Vengono definiti i seguenti valori:

- MQC.MQPER_NOT_PERSISTENT

  Se si imposta questa opzione in un client ricollegabile, il codice di errore MQRC_NONE viene restituito all'applicazione quando la connessione ha esito positivo.

- MQC.MQPER_PERSISTENT

  Se si imposta questa opzione in un client ricollegabile, il codice di errore MQRC_CALL_INTERRUPTED viene restituito all'applicazione dopo che la connessione ha avuto esito positivo.

- MQC.MQPER_PERSISTENCE_AS_Q_DEF

Il valore predefinito è MQC.MQPER_PERSISTENCE_AS_Q_DEF, che prende la persistenza per il messaggio dall'attributo di persistenza predefinito della coda di destinazione.

public int Priority {get; set;}
La priorità del messaggio. Il valore speciale MQC.MQPRI_PRIORITY_AS_Q_DEF può essere impostato anche nel messaggio in uscita. La priorità per il messaggio viene quindi presa dall'attributo di priorità predefinito della coda di destinazione. Il valore predefinito è MQC.MQPRI_PRIORITY_AS_Q_DEF.

public int PropertyValidation {get; set;}
Specifica se la convalida delle proprietà ha luogo quando viene impostata una proprietà del messaggio. I possibili valori sono:

- MQCMHO_DEFAULT_VALIDATION
- MQCMHO_VALIDATE
- MQCMHO_NO_VALIDATION

Il valore predefinito è MQCMHO_DEFAULT_VALIDATION.

public string PutApplicationName {get; set;}
Il nome dell'applicazione che ha inserito il messaggio. Il valore predefinito è "".

public int PutApplicationType {get; set;}
Il tipo di applicazione che ha inserito il messaggio. PutApplicationIl tipo può essere un valore definito dal sistema o dall'utente. I seguenti valori sono definiti dal sistema:

- MQC.MQAT_AIX
- MQC.MQAT_CICS
- MQC.MQAT_DOS
- MQC.MQAT_IMS
- MQC.MQAT_MVS
- MQC.MQAT_OS2
- MQC.MQAT_OS400

- MQC.MQAT_QMGR
- MQC.MQAT_UNIX
- MQC.MQAT_WINDOWS
- MQC.MQAT_JAVA

Il valore predefinito è MQC.MQAT_NO_CONTEXT, che indica che non sono presenti informazioni di contesto nel messaggio.

public DateTime PutDateTime {get; set;}
La data e l'ora in cui è stato inserito il messaggio.

public string ReplyToQueueManagerName {get; set;}
Il nome del gestore code per inviare messaggi di risposta o di report. Il valore predefinito è ""e il gestore code fornisce il nome ReplyToQueueManager.

public string ReplyToQueueName {get; set;}
Il nome della coda messaggi a cui l'applicazione che ha emesso la richiesta get per il messaggio invia i messaggi MQC.MQMT_REPLY e MQC.MQMT_REPORT . Il valore predefinito ReplyToQueueName è "".

public int Report {get; set;}
Utilizzare Report per specificare le opzioni relative ai messaggi di risposta e di report:

- Se i report sono obbligatori.
- Indica se i dati del messaggio dell'applicazione devono essere inclusi nei report.
- Come impostare il messaggio e gli identificativi di correlazione nel report o nella risposta.

Qualsiasi combinazione dei quattro tipi di report può essere richiesta:

- Specificare qualsiasi combinazione dei quattro tipi di report. La selezione di una delle tre opzioni per ciascun tipo di report, a seconda se i dati del messaggio dell'applicazione devono essere inclusi nel messaggio del report.

  1. Conferma all'arrivo
     - MQC.MQRO_COA
     - MQC.MQRO_COA_WITH_DATA
     - MQC.MQRO_COA_WITH_FULL_DATA **
  2. Conferma alla consegna
     - MQC.MQRO_COD
     - MQC.MQRO_COD_WITH_DATA
     - MQC.MQRO_COD_WITH_FULL_DATA **
  3. Eccezione
     - MQC.MQRO_EXCEPTION
     - MQC.MQRO_EXCEPTION_WITH_DATA
     - MQC.MQRO_EXCEPTION_WITH_FULL_DATA **
  4. Scadenza
     - MQC.MQRO_EXPIRATION
     - MQC.MQRO_EXPIRATION_WITH_DATA
     - MQC.MQRO_EXPIRATION_WITH_FULL_DATA **

  Nota: I valori contrassegnati con ** nell'elenco non sono supportati dai gestori code z/OS . Non utilizzarli se è probabile che l'applicazione acceda a un gestore code z/OS , indipendentemente dalla piattaforma su cui è in esecuzione l'applicazione.
- Specificare una delle seguenti opzioni per controllare il modo in cui viene generato l'ID messaggio per il report o il messaggio di risposta:
  - MQC.MQRO_NEW_MSG_ID

- MQC.MQRO_PASS_MSG_ID

- Specificare una delle seguenti opzioni per controllare come deve essere impostato l'ID di correlazione del messaggio di risposta o del report:
  - MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID
  - MQC.MQRO_PASS_CORREL_ID
- Specificare una delle seguenti opzioni per controllare la disposizione del messaggio originale quando non può essere consegnato alla coda di destinazione:
  - MQC.MQRO_DEAD_LETTER_Q
  - MQC.MQRO_DISCARD_MSG **
- Se non viene specificata alcuna opzione di report, il valore predefinito è:

```
MQC.MQRO_NEW_MSG_ID |
MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID |
MQC.MQRO_DEAD_LETTER_Q
```

- È possibile specificare uno o entrambi i seguenti valori per richiedere che l'applicazione ricevente invii un messaggio di azione positiva o di report di azione negativa.
  - MQC.MQRO_PAN
  - MQC.MQRO_NAN

public int TotalMessageLength {get;}
Il numero totale di byte nel messaggio come memorizzati nella coda messaggi da cui è stato ricevuto questo messaggio.

public string UserId {get; set;}
UserId fa parte del contesto di identità del messaggio. Il gestore code in genere fornisce il valore. È possibile sovrascrivere il valore se si dispone dell'autorizzazione per impostare il contesto di identità.

public int Version {get; set;}
La versione della struttura MQMD in uso.

Metodi del messaggio Read e Write

I metodi Read e Write eseguono le stesse funzioni dei membri delle classi BinaryReader e BinaryWriter nello spazio dei nomi .NET System.IO . Consultare MSDN per la sintassi del linguaggio completo e gli esempi di utilizzo. I metodi di lettura o scrittura dalla posizione corrente nel buffer dei messaggi. Spostano la posizione corrente in avanti in base al numero di byte letti o scritti.

Nota: Se i dati del messaggio contengono un'intestazione MQRFH o MQRFH2 , è necessario utilizzare il metodo ReadBytes per leggere i dati.

- Tutti i metodi generano IOException.
- I metodi ReadFully ridimensionano automaticamente l'array di destinazione byte o sbyte per adattarlo esattamente al messaggio. Viene ridimensionato anche un array null.
- Read metodi throw EndOfStreamException.
- WriteDecimal metodi throw MQException.
- I metodi ReadString, ReadLine e WriteString vengono convertiti tra Unicode e la serie di caratteri del messaggio; consultare CharacterSet .
- I metodi Decimal leggono e scrivono numeri decimali compressi codificati in formato big - endian, MQC.MQENC_DECIMAL_NORMALo little - endian MQC.MQENC_DECIMAL_REVERSE , in base al valore di Codifica. Gli intervalli decimali e i tipi .NET corrispondenti sono i seguenti:

  Decimal2/short
  Da -999 a 999

  Decimal4/int
  Da -9999999 a 9999999

Decimal8/long
Da -999999999999999 a 999999999999999

- I metodi `Double` e `Float` leggono e scrivono i valori mobili codificati nei formati IEE big - endian e little - endian, `MQC.MQENC_FLOAT_IEEE_NORMAL` e `MQC.MQENC_FLOAT_IEEE_REVERSED`, o in formato S/390 , `MQC.MQENC_FLOAT_S390`, in base al valore di `Codifica`.
- I metodi `Int` leggono e scrivono valori interi codificati in formato big - endian, `MQC.MQENC_INTEGER_NORMAL`, o little - endian, `MQC.MQENC_INTEGER_REVERSED`, in base al valore di `Codifica`. I numeri interi sono tutti con segno, ad eccezione dell'aggiunta di un tipo di numero intero a 2 byte senza segno. Le dimensioni dei numeri interi e i tipi .NET e IBM MQ sono i seguenti:

  2 byte
  `short`, `Int2`, `ushort`, `UInt2`

  4 byte
  `int`, `Int4`

  8 byte
  `long`, `Int8`
- `WriteObject` trasferisce al buffer di messaggi la classe di un oggetto, i valori dei relativi campi non transitori e non statici e i campi dei relativi supertipi.
- `ReadObject` crea un oggetto dalla classe dell'oggetto, la firma della classe, i valori dei campi non transitori e non statici e i campi dei relativi supertipi.

Tabella 843. Metodi di lettura e scrittura dei messaggi

Tipo di destinazione	Firme metodo

Boolean	`public bool ReadBoolean();`
	
	`public void WriteBoolean(bool value);`

Byte	`public byte ReadByte()`
	`public byte ReadUnsignedByte()`
	
	`public void Write(int value)`
	`public void WriteByte(int value)`
	`public void WriteByte(byte value)`
	`public void WriteByte(sbyte value)`

Bytes	`public byte[] ReadBytes(int count)`
	`public void ReadFully(ref byte[] value)`
	`public void ReadFully(ref sbyte[] value)`
	`public void ReadFully(ref byte[] value, int offset,int length)`
	`public void ReadFully(ref sbyte[] value, int offset,int length)`
	
	`public void Write(byte[] value)`
	`public void Write(sbyte[] value)`
	`public void Write(byte[] value, int offset,int length)`
	`public void Write(sbyte[] value, int offset,int length)`
	`public void WriteBytes(string value)`

Decimal2	`public void WriteDecimal2(short value)`

Tabella 843. Metodi di lettura e scrittura dei messaggi (Continua)

Tipo di destinazione	Firme metodo
Decimal4	`public void WriteDecimal4(short value)`
Decimal8	`public void WriteDecimal8(short value)`
Double	`public double ReadDouble()` `public void WriteDouble(double value)`
Float	`public float ReadFloat()` `public void WriteFloat(float value)`
Int2	`public void WriteInt2(int value)`

Int4	`public int readDecimal4()`
	`public int ReadInt()`
	`public int ReadInt4()`
	`public void WriteInt(int value)`
	`public void WriteInt4(int value)`

Int8	`public void WriteInt8(long value)`

Long	`public long ReadDecimal8()`
	`public long ReadLong()`
	`public long ReadInt8()`
	`public void WriteLong(long value)`

Object	`public Object ReadObject()` `public void WriteObject(Object object)`

Tabella 843. Metodi di lettura e scrittura dei messaggi (Continua)	
Tipo di destinazione	**Firme metodo**

Short	`public short ReadShort()`
	`public short ReadDecimal2()`
	`public short ReadInt2()`
	
	`public void WriteShort(int value)`

string	`public string ReadString(int length)`
	
	`public void WriteString(string string)`

Unsigned Short	`public ushort ReadUnsignedShort()` `public ushort ReadUInt2()`

Unicode	`public string ReadLine()`
	`public char ReadChar()`
	
	`public void WriteChar(int value)`
	`public void WriteChars(string string)`

UTF	`public string ReadUTF()`
	
	`public void WriteUTF(string string)`

Metodi buffer

public void ClearMessage();

Genera IOException.

Elimina tutti i dati nel buffer di messaggi e imposta di nuovo lo scostamento dei dati su zero.

public void ResizeBuffer(int *size*)

Genera IOException.

Un suggerimento all'oggetto MQMessage sulla dimensione del buffer che potrebbe essere richiesto per le successive operazioni get. Se il messaggio attualmente contiene dati del messaggio e la nuova dimensione è inferiore alla dimensione corrente, i dati del messaggio vengono troncati.

public void Seek(int *pos*)

Genera IOException, ArgumentOutOfRangeException, ArgumentException.

Sposta il cursore nella posizione assoluta nel buffer di messaggi fornito da *pos*. Le letture e scritture successive agiscono in questa posizione nel buffer.

public int SkipBytes(int *i*)

Genera IOException, EndOfStreamException.

Sposta in avanti n byte nel buffer dei messaggi e restituisce n, il numero di byte ignorati.

Il metodo SkipBytes si blocca fino a quando si verifica uno dei seguenti eventi:

- Tutti i byte vengono ignorati
- È stata rilevata la fine del buffer di messaggi
- È stata generata un'eccezione

Metodi proprietà

public void DeleteProperty(string *name*);

Genera MQException.

Elimina una proprietà con il nome specificato dal messaggio.

nome
Il nome della proprietà da eliminare.

public System.Collections.IEnumerator GetPropertyNames(string *name*)

Genera MQException.

Restituisce un IEnumerator di tutti i nomi di proprietà corrispondenti al nome specificato. Il simbolo di percentuale '%' può essere utilizzato alla fine del nome come carattere jolly per filtrare le proprietà del messaggio, corrispondenti a zero o a più caratteri, incluso il punto.

nome
Il nome della proprietà per la corrispondenza.

Metodi SetProperty e GetProperty

Tutti i metodi SetProperty e GetProperty generano MQException.

Il metodo SetProperty della classe MQMessage .NET aggiunge una nuova proprietà se una proprietà non esiste già. Tuttavia, se la proprietà esiste già, il valore della proprietà fornito viene aggiunto alla fine dell'elenco. Quando più valori vengono impostati su un nome proprietà utilizzando SetProperty, la chiamata GetProperty per tale nome restituisce tali valori in modo sequenziale nell'ordine in cui sono stati impostati.

Il comportamento è lo stesso per tutti i metodi di tipo Set*Property e Get*Property come GetLongProperty, SetLongProperty, GetBooleanProperty, SetBooleanProperty, GetStringPropertye SetStringProperty.

Tabella 844. Metodi SetProperty e GetProperty

Tipo	Firme metodo

Boolean	`public boolean GetBooleanProperty(string name);`
	`public boolean GetBooleanProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetBooleanProperty(string name, boolean value);`
	`public void SetBooleanProperty(string name, MQPropertyDescriptor pd, boolean value);`

Tabella 844. Metodi `SetProperty` e `GetProperty` (Continua)

Tipo	Firme metodo

Byte	`public sbyte GetByteProperty(string name);`
	`public sbyte GetByteProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetByteProperty(string name, sbyte value);`
	`public void SetByteProperty(string name, MQPropertyDescriptor pd, sbyte value);`

Bytes	`public sbyte[] GetBytesProperty(string name);`
	`public sbyte[] GetBytesProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetBytesProperty(string name, sbyte[] value);`
	`public void SetBytesProperty(string name, MQPropertyDescriptor pd, sbyte[] value);`

Double	`public double GetDoubleProperty(string name);`
	`public double GetDoubleProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetDoubleProperty(string name, double value);`
	`public void SetDoubleProperty(string name, MQPropertyDescriptor pd, double value);`

Float	`public float GetFloatProperty(string name);`
	`public float GetFloatProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetFloatProperty(string name, float value);`
	`public void SetFloatProperty(string name, MQPropertyDescriptor pd, float value);`

Int2	`public short GetInt2Property(string name);`
	`public short GetInt2Property(string name, MQPropertyDescriptor pd);`
	
	`public void SetInt2Property(string name, short value);`
	`public void SetInt2Property(string name, MQPropertyDescriptor pd, short value);`

Int4	`public int GetInt4Property(string name);`
	`public int GetInt4Property(string name, MQPropertyDescriptor pd);`
	
	`public void SetInt4Property(string name, int value);`
	`public void SetInt4Property(string name, MQPropertyDescriptor pd, int value);`

Int8	`public long GetInt8Property(string name);`
	`public long GetInt8Property(string name, MQPropertyDescriptor pd);`
	
	`public void SetInt8Property(string name, long value);`
	`public void SetInt8Property(string name, MQPropertyDescriptor pd, long value);`

Tabella 844. Metodi SetProperty e GetProperty (Continua)

Tipo	Firme metodo

Long	`public long GetLongProperty(string name);`
	`public long GetLongProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetLongProperty(string name, long value);`
	`public void SetLongProperty(string name, MQPropertyDescriptor pd, long value);`

Object	`public Object GetObjectProperty(string name);`
	`public Object GetObjectProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetObjectProperty(string name, Object value);`
	`public void SetObjectProperty(string name, MQPropertyDescriptor pd, Object value);`

Short	`public short GetShortProperty(string name);`
	`public short GetShortProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetShortProperty(string name, short value);`
	`public void SetShortProperty(string name, MQPropertyDescriptor pd, short value);`

string	`public string GetStringProperty(string name);`
	`public string GetStringProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetStringProperty(string name, string value);`
	`public void SetStringProperty(string name, MQPropertyDescriptor pd, string value);`

Costruttori

public MQMessage();
Crea un oggetto MQMessage con informazioni sul descrittore di messaggi predefinito e un buffer di messaggi vuoto.

Classe MQProcess.NET

Utilizzare MQProcess per interrogare gli attributi di un processo IBM MQ . Creare un oggetto MQProcess utilizzando un costruttore o un metodo MQQueueManager AccessProcess .

Classe

```
System.Object
        |
        └─ IBM.WMQ.MQBase
                |
                └─ IBM.WMQ.MQBaseObject
                        |
                        └─ IBM.WMQ.MQManagedObject
                                |
                                └─ IBM.WMQ.MQProcess
```

```
public class IBM.WMQ.MQProcess extends IBM.WMQ.MQManagedObject;
```

Proprietà

Test per MQException generato durante il richiamo delle proprietà.

public string ApplicationId {get;}

Richiama la stringa di caratteri che identifica l'applicazione da avviare. ApplicationId viene utilizzato da un'applicazione di controllo trigger. ApplicationId viene inviato alla coda di iniziazione come parte del messaggio trigger.

Il valore predefinito è null.

public int ApplicationType {get;}

Identifica il tipo di processo che deve essere avviato da un'applicazione di controllo trigger. I tipi standard sono definiti, ma altri possono essere utilizzati:

- MQAT_AIX
- MQAT_CICS
- MQAT_IMS
- MQAT_MVS
- MQAT_NATIVE
- MQAT_OS400
- MQAT_UNIX
- MQAT_WINDOWS
- MQAT_JAVA
- MQAT_USER_FIRST
- MQAT_USER_LAST

Il valore predefinito è MQAT_NATIVE.

public string EnvironmentData {get;}

Ottiene informazioni sull'ambiente dell'applicazione da avviare.

Il valore predefinito è null.

public string UserData {get;}

Ottiene le informazioni fornite dall'utente sull'applicazione da avviare.

Il valore predefinito è null.

Costruttori

public MQProcess(MQQueueManager *queueManager*, string *processName*, int *openOptions*);
public MQProcess(MQQueueManager *qMgr*, string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Genera MQException.

Accedere a un processo IBM MQ sul gestore code *qMgr* per analizzare gli attributi del processo.

qMgr
Gestore code a cui accedere.

processName
Il nome del processo da aprire.

openOptions
Opzioni che controllano l'apertura del processo. Le opzioni valide che possono essere aggiunte o combinate utilizzando un OR bitwise sono:

- MQC.MQOO_FAIL_IF_QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName
Il nome del gestore code su cui è definito il processo. È possibile lasciare un nome gestore code vuoto o null se il gestore code è lo stesso a cui accede il processo.

AlternateUserid

Se MQC.MQOO_ALTERNATE_USER_AUTHORITY è specificato nel parametro **openOptions** , *alternateUserId* specifica l'ID utente alternativo utilizzato per controllare l'autorizzazione per l'azione. Se MQOO_ALTERNATE_USER_AUTHORITY non è specificato, *alternateUserId* può essere vuoto o null.

L'autorizzazione utente predefinita viene utilizzata per la connessione al gestore code se MQC.MQOO_ALTERNATE_USER_AUTHORITY non è specificato.

public MQProcess MQQueueManager.AccessProcess(string *processName*, int *openOptions*);
public MQProcess MQQueueManager.AccessProcess(string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Genera MQException.

Accedere a un processo IBM MQ su questo gestore code per analizzare gli attributi del processo.

processName
Il nome del processo da aprire.

openOptions
Opzioni che controllano l'apertura del processo. Le opzioni valide che possono essere aggiunte o combinate utilizzando un OR bitwise sono:

- MQC.MQOO_FAIL_IF_QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName
Il nome del gestore code su cui è definito il processo. È possibile lasciare un nome gestore code vuoto o null se il gestore code è lo stesso a cui accede il processo.

AlternateUserid

Se MQC.MQOO_ALTERNATE_USER_AUTHORITY è specificato nel parametro **openOptions** , *alternateUserId* specifica l'ID utente alternativo utilizzato per controllare l'autorizzazione per l'azione. Se MQOO_ALTERNATE_USER_AUTHORITY non è specificato, *alternateUserId* può essere vuoto o null.

L'autorizzazione utente predefinita viene utilizzata per la connessione al gestore code se MQC.MQOO_ALTERNATE_USER_AUTHORITY non è specificato.

Classe MQPropertyDescriptor.NET

Utilizzare MQPropertyDescriptor come parametro per i metodi MQMessage GetProperty e SetProperty . MQPropertyDescriptor descrive una proprietà MQMessage .

Classe

```
System.Object
        │
        └─ IBM.WMQ.MQPropertyDescriptor
```

```
public class IBM.WMQ.MQPropertyDescriptor extends System.Object;
```

- “Proprietà” a pagina 1798
- “Costruttori” a pagina 1799

Proprietà

Test per `MQException` generato durante il richiamo delle proprietà.

`public int Context {get; set;}`

Il contesto del messaggio a cui appartiene la proprietà. I possibili valori sono:

MQC.MQPD_NO_CONTEXT
La proprietà non è associata a un contesto di messaggio.

MQC.MQPD_USER_CONTEXT
La proprietà è associata al contesto utente.

Se l'utente è autorizzato, una proprietà associata al contesto utente viene salvata quando viene richiamato un messaggio. Un metodo `Put` successivo che fa riferimento al contesto salvato, può passare la proprietà nel nuovo messaggio.

`public int CopyOptions {get; set;}`

`CopyOptions` descrive in quale tipo di messaggio è possibile copiare la proprietà.

Quando un gestore code riceve un messaggio contenente una IBM MQ proprietà definita che il gestore code riconosce come non corretta, corregge il valore del campo `CopyOptions` .

È possibile specificare qualsiasi combinazione delle seguenti opzioni. Combinare le opzioni aggiungendo i valori o utilizzando ORbitwise.

MQC.MQCOPY_ALL
La proprietà viene copiata in tutti i messaggi successivi.

MQC.MQCOPY_FORWARD
La proprietà viene copiata in un messaggio inoltrato.

MQC.MQCOPY_PUBLISH
La proprietà viene copiata nel messaggio ricevuto da un sottoscrittore quando viene pubblicato un messaggio.

MQC.MQCOPY_REPLY
La proprietà viene copiata in un messaggio di risposta.

MQC.MQCOPY_REPORT
La proprietà viene copiata in un messaggio di report.

MQC.MQCOPY_DEFAULT
Il valore indica che non sono state specificate altre opzioni di copia. Non esiste alcuna relazione tra la proprietà e i messaggi successivi. `MQC.MQCOPY_DEFAULT` viene sempre restituito per le proprietà del descrizione del messaggio.

MQC.MQCOPY_NONE
Uguale a `MQC.MQCOPY_DEFAULT`

`public int Options { set; }`
`Opzioni` assume il valore predefinito `CMQC.MQPD_NONE`. Non è possibile impostare nessun altro valore.

public int Support { get; set; }

Impostare Supporto per specificare il livello di supporto richiesto per le proprietà del messaggio definite da IBM MQ. Il supporto per tutte le altre proprietà è facoltativo. È possibile specificare uno o nessuno dei seguenti valori

MQC.MQPD_SUPPORT_OPTIONAL
La proprietà viene accettata da un gestore code anche se non è supportata. La proprietà può essere eliminata in modo che il messaggio possa fluire in un gestore code che non supporta le proprietà del messaggio. Questo valore viene assegnato anche alle proprietà non definite da IBM MQ .

MQC.MQPD_SUPPORT_REQUIRED
È richiesto il supporto per la proprietà. Se si inserisce il messaggio in un gestore code che non supporta la proprietà definita da IBM MQ, il metodo non riesce. Restituisce il codice di completamento MQC.MQCC_FAILED e il codice motivo MQC.MQRC_UNSUPPORTED_PROPERTY.

MQC.MQPD_SUPPORT_REQUIRED_IF_LOCAL

È richiesto il supporto per la proprietà, se il messaggio è destinato a una coda locale. Se si inserisce il messaggio in una coda locale su un gestore code che non supporta la proprietà definita da IBM MQ, il metodo non riesce. Restituisce il codice di completamento MQC.MQCC_FAILED e il codice motivo MQC.MQRC_UNSUPPORTED_PROPERTY.

Non viene eseguito alcun controllo se il messaggio viene inserito in un gestore code remoto.

Costruttori

PropertyDescriptor();
Creare un descrittore di proprietà.

Classe MQPutMessageOptions.NET

Utilizzare MQPutMessageOptions per specificare il modo in cui vengono inviati i messaggi. Modifica il funzionamento di MQDestination.Put.

Classe


```
public class IBM.WMQ.MQPutMessageOptions extends IBM.WMQ.MQBaseObject;
```

- “Proprietà” a pagina 1799 “Costruttori” a pagina 1802

Proprietà

Test per MQException generato durante il richiamo delle proprietà.

Nota: Il comportamento di alcune delle opzioni disponibili in questa classe dipende dall'ambiente in cui vengono utilizzate. Questi elementi sono contrassegnati con un asterisco, *.

public MQQueue ContextReference {get; set;}

Se il campo options include MQC.MQPMO_PASS_IDENTITY_CONTEXT o MQC.MQPMO_PASS_ALL_CONTEXT, impostare questo campo per fare riferimento al MQQueue da cui prendere le informazioni di contesto.

Il valore iniziale di questo campo è null.

public int InvalidDestCount {get;}*

Generalmente, utilizzato per elenchi di distribuzione, `InvalidDestCount` indica il numero di messaggi che non è stato possibile inviare alle code in un elenco di distribuzione. Il conteggio include le code che non è stato possibile aprire e anche le code che sono state aperte correttamente, ma per cui l'operazione di inserimento non è riuscita.

.NET non supporta gli elenchi di distribuzione, ma `InvalidDestCount` è impostato quando si apre una coda singola.

public int KnownDestCount {get;}*

Generalmente utilizzato per gli elenchi di distribuzione, `KnownDestCount` indica il numero di messaggi che la chiamata corrente ha inviato correttamente alle code che si risolvono in code locali.

.NET non supporta gli elenchi di distribuzione, ma `InvalidDestCount` è impostato quando si apre una coda singola.

public int Options {get; set;}

Opzioni che controllano l'azione di `MQDestination.put` e `MQQueueManager.put`. È possibile specificare uno o nessuno dei seguenti valori. Se è richiesta più di un'opzione, i valori possono essere aggiunti o combinati utilizzando l'operatore OR bit per bit.

MQC.MQPMO_ASYNC_RESPONSE
Questa opzione fa sì che la chiamata `MQDestination.put` venga effettuata in modo asincrono, con alcuni dati di risposta.

MQC.MQPMO_DEFAULT_CONTEXT
Associare il contesto predefinito al messaggio.

MQC.MQPMO_FAIL_IF_QUIESCING
Errore se il gestore code è in attesa.

MQC.MQPMO_LOGICAL_ORDER*
Inserire i messaggi logici e i segmenti nei gruppi di messaggi nel proprio ordine logico.

Se si utilizza l'opzione `MQPMO_LOGICAL_ORDER` in un client ricollegabile, il codice di errore `MQRC_RECONNECT_INCOMPATIBLE` viene restituito all'applicazione.

MQC.MQPMO_NEW_CORREL_ID*
Generare un nuovo ID correlazione per ogni messaggio inviato.

MQC.MQPMO_NEW_MSG_ID*
Creare un nuovo ID messaggio per ciascun messaggio inviato.

MQC.MQPMO_NONE
Nessuna opzione specificata. Non utilizzare con altre opzioni.

MQC.MQPMO_NO_CONTEXT
Nessun contesto deve essere associato al messaggio.

MQC.MQPMO_NO_SYNCPOINT
Inserire un messaggio senza il controllo del punto di sincronizzazione. Se l'opzione di controllo del punto di sincronizzazione non viene specificata, viene assunto un valore predefinito di nessun punto di sincronizzazione.

MQC.MQPMO_PASS_ALL_CONTEXT
Passare tutto il contesto da un handle di coda di immissione.

MQC.MQPMO_PASS_IDENTITY_CONTEXT
Passare il contesto di identità da un handle della coda di input.

MQC.MQPMO_RESPONSE_AS_Q_DEF

Per una chiamata `MQDestination.put` , questa opzione prende il tipo di risposta di inserimento dall'attributo DEFPRESP della coda.

Per una chiamata `MQQueueManager.put` , questa opzione fa sì che la chiamata venga effettuata in modo sincrono.

MQC.MQPMO_RESPONSE_AS_TOPIC_DEF
MQC.MQPMO_RESPONSE_AS_TOPIC_DEF è un sinonimo di MQC.MQPMO_RESPONSE_AS_Q_DEF da utilizzare con gli oggetti argomento.

MQC.MQPMO_RETAIN

La pubblicazione inviata deve essere conservata dal gestore code. Se questa opzione viene utilizzata e la pubblicazione non può essere conservata, il messaggio non viene pubblicato e la chiamata ha esito negativo con MQC.MQRC_PUT_NOT_RETAINED.

Richiedere una copia di questa pubblicazione dopo la sua pubblicazione, richiamando il metodo MQSubscription.RequestPublicationUpdate . La pubblicazione salvata viene inoltrata alle applicazioni che creano una sottoscrizione senza impostare l'opzione MQC.MQSO_NEW_PUBLICATIONS_ONLY . Controllare la proprietà del messaggio MQIsRetained di una pubblicazione, quando viene ricevuta, per verificare se si trattava della pubblicazione conservata.

Quando le pubblicazioni conservate vengono richieste da un sottoscrittore, la sottoscrizione utilizzata potrebbe contenere un carattere jolly nella stringa dell'argomento. Se ci sono più pubblicazioni conservate nella struttura ad albero degli argomenti che corrispondono alla sottoscrizione, vengono tutte inviate.

MQC.MQPMO_SET_ALL_CONTEXT
Impostare tutto il contesto dall'applicazione.

MQC.MQPMO_SET_IDENTITY_CONTEXT
Impostare il contesto di identità dall'applicazione.

MQC.MQPMO_SYNC_RESPONSE
Questa opzione fa sì che la chiamata MQDestination.put o MQQueueManager.put venga effettuata in modo sincrono, con dati di risposta completi.

MQC.MQPMO_SUPPRESS_REPLYTO
Le informazioni inserite nei campi ReplyToQueueName e ReplyToQueueManagerName della pubblicazione non vengono trasmesse ai sottoscrittori. Se questa opzione viene utilizzata in combinazione con un'opzione di report che richiede un ReplyToQueueName, la chiamata ha esito negativo con MQC.MQRC_MISSING_REPLY_TO_Q.

MQC.MQPMO_SYNCPOINT
Inserire un messaggio con il controllo del punto di sincronizzazione. Il messaggio non è visibile all'esterno dell'unità di lavoro fino a quando non viene eseguito il commit dell'unità di lavoro. Se viene eseguito il backout dell'unità di lavoro, il messaggio viene eliminato.

public int RecordFields {get; set;}*

Informazioni sugli elenchi di distribuzione. Gli elenchi di distribuzione non sono supportato in .NET.

public string ResolvedQueueManagerName {get;}

Un campo di output impostato dal gestore code sul nome del gestore code proprietario della coda specificata dal nome della coda remota. ResolvedQueueManagerName potrebbe essere diverso dal nome del gestore code da cui è stato effettuato l'accesso alla coda se la coda è una coda remota.

Un valore non vuoto viene restituito solo se l'oggetto è una singola coda. Se l'oggetto è un elenco di distribuzione o un argomento, il valore restituito non è definito.

public string ResolvedQueueName {get;}

Un campo di output impostato dal gestore code sul nome della coda in cui si trova il messaggio. ResolvedQueueIl nome potrebbe essere diverso dal nome utilizzato per aprire la coda se la coda aperta era un alias o una coda modello.

Un valore non vuoto viene restituito solo se l'oggetto è una singola coda. Se l'oggetto è un elenco di distribuzione o un argomento, il valore restituito non è definito.

public int UnknownDestCount {get;}*

Generalmente utilizzato per gli elenchi di distribuzione, `UnknownDestCount` è un campo di output impostato dal gestore code. Riporta il numero di messaggi che la chiamata corrente ha inviato correttamente alle code che si risolvono in code remote.

.NET non supporta gli elenchi di distribuzione, ma `InvalidDestCount` è impostato quando si apre una coda singola.

Costruttori

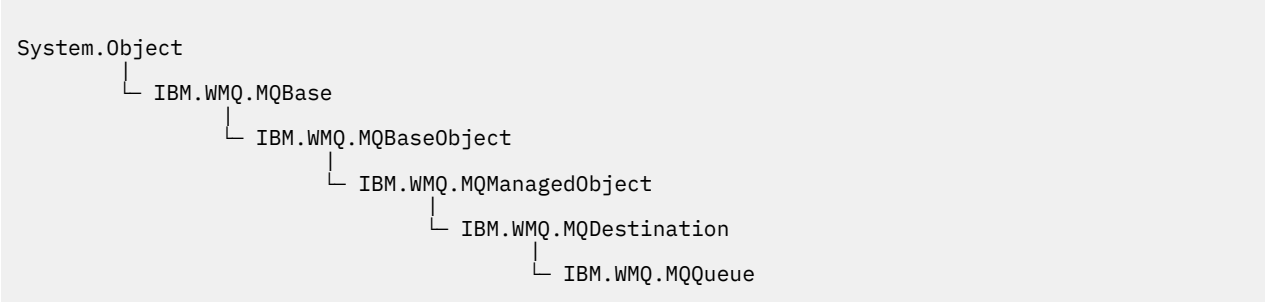
public MQPutMessageOptions();

Creare un nuovo oggetto `MQPutMessageOptions` senza opzioni impostate e un `ResolvedQueueName` vuoto e `ResolvedQueueManagerName`.

Classe MQQueue.NET

Utilizzare `MQQueue` per inviare e ricevere messaggi e interrogare gli attributi di una coda IBM MQ . Creare un oggetto `MQQueue` utilizzando un costruttore o un metodo `MQQueueManager.AccessProcess` .

Classe



`public class IBM.WMQ.MQQueue extends IBM.WMQ.MQDestination;`

- “Proprietà” a pagina 1802
- “Metodi” a pagina 1804
- “Costruttori” a pagina 1807

Proprietà

Test per `MQException` generato durante il richiamo delle proprietà.

public int ClusterWorkLoadPriority {get;}
Specifica la priorità della coda. Questo parametro è valido solo per code locali, remote e alias.

public int ClusterWorkLoadRank {get;}
Specifica la classificazione della coda. Questo parametro è valido solo per code locali, remote e alias.

public int ClusterWorkLoadUseQ {get;}
Specifica il comportamento di un'operazione MQPUT quando la coda di destinazione dispone di un'istanza locale e di almeno un'istanza cluster remoto. Questo parametro non si applica se MQPUT ha origine da un canale cluster. Questo parametro è valido solo per le code locali.

public DateTime CreationDateTime {get;}
La data e l'ora di creazione di questa coda.

public int CurrentDepth {get;}
Richiama il numero di messaggi attualmente in coda. Questo valore viene incrementato durante una chiamata put e durante il backout di una chiamata get. Viene ridotto durante un richiamo non - browse e durante il backout di una chiamata put.

public int DefinitionType {get;}
Come è stata definita la coda. I valori possibili sono:

- MQC.MQQDT_PREDEFINED
- MQC.MQQDT_PERMANENT_DYNAMIC
- MQC.MQQDT_TEMPORARY_DYNAMIC

public int InhibitGet {get; set;}
Controlla se è possibile richiamare i messaggi su questa coda o per questo argomento. I valori possibili sono:

- MQC.MQQA_GET_INHIBITED
- MQC.MQQA_GET_ALLOWED

public int InhibitPut {get; set;}
Controlla se è possibile inserire messaggi in questa coda o per questo argomento. I valori possibili sono:

- MQQA_PUT_INHIBITED
- MQQA_PUT_ALLOWED

public int MaximumDepth {get;}
Il numero massimo di messaggi che possono esistere sulla coda in qualsiasi momento. Un tentativo di inserire un messaggio in una coda che contiene già questo numero di messaggi ha esito negativo con il codice di errore MQC.MQRC_Q_FULL.

public int MaximumMessageLength {get;}
La lunghezza massima dei dati dell'applicazione che possono esistere in ogni messaggio su questa coda. Un tentativo di inserire un messaggio più grande di questo valore ha esito negativo con codice di errore MQC.MQRC_MSG_TOO_BIG_FOR_Q.

public int NonPersistentMessageClass {get;}
Il livello di affidabilità per i messaggi non persistenti inseriti in questa coda.

public int OpenInputCount {get;}
Il numero di handle attualmente validi per la rimozione dei messaggi dalla coda. OpenInputOpenInput è il numero totale di handle di input validi noti al gestore code locale, non solo gli handle creati dall'applicazione.

public int OpenOutputCount {get;}
Il numero di handle attualmente validi per aggiungere messaggi alla coda. OpenOutputOpenOutput è il numero totale di handle di emissione validi noti al gestore code locale, non solo gli handle creati dall'applicazione.

public int QueueAccounting {get;}
Specifica se è possibile abilitare la raccolta di informazioni di account per la coda.

public int QueueMonitoring {get;}
Specifica se è possibile abilitare il monitoraggio per la coda.

public int QueueStatistics {get;}
Specifica se è possibile abilitare la raccolta di statistiche per la coda.

public int QueueType {get;}
Il tipo di questa coda con uno dei seguenti valori:

- MQC.MQQT_ALIAS
- MQC.MQQT_LOCAL
- MQC.MQQT_REMOTE
- MQC.MQQT_CLUSTER

public int Shareability {get;}
Indica se la coda può essere aperta per l'input più volte. I valori possibili sono:

- MQC.MQQA_SHAREABLE

- MQC.MQQA_NOT_SHAREABLE

public string TPIPE {get;}
Il nome TPIPE utilizzato per la comunicazione con OTMA utilizzando il bridge IBM MQ IMS .

public int TriggerControl {get; set;}
Se i messaggi trigger vengono scritti in una coda di iniziazione, per avviare un'applicazione per servire la coda. I valori possibili sono:

- MQC.MQTC_OFF
- MQC.MQTC_ON

public string TriggerData {get; set;}
I dati in formato libero che il gestore code inserisce nel messaggio trigger. Inserisce TriggerData quando un messaggio che arriva su questa coda fa sì che un messaggio trigger venga scritto nella coda di iniziazione. La lunghezza massima consentita della stringa è fornita da MQC.MQ_TRIGGER_DATA_LENGTH.

public int TriggerDepth {get; set;}
Il numero di messaggi che devono essere sulla coda prima che un messaggio trigger venga scritto quando il tipo di trigger è impostato su MQC.MQTT_DEPTH.

public int TriggerMessagePriority {get; set;}
La priorità del messaggio con cui i messaggi non contribuiscono alla creazione dei messaggi trigger. In altre parole, il gestore code ignora questi messaggi quando si decide se generare un trigger. Un valore pari a zero fa sì che tutti i messaggi contribuiscano alla generazione dei messaggi trigger.

public int TriggerType {get; set;}
Le condizioni in cui i messaggi trigger vengono scritti come risultato dei messaggi in arrivo su questa coda. I valori possibili sono:

- MQC.MQTT_NONE
- MQC.MQTT_FIRST
- MQC.MQTT_EVERY
- MQC.MQTT_DEPTH

Metodi

public void Get(MQMessage *message*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*, int *MaxMsgSize*);

Genera MQException.

Richiama un messaggio da una coda.

Se il richiamo ha esito negativo, l'oggetto MQMessage non viene modificato. Se ha esito positivo, il descrittore del messaggio e le parti di dati del messaggio di MQMessage vengono sostituiti con il descrittore del messaggio e i dati del messaggio dal messaggio in arrivo.

Tutte le chiamate a IBM MQ da un particolare MQQueueManager sono sincrone. Pertanto, se si esegue un richiamo con attesa, a tutti gli altri thread che utilizzano lo stesso MQQueueManager viene impedito di effettuare ulteriori chiamate IBM MQ fino a quando non viene eseguita la chiamata Get. Se sono necessari più thread per accedere simultaneamente a IBM MQ , ogni thread deve creare il proprio oggetto MQQueueManager .

messaggio
Contiene il descrittore del messaggio e i dati del messaggio restituiti. Alcuni dei campi nel descrittore del messaggio sono parametri di input. È importante assicurarsi che i parametri di input MessageId e CorrelationId vengano impostati come richiesto.

Un client ricollegabile restituisce il codice di errore MQRC_BACKED_OUT dopo una riconnessione riuscita, per i messaggi ricevuti in MQGM_SYNCPOINT.

Opzioni getMessage
Opzioni che controllano l'azione del get.

L'utilizzo dell'opzione MQC.MQGMO_CONVERT potrebbe causare un'eccezione con codice di errore MQC.MQRC_CONVERTED_STRING_TOO_BIG durante la conversione da codici di caratteri a byte singolo in codici a byte doppio. In tal caso, il messaggio viene copiato nel buffer senza conversione.

Se *getMessageOptions* non è specificato, l'opzione del messaggio utilizzata è MQGMO_NOWAIT.

Se si utilizza l'opzione MQGMO_LOGICAL_ORDER in un client ricollegabile, viene restituito il codice di errore MQRC_RECONNECT_INCOMPATIBLE .

Dimensione MaxMsg
Il messaggio più grande che questo oggetto messaggio deve ricevere. Se il messaggio sulla coda è più grande di questa dimensione, si verifica una delle due seguenti situazioni:

- Se l'indicatore MQGMO_ACCEPT_TRUNCATED_MSG è impostato nell'oggetto MQGetMessageOptions , il messaggio viene riempito con il maggior numero possibile di dati del messaggio. Viene generata un'eccezione con il codice di completamento MQCC_WARNING e il codice motivo MQRC_TRUNCATED_MSG_ACCEPTED .
- Se l'indicatore MQGMO_ACCEPT_TRUNCATED_MSG non è impostato, il messaggio rimane nella coda. Viene generata un'eccezione con il codice di completamento MQCC_WARNING e il codice motivo MQRC_TRUNCATED_MSG_FAILED .

Se *MaxMsgSize* non è specificato, viene richiamato l'intero messaggio.

```
public void Put(MQMessage message);
public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);
```

Genera MQException.

Inserisce un messaggio in una coda.

Le modifiche all'oggetto MQMessage dopo che la chiamata Put è stata effettuata non influiscono sul messaggio effettivo sulla coda IBM MQ o sull'argomento di pubblicazione.

Put aggiorna le proprietà MessageId e CorrelationId dell'oggetto MQMessage e non cancella i dati del messaggio. Ulteriori chiamate Put o Get fanno riferimento alle informazioni aggiornate nell'oggetto MQMessage . Ad esempio, nel seguente frammento di codice, il primo messaggio contiene a e il secondo ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

messaggio
Un MQMessage oggetto contenente i dati del descrittore del messaggio e il messaggio da inviare. Il descrittore del messaggio può essere modificato come conseguenza di questo metodo. I valori nel descrittore del messaggio immediatamente dopo il completamento di questo metodo sono i valori inseriti nella coda o pubblicati nell'argomento.

I seguenti codici di errore vengono restituiti ad un client ricollegabile:

- MQRC_CALL_INTERRUPTED se la connessione viene interrotta durante l'esecuzione di una chiamata Put su un messaggio persistente e la riconnessione ha esito positivo.
- MQRC_NONE se la connessione ha esito positivo durante l'esecuzione di una chiamata Put su un messaggio non persistente (consultare Ripristino applicazione).

Opzioni putMessage
Opzioni che controllano l'azione dell'inserimento.

Se *putMessageOptions* non è specificato, viene utilizzata l'istanza predefinita di MQPutMessageOptons .

Se si utilizza l'opzione MQPMO_LOGICAL_ORDER in un client ricollegabile, viene restituito il codice di errore MQRC_RECONNECT_INCOMPATIBLE .

Nota: Per semplicità e prestazioni, se si desidera inserire un singolo messaggio in una coda, utilizzare l'oggetto MQQueueManager.Put . Si dovrebbe avere un oggetto MQQueue per questo.

public void PutForwardMessage(MQMessage *message*);
public void PutForwardMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);

Eccezioni generate MQException

Inserire un messaggio inoltrato nella coda, dove *message* è il messaggio originale.

messaggio
Un MQMessage oggetto contenente i dati del descrittore del messaggio e il messaggio da inviare. Il descrittore del messaggio può essere modificato come conseguenza di questo metodo. I valori nel descrittore del messaggio immediatamente dopo il completamento di questo metodo sono i valori inseriti nella coda o pubblicati nell'argomento.

I seguenti codici di errore vengono restituiti ad un client ricollegabile:

- MQRC_CALL_INTERRUPTED se la connessione viene interrotta durante l'esecuzione di una chiamata Put su un messaggio persistente e la riconnessione ha esito positivo.
- MQRC_NONE se la connessione ha esito positivo durante l'esecuzione di una chiamata Put su un messaggio non persistente (consultare Ripristino applicazione).

Opzioni putMessage
Opzioni che controllano l'azione dell'inserimento.

Se *putMessageOptions* non è specificato, viene utilizzata l'istanza predefinita di MQPutMessageOptons .

Se si utilizza l'opzione MQPMO_LOGICAL_ORDER in un client ricollegabile, viene restituito il codice di errore MQRC_RECONNECT_INCOMPATIBLE .

public void PutReplyMessage(MQMessage *message*)
public void PutReplyMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*)

Genera MQException.

Inserire un messaggio di risposta nella coda, dove *message* è il messaggio originale.

messaggio
Contiene il descrittore del messaggio e i dati del messaggio restituiti. Alcuni dei campi nel descrittore del messaggio sono parametri di input. È importante assicurarsi che i parametri di input MessageId e CorrelationId vengano impostati come richiesto.

Un client ricollegabile restituisce il codice di errore MQRC_BACKED_OUT dopo una riconnessione riuscita, per i messaggi ricevuti in MQGM_SYNCPOINT.

Opzioni putMessage
Opzioni che controllano l'azione dell'inserimento.

Se *putMessageOptions* non è specificato, viene utilizzata l'istanza predefinita di MQPutMessageOptons .

Se si utilizza l'opzione MQPMO_LOGICAL_ORDER in un client ricollegabile, viene restituito il codice di errore MQRC_RECONNECT_INCOMPATIBLE .

public void PutReportMessage(MQMessage *message*)
public void PutReportMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*)

Genera MQException.

Inserire un messaggio di report nella coda, dove *message* è il messaggio originale.

messaggio
Contiene il descrittore del messaggio e i dati del messaggio restituiti. Alcuni dei campi nel descrittore del messaggio sono parametri di input. È importante assicurarsi che i parametri di input `MessageId` e `CorrelationId` vengano impostati come richiesto.

Un client ricollegabile restituisce il codice di errore `MQRC_BACKED_OUT` dopo una riconnessione riuscita, per i messaggi ricevuti in `MQGM_SYNCPOINT`.

Opzioni putMessage
Opzioni che controllano l'azione dell'inserimento.

Se *`putMessageOptions`* non è specificato, viene utilizzata l'istanza predefinita di `MQPutMessageOptons` .

Se si utilizza l'opzione `MQPMO_LOGICAL_ORDER` in un client ricollegabile, viene restituito il codice di errore `MQRC_RECONNECT_INCOMPATIBLE` .

Costruttori

```
public MQQueue MQQueueManager.AccessQueue(string queueName, int openOptions);
public MQQueue MQQueueManager.AccessQueue(string queueName, int openOptions,
string queueManagerName, string dynamicQueueName, string alternateUserId);
```

Genera `MQException`.

Accede a una coda su questo gestore code.

È possibile richiamare o sfogliare i messaggi, inserire i messaggi, richiedere informazioni sugli attributi della coda o impostare gli attributi della coda. Se la coda denominata è una coda modello, viene creata una coda locale dinamica. Interrogare l'attributo `name` dell'oggetto `MQQueue` risultante per individuare il nome della coda dinamica.

queueName
Nome della coda da aprire.

openOptions
Opzioni che controllano l'apertura della coda.

MQC.MQOO_ALTERNATE_USER_AUTHORITY
Convalidare con l'identificativo utente specificato.

MQC.MQOO_BIND_AS_QDEF
Utilizzare il binding predefinito per la coda.

MQC.MQOO_BIND_NOT_FIXED
Non eseguire il collegamento a una specifica destinazione.

MQC.MQOO_BIND_ON_OPEN
Collega l'handle alla destinazione quando la coda è aperta.

MQC.MQOO_BROWSE
Aprire per sfogliare il messaggio.

MQC.MQOO_FAIL_IF_QUIESCING
Errore se il gestore code è in attesa.

MQC.MQOO_INPUT_AS_Q_DEF
Aprire per richiamare i messaggi utilizzando il valore predefinito definito dalla coda.

MQC.MQOO_INPUT_SHARED
Aprire per ottenere messaggi con accesso condiviso.

MQC.MQOO_INPUT_EXCLUSIVE
Aprire per ottenere messaggi con accesso esclusivo.

MQC.MQOO_INQUIRE
Aperto per richiesta - obbligatorio se si desidera interrogare le proprietà.

MQC.MQOO_OUTPUT
Aprire per inserire i messaggi.

MQC.MQOO_PASS_ALL_CONTEXT
Consenti il passaggio di tutto il contesto.

MQC.MQOO_PASS_IDENTITY_CONTEXT
Consenti il passaggio del contesto di identità.

MQC.MQOO_SAVE_ALL_CONTEXT
Salva contesto quando viene richiamato il messaggio.

MQC.MQOO_SET
Aprire per impostare gli attributi - obbligatorio se si desidera impostare le proprietà.

MQC.MQOO_SET_ALL_CONTEXT
Consente l'impostazione di tutto il contesto.

MQC.MQOO_SET_IDENTITY_CONTEXT
Consente l'impostazione del contesto identità.

queueManagerName
Nome del gestore code su cui è definita la coda. Un nome completamente vuoto o null indica il gestore code a cui è connesso l'oggetto `MQQueueManager` .

Nome dynamicQueue
dynamicQueueName viene ignorato a meno che `queueName` non specifichi il nome di una coda modello. In caso affermativo, *dynamicQueueName* specifica il nome della coda dinamica da creare. Un nome vuoto o null non è valido se `queueName` specifica il nome di una coda modello. Se l'ultimo carattere non vuoto nel nome è un asterisco, `*`, il gestore code sostituisce l'asterisco con una stringa di caratteri. I caratteri garantiscono che il nome generato per la coda sia univoco su questo gestore code.

AlternateUserid
Se `MQC.MQOO_ALTERNATE_USER_AUTHORITY` è specificato nel parametro `openOptions` , *alternateUserId* specifica l'identificativo utente alternativo utilizzato per controllare l'autorizzazione per l'apertura. Se `MQC.MQOO_ALTERNATE_USER_AUTHORITY` non è specificato, *alternateUserId* può essere lasciato vuoto o null.

public MQQueue(MQQueueManager *queueManager*, string *queueName*, int *openOptions*, string *queueManagerName*, string *dynamicQueueName*, string *alternateUserId*);

Genera `MQException`.

Accede a una coda su `queueManager`.

È possibile richiamare o sfogliare i messaggi, inserire i messaggi, richiedere informazioni sugli attributi della coda o impostare gli attributi della coda. Se la coda denominata è una coda modello, viene creata una coda locale dinamica. Interrogare l'attributo `name` dell'oggetto `MQQueue` risultante per individuare il nome della coda dinamica.

queueManager
Gestore code su cui accedere alla coda.

queueName
Nome della coda da aprire.

openOptions
Opzioni che controllano l'apertura della coda.

MQC.MQOO_ALTERNATE_USER_AUTHORITY
Convalidare con l'identificativo utente specificato.

MQC.MQOO_BIND_AS_QDEF
Utilizzare il binding predefinito per la coda.

MQC.MQOO_BIND_NOT_FIXED
Non eseguire il collegamento a una specifica destinazione.

MQC.MQOO_BIND_ON_OPEN
Collega l'handle alla destinazione quando la coda è aperta.

MQC.MQOO_BROWSE
Aprire per sfogliare il messaggio.

MQC.MQOO_FAIL_IF_QUIESCING
Errore se il gestore code è in attesa.

MQC.MQOO_INPUT_AS_Q_DEF
Aprire per richiamare i messaggi utilizzando il valore predefinito definito dalla coda.

MQC.MQOO_INPUT_SHARED
Aprire per ottenere messaggi con accesso condiviso.

MQC.MQOO_INPUT_EXCLUSIVE
Aprire per ottenere messaggi con accesso esclusivo.

MQC.MQOO_INQUIRE
Aperto per richiesta - obbligatorio se si desidera interrogare le proprietà.

MQC.MQOO_OUTPUT
Aprire per inserire i messaggi.

MQC.MQOO_PASS_ALL_CONTEXT
Consenti il passaggio di tutto il contesto.

MQC.MQOO_PASS_IDENTITY_CONTEXT
Consenti il passaggio del contesto di identità.

MQC.MQOO_SAVE_ALL_CONTEXT
Salva contesto quando viene richiamato il messaggio.

MQC.MQOO_SET
Aprire per impostare gli attributi - obbligatorio se si desidera impostare le proprietà.

MQC.MQOO_SET_ALL_CONTEXT
Consente l'impostazione di tutto il contesto.

MQC.MQOO_SET_IDENTITY_CONTEXT
Consente l'impostazione del contesto identità.

queueManagerName
Nome del gestore code su cui è definita la coda. Un nome completamente vuoto o null indica il gestore code a cui è connesso l'oggetto `MQQueueManager` .

Nome dynamicQueue
dynamicQueueName viene ignorato a meno che `queueName` non specifichi il nome di una coda modello. In caso affermativo, *dynamicQueueName* specifica il nome della coda dinamica da creare. Un nome vuoto o null non è valido se `queueName` specifica il nome di una coda modello. Se l'ultimo carattere non vuoto nel nome è un asterisco, `*`, il gestore code sostituisce l'asterisco con una stringa di caratteri. I caratteri garantiscono che il nome generato per la coda sia univoco su questo gestore code.

AlternateUserid
Se `MQC.MQOO_ALTERNATE_USER_AUTHORITY` è specificato nel parametro `openOptions` , *alternateUserId* specifica l'identificativo utente alternativo utilizzato per controllare l'autorizzazione per l'apertura. Se `MQC.MQOO_ALTERNATE_USER_AUTHORITY` non è specificato, *alternateUserId* può essere lasciato vuoto o null.

Classe MQQueueManager.NET

Utilizzare `MQQueueManager` per connettersi a un gestore code e accedere agli oggetti del gestore code. Controlla anche le transazioni. Il costruttore `MQQueueManager` crea una connessione client o server.

Classe

```
System.Object
        │
        └─ IBM.WMQ.MQBase
                │
                └─ IBM.WMQ.MQBaseObject
                        │
                        └─ IBM.WMQ.ManagedObject
                                │
                                └─ IBM.WMQ.MQQueueManager
```

```
public class IBM.WMQ.MQQueueManager extends IBM.WMQ.MQManagedObject;
```


- “Proprietà” a pagina 1810
- “Metodi” a pagina 1813
- “Costruttori” a pagina 1819

Proprietà

Test per `MQException` generato durante il richiamo delle proprietà.

public int AccountingConnOverride {get;}
Indica se le applicazioni possono sovrascrivere l'impostazione dei valori di account della coda e account `MQI` .

public int AccountingInterval {get;}
Il tempo prima che vengano scritti i record di account intermedi (in secondi).

public int ActivityRecording {get;}
Controlla la generazione dei prospetti delle attività.

public int AdoptNewMCACheck {get;}
Specifica quali elementi vengono controllati per stabilire se l'MCA viene adottato quando viene rilevato un nuovo canale in entrata. Per essere adottato, il nome MCA deve corrispondere al nome di un MCA attivo.

public int AdoptNewMCAInterval {get;}
L'intervallo di tempo, in secondi, durante il quale il nuovo canale attende la fine del canale orfano.

public int AdoptNewMCAType {get;}
Indica se un'istanza MCA orfana deve essere adottata (riavviata) quando viene rilevata una nuova richiesta di canale in entrata che corrisponde al valore MCACheck AdoptNew.

public int BridgeEvent {get;}
Se vengono generati eventi bridge IMS .

public int ChannelEvent {get;}
Indica se vengono generati eventi del canale.

public int ChannelInitiatorControl {get;}
Indica se l'iniziatore di canali viene avviato automaticamente all'avvio del gestore code.

public int ChannelInitiatorAdapters {get;}
Il numero di attività secondarie dell'adattatore per elaborare le chiamate IBM MQ .

public int ChannelInitiatorDispatchers {get;}
Il numero di dispatcher da utilizzare per l'iniziatore di canali.

public int ChannelInitiatorTraceAutoStart {get;}
Specifica se la traccia dell'iniziatore di canali viene avviata automaticamente.

public int ChannelInitiatorTraceTableSize {get;}
La dimensione, in megabyte, dello spazio dati di traccia di un iniziatore di canali.

public int ChannelMonitoring {get;}
Indica se viene utilizzato il monitoraggio del canale.

public int ChannelStatistics {get;}
Controlla la raccolta dei dati statistici per i canali.

public int CharacterSet {get;}
Restituisce il CCSID (coded character set identifier) del gestore code. `CharacterSet` viene utilizzato dal gestore code per tutti i campi stringa di caratteri nell'API (application programming interface).

public int ClusterSenderMonitoring {get;}
Controlla la raccolta dei dati di controllo in linea per i canali mittenti del cluster definiti automaticamente.

public int ClusterSenderStatistics {get;}
Controlla la raccolta dei dati statistici per i canali mittenti del cluster definiti automaticamente.

public int ClusterWorkLoadMRU {get;}
Il numero massimo di canali cluster in uscita.

public int ClusterWorkLoadUseQ {get;}
Il valore predefinito della proprietà `MQQueue` , `ClusterWorkLoadUseQ`, se specifica un valore `QMGR`.

public int CommandEvent {get;}
Specifica se vengono generati eventi dei comandi.

public string CommandInputQueueName {get;}
Restituisce il nome della coda di input dei comandi definita nel gestore code. Le applicazioni possono inviare comandi a questa coda, se autorizzate.

public int CommandLevel {get;}
Indica il livello funzione del gestore code. La serie di funzioni che corrisponde a un particolare livello di funzione dipende dalla piattaforma. Su una particolare piattaforma, è possibile fare affidamento su ogni gestore code che supporta le funzioni al livello funzionale più basso comune a tutti i gestori code.

public int CommandLevel {get;}
Indica se il server dei comandi viene avviato automaticamente all'avvio del gestore code.

public string DNSGroup {get;}
Non più utilizzato.

public int DNSWLM {get;}
Non più utilizzato.

public int IPAddressVersion {get;}
Quale protocollo IP (IPv4 o IPv6) utilizzare per una connessione di canale.

public boolean IsConnected {get;}
Restituisce il valore di `isConnected`.

Se true, è stata effettuata una connessione al gestore code e non si sa se è stata interrotta. Qualsiasi chiamata a IsConnected non tenta attivamente di raggiungere il gestore code, quindi è possibile che la connettività fisica possa interrompersi, ma IsConnected può ancora restituire true. Lo stato IsConnected viene aggiornato solo quando l'attività, ad esempio l'inserimento di un messaggio, l'acquisizione di un messaggio, viene eseguita nel gestore code.

Se false, una connessione al gestore code non è stata effettuata, è stata interrotta o è stata disconnessa.

public int KeepAlive {get;}
Specifica se la funzione TCP KEEPALIVE deve essere utilizzata per controllare che l'altra estremità della connessione sia ancora disponibile. Se non è disponibile, il canale viene chiuso.

public int ListenerTimer {get;}
L'intervallo di tempo, in secondi, tra i tentativi di IBM MQ di riavviare il listener dopo un errore APPC o TCP/IP.

public int LoggerEvent {get;}
Indica se vengono generati gli eventi del programma di registrazione.

public string LU62ARMSuffix {get;}
Il suffisso del membro APPCPM di SYS1.PARMLIB. Questo suffisso nomina il LUADD per questo iniziatore di canali. Quando ARM (automatic restart manager) riavvia l'iniziatore di canali, viene emesso il comando z/OS SET APPC=xx.

public string LUGroupName {get; z/os}
Il nome LU generico che deve essere utilizzato dal listener LU 6.2 che gestisce le trasmissioni in entrata per il gruppo di condivisione code.

public string LUName {get;}
Il nome della LU da utilizzare per le trasmissioni LU in uscita 6.2 .

public int MaximumActiveChannels {get;}
Indica il numero massimo di canali che possono essere attivi contemporaneamente.

public int MaximumCurrentChannels {get;}
Il numero massimo di canali che possono essere correnti in qualsiasi momento (inclusi i canali di connessione server con i client connessi).

public int MaximumLU62Channels {get;}
Il numero massimo di canali che possono essere correnti o di client che possono essere connessi, che utilizzano il protocollo di trasmissione LU 6.2 .

public int MaximumMessageLength {get;}
Restituisce la lunghezza massima di un messaggio (in byte) che può essere gestito dal gestore code. Nessuna coda può essere definita con una lunghezza massima del messaggio maggiore di `MaximumMessageLength`.

public int MaximumPriority {get;}
Restituisce la priorità massima del messaggio supportata dal gestore code. Le priorità vanno da zero (più basso) a questo valore. Genera `MQException` se si richiama questo metodo dopo la disconnessione dal gestore code.

public int MaximumTCPChannels {get;}
Il numero massimo di canali che possono essere correnti o di client che possono essere connessi che utilizzano il protocollo di trasmissione TCP/IP.

public int MQIAccounting {get;}
Controlla la raccolta delle informazioni di account per i dati MQI.

public int MQIStatistics {get;}
Controlla la raccolta delle informazioni di controllo per il gestore code.

public int OutboundPortMax {get;}
Il valore massimo nell'intervallo di numeri di porta da utilizzare durante il bind dei canali in uscita.

public int OutboundPortMin {get;}
Il valore minimo nell'intervallo di numeri di porta da utilizzare durante il bind dei canali in uscita.

public int QueueAccounting {get;}
Indica se i dati di account di classe 3 (account a livello di thread e a livello di coda) devono essere utilizzati per tutte le code.

public int QueueMonitoring {get;}
Controlla la raccolta dei dati di controllo online per le code.

public int QueueStatistics {get;}
Controlla la raccolta dei dati statistici per le code.

public int ReceiveTimeout {get;}
Il periodo di tempo durante il quale un canale TCP/IP attende di ricevere i dati, inclusi gli heartbeat, dal partner prima di tornare allo stato inattivo.

public int ReceiveTimeoutMin {get;}
L'intervallo di tempo minimo durante il quale un canale TCP/IP attende di ricevere i dati, inclusi gli heartbeat, dal proprio partner prima di tornare ad uno stato inattivo.

public int ReceiveTimeoutType {get;}
Il qualificatore da applicare al valore in `ReceiveTimeout`.

public int SharedQueueQueueManagerName {get;}
Specifica come consegnare i messaggi a una coda condivisa. Se l'inserimento specifica un gestore code diverso dallo stesso gruppo di condivisione code del gestore code di destinazione, il messaggio viene recapitato in due modi:

MQC.MQSQQM_USE
I messaggi vengono consegnati al gestore code oggetti prima di essere inseriti nella coda condivisa.

MQCMQSQQM_IGNORE
I messaggi vengono inseriti direttamente nella coda condivisa.

public int SSLEvent {get;}
Se vengono generati eventi TLS.

public int SSLFips {get;}
Indica se devono essere utilizzati solo algoritmi certificati FIPS se la crittografia viene eseguita in IBM MQ, piuttosto che in hardware crittografico.

public int SSLKeyResetCount {get;}
Indica il numero di byte non crittografati inviati e ricevuti all'interno di una conversazione TLS prima che la chiave segreta venga rinegoziata.

public int ClusterSenderStatistics {get;}
Specifica l'intervallo, in minuti, tra raccolte consecutive di statistiche.

public int SyncpointAvailability {get;}
Indica se il gestore code supporta le unità di lavoro e i punti di sincronizzazione con i metodi `MQQueue.get` e `MQQueue.put` .

public string TCPName {get;}
Il nome dell'unico sistema TCP/IP o del sistema predefinito da utilizzare, a seconda del valore di TCPStackType.

public int TCPStackType {get;}
Specifica se l'iniziatore di canali utilizza solo lo spazio di indirizzo TCP/IP specificato in `TCPName`. In alternativa, l'iniziatore di canali può collegarsi a qualsiasi indirizzo TCP/IP.

public int TraceRouteRecording {get;}
Controlla la registrazione delle informazioni di traccia del percorso.

Metodi

public MQProcess AccessProcess(string *processName*, int *openOptions*);
public MQProcess AccessProcess(string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Genera `MQException`.

Accedere a un processo IBM MQ su questo gestore code per analizzare gli attributi del processo.

processName
Il nome del processo da aprire.

openOptions
Opzioni che controllano l'apertura del processo. Le opzioni valide che possono essere aggiunte o combinate utilizzando un OR bitwise sono:

- `MQC.MQOO_FAIL_IF_QUIESCING`
- `MQC.MQOO_INQUIRE`
- `MQC.MQOO_SET`
- `MQC.MQOO_ALTERNATE_USER_AUTHORITY`

queueManagerName
Il nome del gestore code su cui è definito il processo. È possibile lasciare un nome gestore code vuoto o null se il gestore code è lo stesso a cui accede il processo.

AlternateUserid

Se `MQC.MQOO_ALTERNATE_USER_AUTHORITY` è specificato nel parametro **openOptions** , *`alternateUserId`* specifica l'ID utente alternativo utilizzato per controllare l'autorizzazione

per l'azione. Se MQOO_ALTERNATE_USER_AUTHORITY non è specificato, *alternateUserId* può essere vuoto o null.

L'autorizzazione utente predefinita viene utilizzata per la connessione al gestore code se MQC.MQOO_ALTERNATE_USER_AUTHORITY non è specificato.

public MQQueue AccessQueue(string *queueName*, int *openOptions*);
public MQQueue AccessQueue(string *queueName*, int *openOptions*, string *queueManagerName*, string *dynamicQueueName*, string *alternateUserId*);

Genera MQException.

Accede a una coda su questo gestore code.

È possibile richiamare o sfogliare i messaggi, inserire i messaggi, richiedere informazioni sugli attributi della coda o impostare gli attributi della coda. Se la coda denominata è una coda modello, viene creata una coda locale dinamica. Interrogare l'attributo name dell'oggetto MQQueue risultante per individuare il nome della coda dinamica.

queueName
Nome della coda da aprire.

openOptions
Opzioni che controllano l'apertura della coda.

MQC.MQOO_ALTERNATE_USER_AUTHORITY
Convalidare con l'identificativo utente specificato.

MQC.MQOO_BIND_AS_QDEF
Utilizzare il binding predefinito per la coda.

MQC.MQOO_BIND_NOT_FIXED
Non eseguire il collegamento a una specifica destinazione.

MQC.MQOO_BIND_ON_OPEN
Collega l'handle alla destinazione quando la coda è aperta.

MQC.MQOO_BROWSE
Aprire per sfogliare il messaggio.

MQC.MQOO_FAIL_IF_QUIESCING
Errore se il gestore code è in attesa.

MQC.MQOO_INPUT_AS_Q_DEF
Aprire per richiamare i messaggi utilizzando il valore predefinito definito dalla coda.

MQC.MQOO_INPUT_SHARED
Aprire per ottenere messaggi con accesso condiviso.

MQC.MQOO_INPUT_EXCLUSIVE
Aprire per ottenere messaggi con accesso esclusivo.

MQC.MQOO_INQUIRE
Aperto per richiesta - obbligatorio se si desidera interrogare le proprietà.

MQC.MQOO_OUTPUT
Aprire per inserire i messaggi.

MQC.MQOO_PASS_ALL_CONTEXT
Consenti il passaggio di tutto il contesto.

MQC.MQOO_PASS_IDENTITY_CONTEXT
Consenti il passaggio del contesto di identità.

MQC.MQOO_SAVE_ALL_CONTEXT
Salva contesto quando viene richiamato il messaggio.

MQC.MQOO_SET
Aprire per impostare gli attributi - obbligatorio se si desidera impostare le proprietà.

MQC.MQOO_SET_ALL_CONTEXT
Consente l'impostazione di tutto il contesto.

MQC.MQOO_SET_IDENTITY_CONTEXT
Consente l'impostazione del contesto identità.

queueManagerName
Nome del gestore code su cui è definita la coda. Un nome completamente vuoto o null indica il gestore code a cui è connesso l'oggetto MQQueueManager .

Nome dynamicQueue
dynamicQueueName viene ignorato a meno che queueName non specifichi il nome di una coda modello. In caso affermativo, *dynamicQueueName* specifica il nome della coda dinamica da creare. Un nome vuoto o null non è valido se queueName specifica il nome di una coda modello. Se l'ultimo carattere non vuoto nel nome è un asterisco, *, il gestore code sostituisce l'asterisco con una stringa di caratteri. I caratteri garantiscono che il nome generato per la coda sia univoco su questo gestore code.

AlternateUserid
Se MQC.MQOO_ALTERNATE_USER_AUTHORITY è specificato nel parametro openOptions , *alternateUserId* specifica l'identificativo utente alternativo utilizzato per controllare l'autorizzazione per l'apertura. Se MQC.MQOO_ALTERNATE_USER_AUTHORITY non è specificato, *alternateUserId* può essere lasciato vuoto o null.

```
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName,
System.Collections.Hashtable properties);
public MQTopic AccessTopic(string topicName, string topicObject, int openAs,
int options);
public MQTopic AccessTopic(string topicName, string topicObject, int openAs,
int options, string alternateUserId);
public MQTopic AccessTopic(string topicName, string topicObject, int options,
string alternateUserId, string subscriptionName);
public MQTopic AccessTopic(string topicName, string topicObject, int options,
string alternateUserId, string subscriptionName, System.Collections.Hashtable
properties);
```

Accedere a un argomento su questo gestore code.

Gli oggetti MQTopic sono strettamente correlati agli oggetti argomento di gestione, a volte denominati oggetti argomento. Nell'input, topicObject punta a un oggetto argomento di gestione. Il costruttore MQTopic ottiene una stringa di argomenti dall'oggetto argomento e la combina con topicName per creare un nome argomento. topicObject o topicName possono essere null. Il nome dell'argomento viene associato alla struttura ad albero dell'argomento e il nome dell'oggetto dell'argomento di gestione corrispondente più vicino viene restituito in topicObject.

Gli argomenti associati all'oggetto MQTopic sono il risultato della combinazione di due stringhe di argomenti. La prima stringa di argomento è definita dall'oggetto argomento di amministrazione identificato da *topicObject*. La seconda stringa di argomento è *topicString*. La stringa di argomenti risultante associata all'oggetto MQTopic può identificare più argomenti includendo caratteri jolly.

A seconda che l'argomento sia aperto per la pubblicazione o la sottoscrizione, è possibile utilizzare i metodi MQTopic.Put per la pubblicazione sugli argomenti o i metodi MQTopic.Get per ricevere le pubblicazioni sugli argomenti. Se si desidera pubblicare e sottoscrivere lo stesso argomento, è necessario accedere all'argomento due volte, una per la pubblicazione e una per la sottoscrizione.

Se si crea un oggetto MQTopic per la sottoscrizione, senza fornire un oggetto MQDestination , viene utilizzata una sottoscrizione gestita. Se si passa una coda come un oggetto MQDestination ,

viene utilizzata una sottoscrizione non gestita. È necessario assicurarsi che le opzioni di sottoscrizione impostate siano congruenti con la sottoscrizione gestita o non gestita.

destinazione
destination è un'istanza MQQueue . Fornendo *destination*, MQTopic viene aperto come sottoscrizione non gestita. Le pubblicazioni sull'argomento vengono consegnate alla coda a cui si accede come *destination*.

topicName
Una stringa argomento che è la seconda parte del nome argomento. *topicName* è concatenato con la stringa argomento definita nell'oggetto argomento di amministrazione *topicObject* . È possibile impostare *topicName* su un valore null, nel qual caso il nome dell'argomento è definito dalla stringa di argomenti in *topicObject*.

topicObject
Nell'input, *topicObject* è il nome dell'oggetto argomento che contiene la stringa argomento che costituisce la prima parte del nome argomento. La stringa di argomenti in *topicObject* è concatena con *topicName*. Le regole per la costruzione di stringhe di argomento sono definite in Combinazione di stringhe di argomento.

Nell'output, *topicObject* contiene il nome dell'oggetto dell'argomento di gestione che è la corrispondenza più vicina nella struttura ad albero dell'argomento all'argomento identificato dalla stringa dell'argomento.

openAs
Accedere all'argomento per la pubblicazione o la sottoscrizione. Il parametro può contenere solo una delle seguenti opzioni:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

opzioni

Combinare le opzioni che controllano l'apertura dell'argomento per la pubblicazione o la sottoscrizione. Utilizzare le costanti MQC.MQSO_* per accedere a un argomento per la sottoscrizione e le costanti MQC.MQOO_* per accedere a un argomento per la pubblicazione.

Se è richiesta più di un'opzione, aggiungere i valori insieme o combinare i valori dell'opzione utilizzando l'operatore OR bit per bit.

AlternateUserid
Specificare l'ID utente alternativo utilizzato per controllare l'autorizzazione richiesta per completare l'operazione. È necessario specificare *alternateUserId*, se MQC.MQOO_ALTERNATE_USER_AUTHORITY o MQC.MQSO_ALTERNATE_USER_AUTHORITY è impostato nel parametro delle opzioni.

subscriptionName
subscriptionName è richiesto se vengono fornite le opzioni MQC.MQSO_DURABLE o MQC.MQSO_ALTER . In entrambi i casi, MQTopic viene aperto implicitamente per la sottoscrizione. Viene generata un'eccezione se MQC.MQSO_DURABLE è impostato e la sottoscrizione esiste oppure se MQC.MQSO_ALTER è impostato e la sottoscrizione non esiste.

Proprietà
Impostare le proprietà della sottoscrizione speciale elencate utilizzando una tabella hash. Le voci specificate nella tabella hash vengono aggiornate con i valori di output. Le voci non vengono aggiunte alla tabella hash per riportare i valori di output.

- MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID
- MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY
- MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA
- MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID
- MQC.MQSUB_PROP_PUBLICATION_PRIORITY
- MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN

- MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA

public MQAsyncStatus GetAsyncStatus();

Eccezioni generate MQException

Restituisce un oggetto MQAsyncStatus , che rappresenta l'attività asincrona per la connessione del gestore code.

public void Backout();

Genera MQException.

Eseguire il backout di tutti i messaggi che sono stati letti o scritti nel punto di sincronizzazione dall'ultimo punto di sincronizzazione.

I messaggi scritti con l'indicatore MQC.MQPMO_SYNCPOINT impostato vengono rimossi dalle code. I messaggi letti con l'indicatore MQC.MQGMO_SYNCPOINT vengono reintegrati nelle code da cui provengono. Se i messaggi sono persistenti, le modifiche vengono registrate.

Per i client ricollegabili, il codice di errore MQRC_NONE viene restituito a un client dopo la riuscita della riconnessione.

public void Begin();

Genera MQException.

Begin è supportato solo in modalità di bind del server. Avvia un'unità di lavoro globale.

public void Commit();

Genera MQException.

Eseguire il commit di tutti i messaggi che sono stati letti o scritti nel punto di sincronizzazione dall'ultimo punto di sincronizzazione.

I messaggi scritti con l'indicatore MQC.MQPMO_SYNCPOINT impostato vengono resi disponibili ad altre applicazioni. I messaggi richiamati con l'indicatore MQC.MQGMO_SYNCPOINT impostato vengono eliminati. Se i messaggi sono persistenti, le modifiche vengono registrate.

I seguenti codici di errore vengono restituiti ad un client ricollegabile:

- MQRC_CALL_INTERRUPTED se la connessione viene persa durante l'effettuazione della chiamata di commit.
- MQRC_BACKED_OUT se la chiamata di commit viene emessa dopo la riconnessione.

Disconnect();

Genera MQException.

Chiudere la connessione al gestore code. Tutti gli oggetti a cui si accede su questo gestore code non sono più accessibili a questa applicazione. Per accedere nuovamente agli oggetti, creare un oggetto MQQueueManager .

Generalmente, viene eseguito il commit di qualsiasi lavoro eseguito come parte di un'unità di lavoro. Tuttavia, se l'unità di lavoro è gestita da .NET, potrebbe essere eseguito il rollback dell'unità di lavoro.

```
public void Put(int type, string destinationName, MQMessage message);
public void Put(int type, string destinationName, MQMessage message
MQPutMessageOptions putMessageOptions);
public void Put(int type, string destinationName, string queueManagerName,
string topicString, MQMessage message);
public void Put(string queueName, MQMessage message);
public void Put(string queueName, MQMessage message, MQPutMessageOptions
putMessageOptions);
public void Put(string queueName, string queueManagerName, MQMessage message);
public void Put(string queueName, string queueManagerName, MQMessage message,
MQPutMessageOptions putMessageOptions);
public void Put(string queueName, string queueManagerName, MQMessage message,
MQPutMessageOptions putMessageOptions, string alternateUserId);
```

Genera `MQException`.

Inserisce un singolo messaggio in una coda o in un argomento senza creare prima un oggetto `MQQueue` o `MQTopic` .

queueName
Il nome della coda in cui inserire il messaggio.

destinationName
Il nome di un oggetto di destinazione. È una coda o un argomento a seconda del valore di *type*.

il tipo
Il tipo di oggetto di destinazione. Non è necessario combinare le opzioni.

MQC.MQOT_Q
Coda

MQC.MQOT_TOPIC
Argomento

queueManagerName

Il nome del gestore code o dell'alias del gestore code su cui è definita la coda. Se si specifica il tipo `MQC.MQOT_TOPIC` , questo parametro viene ignorato.

Se la coda è una coda modello e il nome gestore code risolto non è questo gestore code, viene generato un `MQException` .

topicString

topicString viene combinato con il nome argomento nell'oggetto argomento *destinationName* .

topicString viene ignorato se *destinationName* è una coda.

messaggio
Il messaggio da inviare. Il messaggio è un oggetto di input/output.

I seguenti codici di errore vengono restituiti ad un client ricollegabile:

- `MQRC_CALL_INTERRUPTED` se la connessione viene interrotta durante l'esecuzione di una chiamata Put su un messaggio persistente.
- `MQRC_NONE` se la connessione ha esito positivo durante l'esecuzione di una chiamata Put su un messaggio non persistente (consultare Recupero dell'applicazione).

Opzioni putMessage

Opzioni che controllano le azioni dell'inserimento.

Se si omette *putMessageOptions*, viene creata un'istanza predefinita di *putMessageOptions* . *putMessageOptions* è un oggetto input/output.

Se si utilizza l'opzione `MQPMO_LOGICAL_ORDER` in un client ricollegabile, viene restituito il codice di errore `MQRC_RECONNECT_INCOMPATIBLE` .

AlternateUserid

Specifica un identificativo utente alternativo utilizzato per controllare l'autorizzazione quando si inserisce il messaggio su una coda.

È possibile omettere *alternateUserId* se non si imposta MQC.MQOO_ALTERNATE_USER_AUTHORITY in *putMessageOptions*. Se si imposta MQC.MQOO_ALTERNATE_USER_AUTHORITY, è necessario impostare anche *alternateUserId*. *alternateUserId* non ha effetto a meno che non si imposti anche MQC.MQOO_ALTERNATE_USER_AUTHORITY.

Costruttori

```
public MQQueueManager();
public MQQueueManager(string queueManagerName);
public MQQueueManager(string queueManagerName, Int options);
public MQQueueManager(string queueManagerName, Int options, string channel,
string connName);
public MQQueueManager(string queueManagerName, string channel, string
connName);
public MQQueueManager(string queueManagerName, System.Collections.Hashtable
properties);
```

Genera MQException.

Crea una connessione a un gestore code. Selezionare tra la creazione di una connessione client o una connessione server.

È necessario disporre dell'autorizzazione di interrogazione (inq) sul gestore code durante il tentativo di connessione al gestore code. Senza richiedere l'autorizzazione, il tentativo di connessione non riesce.

Una connessione client viene creata se si verifica una delle seguenti condizioni:

1. *channel* o *connName* sono specificati nel costruttore.
2. *HostName*, *Port*o *Channel* sono specificati in *properties*.
3. Sono specificati *MQEnvironment.HostName*, *MQEnvironment.Port*o *MQEnvironment.Channel* .

I valori delle proprietà di connessione sono predefiniti nell'ordine mostrato. *channel* e *connName* nel costruttore hanno la precedenza sui valori delle proprietà nel costruttore. I valori delle proprietà del costruttore hanno la precedenza sulle proprietà MQEnvironment .

Il nome host, il nome canale e la porta sono definiti nella classe MQEnvironment .

queueManagerName
Nome del gestore code o del gruppo di gestori code a cui connettersi.

Omettere il parametro o lasciarlo null o vuoto per effettuare una selezione del gestore code predefinito. La connessione del gestore code predefinito su un server è al gestore code predefinito sul server. La connessione del gestore code predefinita su una connessione client è al gestore code a cui è connesso il listener.

opzioni
Specificare le opzioni di connessione MQCNO . I valori devono essere applicabili al tipo di connessione che si sta effettuando. Ad esempio, se si specificano le seguenti proprietà di connessione server per una connessione client, viene generato un MQException .

- MQC.MQCNO_FASTPATH_BINDING
- MQC.MQCNO_STANDARD_BINDING

Proprietà

Il parametro delle proprietà accetta una serie di coppie chiave / valore che sovrascrivono le proprietà impostate da `MQEnvironment` ; vedere l'esempio, “Sovrascrivi proprietà MQEnvironment” a pagina 1822. È possibile sovrascrivere le seguenti proprietà:

- `MQC.CONNECT_OPTIONS_PROPERTY`
- `MQC.CONNECTION_NAME_PROPERTY`
- `MQC.ENCRYPTION_POLICY_SUITE_B`
- `MQC.HOST_NAME_PROPERTY`
- `MQC.PORT_PROPERTY`
- `MQC.CHANNEL_PROPERTY`
- `MQC.SSL_CIPHER_SPEC_PROPERTY`
- `MQC.SSL_PEER_NAME_PROPERTY`
- `MQC.SSL_CERT_STORE_PROPERTY`
- `MQC.SSL_CRYPTO_HARDWARE_PROPERTY`
- `MQC.SECURITY_EXIT_PROPERTY`
- `MQC.SECURITY_USERDATA_PROPERTY`
- `MQC.SEND_EXIT_PROPERTY`
- `MQC.SEND_USERDATA_PROPERTY`
- `MQC.RECEIVE_EXIT_PROPERTY`
- `MQC.RECEIVE_USERDATA_PROPERTY`
- `MQC.USER_ID_PROPERTY`
- `MQC.PASSWORD_PROPERTY`
- `MQC.MQAIR_ARRAY`
- `MQC.KEY_RESET_COUNT`
- `MQC.FIPS_REQUIRED`
- `MQC.HDR_CMP_LIST`
- `MQC.MSG_CMP_LIST`
- `MQC.TRANSPORT_PROPERTY`

canale
Nome di un canale di connessione server

connName
Nome connessione nel formato *HostName* (*Port*).

È possibile fornire un elenco di *nomi host* e *porte* come argomento al costruttore `MQQueueManager (String queueManagerName, Hashtable properties)` utilizzando CONNECTION_NAME_PROPERTY.

Ad esempio:

```
ConnectionName = "fred.mq.com(2344),nick.mq.com(3746),tom.mq.com(4288)";

Hashtable Properties-new Hashtable();

properties.Add(MQC.CONNECTION_NAME_PROPERTY,ConnectionName);

MQQueueManager qmgr=new MQQueue Manager("qmgrname",properties);
```

Quando viene effettuato un tentativo di collegamento, l'elenco dei nomi di connessione viene elaborato in ordine. Se il tentativo di connessione al primo nome host e alla porta ha esito negativo, viene tentata la connessione alla seconda coppia di attributi. Il client ripete questo processo fino a quando non viene stabilita una connessione corretta o fino a quando l'elenco non

viene esaurito. Se l'elenco è esaurito, all'applicazione client vengono restituiti un codice di errore e un codice di completamento appropriati.

Quando non viene fornito un numero di porta per il nome della connessione, la porta predefinita (configurata in `mqclient.ini`) viene utilizzato.

Imposta l'elenco connessioni

È possibile impostare l'elenco di connessioni utilizzando i seguenti metodi quando sono impostate le opzioni di riconnessione client automatica:

Impostare l'elenco di connessioni tramite MQSERVER

È possibile impostare l'elenco delle connessioni tramite il prompt dei comandi.

Al prompt dei comandi, impostare il seguente comando:

```
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/Hostname1(Port1),Hostname2(Por2),Hostname3(Port3)
```

Ad esempio:

```
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/fred.mq.com(5266),nick.mq.com(6566),jack.mq.com(8413)
```

Se si imposta la connessione in MQSERVER, non impostarla nell'applicazione.

Se si imposta l'elenco di connessioni nell'applicazione, l'applicazione sovrascrive quanto impostato nella variabile di ambiente MQSERVER.

Imposta l'elenco delle connessioni tramite l'applicazione

È possibile impostare l'elenco di connessioni nell'applicazione specificando il nome host e le proprietà della porta.

```
String connName = "fred.mq.com(2344), nick.mq.com(3746), chris.mq.com(4288)";
MQQueueManager qm = new MQQueueManager("QM1", "TestChannel", connName);
```

Impostare l'elenco delle connessioni mediante `app.config`

`App.config` è un file XML in cui si specificano le coppie chiave - valore.

Nell'elenco delle connessioni specificare

```
<app.Settings>
<add key="Connection1" value="Hostname1(Port1)"/>
<add key="Connection2" value="Hostname2(Port2)"/>
<app.Settings>
```

Ad esempio:

```
<app.Settings>
<add key>="Connection1" value="fred.mq.com(2966)"/>
<add key>="Connection2" value="alex.mq.com(6533)"/>
<app.Settings>
```

È possibile modificare direttamente l'elenco di connessioni nel file `app.config` .

Impostare l'elenco delle connessioni mediante `MQEnvironment`

Per impostare l'elenco Connessioni tramite `MQEnvironment`, utilizzare la proprietà *ConnectionName* .

```
MQEnvironment.ConnectionName = "fred.mq.com(4288),"alex.mq.com(5211);
```

La proprietà *ConnectionName* sovrascrive il nome host e le proprietà della porta impostati in `MQEnvironment`.

Crea una connessione client

Il seguente esempio mostra come creare una connessione client a un gestore code. È possibile creare una connessione client impostando le variabili `MQEnvironment` prima di creare un nuovo oggetto `MQQueueManager` .

```
MQEnvironment.Hostname = "fred.mq.com";  // host to connect to
MQEnvironment.Port     = 1414;           // port to connect to
                                         //If not explicitly set,
                                         // defaults to 1414
                                         // (the default IBM MQ port)
MQEnvironment.Channel  = "channel.name"; // the case sensitive
                                         //  name of the
                                         // SVR CONN channel on
                                         // the queue manager
MQQueueManager qMgr    = new MQQueueManager("MYQM");
```

Figura 11. Connessione client

Sovrascrivi proprietà `MQEnvironment`

L'esempio riportato di seguito mostra come creare un gestore code con ID utente e password definiti in una tabella hash.

```
Hashtable properties = new Hashtable();

properties.Add( MQC.USER_ID_PROPERTY, "ExampleUserId" );
properties.Add( MQC.PASSWORD_PROPERTY, "ExamplePassword" );

try
{
    MQQueueManager qMgr = new MQQueueManager("qmgrname", properties);
}
catch (MQException mqe)
{
    System.Console.WriteLine("Connect failed with " + mqe.Message);
    return((int)mqe.Reason);
}
```

Figura 12. Sovrascrittura delle proprietà `MQEnvironment`

Crea una connessione ricollegabile

Il seguente esempio mostra come riconnettere automaticamente un client a un gestore code.

```
Hashtable properties = new Hashtable(); // The queue manager name and the
                                 // properties how it has to be connected

properties.Add(MQC.CONNECT_OPTIONS_PROPERTY, MQC.MQCNO_RECONNECT); // Options
                                 // through which reconnection happens

properties.Add(MQC.CONNECTION_NAME_PROPERTY,"fred.mq.com(4789),nick.mq.com(4790)"); // The list
                                 // of queue managers through which reconnection happens

MQ QueueManager qmgr = new MQQueueManager("qmgrname", properties);
```

Figura 13. Riconnessione automatica di un client a un gestore code

Classe `MQSubscription.NET`

Utilizzare `MQSubscription` per richiedere che le pubblicazioni conservate vengano inviate al sottoscrittore. `MQSubscription` è una proprietà di un oggetto `MQTopic` aperto per la sottoscrizione.

Classe

```
System.Object
        │
        └─ IBM.WMQ.MQBase
                │
                └─ IBM.WMQ.MQBaseObject
                        │
                        └─ IBM.WMQ.MQManagedObject
                                │
                                └─ IBM.WMQ.MQSubscription
```

```
public class IBM.WMQ.MQSubscription extends IBM.WMQ.MQManagedObject;
```


- “Proprietà” a pagina 1823
- “Metodi” a pagina 1823
- “Costruttori” a pagina 1823

Proprietà

Accedere alle proprietà di sottoscrizione utilizzando la classe `MQManagedObject` ; consultare “Proprietà” a pagina 1781.

Metodi

Accedere ai metodi di sottoscrizione `Inquire`, `Set` e `Get` utilizzando la classe `MQManagedObject` ; consultare “Metodi” a pagina 1782.

`public int RequestPublicationUpdate(int options);`

Genera `MQException`.

Richiedere una pubblicazione aggiornata per l'argomento corrente. Se il gestore code dispone di pubblicazioni conservate per l'argomento, vengono inviate al sottoscrittore.

Prima di richiamare `RequestPublicationUpdate`, aprire un argomento per la sottoscrizione per ottenere un oggetto `MQSubscription` .

Generalmente, aprire la sottoscrizione con l'opzione `MQC.MQSO_PUBLICATIONS_ON_REQUEST` . Se nella stringa di argomenti non sono presenti caratteri jolly, viene inviata una sola pubblicazione come risultato di questa chiamata. Se la stringa di argomenti contiene caratteri jolly, è possibile che vengano inviate molte pubblicazioni. Il metodo restituisce il numero di pubblicazioni conservate inviate alla coda di sottoscrizione. Non vi è alcuna garanzia che questo numero di pubblicazioni venga ricevuto, soprattutto se si tratta di messaggi non persistenti.

opzioni

`MQC.MQSRO_FAIL_IF_QUIESCING`
Il metodo ha esito negativo se il gestore code è in uno stato di inattività. Su z/OS, per un'applicazione CICS o IMS , `MQC.MQSRO_FAIL_IF_QUIESCING` forza anche l'esito negativo del metodo se la connessione è in uno stato di inattività.

`MQC.MQSRO_NONE`
Non è stata specificata alcuna opzione.

Costruttori

Nessun costruttore pubblico .

Un oggetto `MQSubscription` viene restituito nella proprietà `SubscriptionReference` di un oggetto `MQTopic` aperto per la sottoscrizione,

Richiamare il metodo `RequestPublicationUpdate` . `MQSubscription` è una sottoclasse di `MQManagedObject`. Utilizzare il riferimento per accedere alle proprietà e ai metodi di `MQManagedObject`.

Classe MQTopic.NET

Utilizzare MQTopic per pubblicare o sottoscrivere messaggi su un argomento o per interrogare o impostare attributi di un argomento. Creare un oggetto MQTopic per la pubblicazione o la sottoscrizione utilizzando un costruttore o il metodo MQQueueManager.AccessTopic .

Classe

```
System.Object
        |
        └ IBM.WMQ.MQBase
                |
                └ IBM.WMQ.MQBaseObject
                        |
                        └ IBM.WMQ.MQManagedObject
                                |
                                └ IBM.WMQ.MQDestination
                                        |
                                        └ IBM.WMQ.MQTopic
```

```
public class IBM.WMQ.MQTopic extends IBM.WMQ.MQDestination;
```


- “Proprietà” a pagina 1824
- “Metodi” a pagina 1824
- “Costruttori” a pagina 1826

Proprietà

Test per MQException generato durante il richiamo delle proprietà.

public Boolean IsDurable {get;}
Proprietà di sola lettura che restituisce True se la sottoscrizione è durevole o False in caso contrario. Se l'argomento è stato aperto per la pubblicazione, la proprietà viene ignorata e restituisce sempre False.

public Boolean IsManaged {get;};
Proprietà di sola lettura che restituisce True se la sottoscrizione è gestita dal gestore code o False in caso contrario. Se l'argomento è stato aperto per la pubblicazione, la proprietà viene ignorata e restituisce sempre False.

public Boolean IsSubscribed {get;};
Proprietà di sola lettura che restituisce True se l'argomento è stato aperto per la sottoscrizione e False se l'argomento è stato aperto per la pubblicazione.

public MQSubscription SubscriptionReference {get;};
Proprietà di sola lettura che restituisce l'oggetto MQSubscription associato a un oggetto argomento aperto per la sottoscrizione. Il riferimento è disponibile se si desidera modificare le opzioni di chiusura o avviare uno dei metodi degli oggetti.

public MQDestination UnmanagedDestinationReference {get;};
Proprietà di sola lettura che restituisce il MQQueue associato a una sottoscrizione non gestita. È la destinazione specificata quando è stato creato l'oggetto argomento. La proprietà restituisce un valore null per tutti gli oggetti argomento aperti per la pubblicazione o con una sottoscrizione gestita.

Metodi

public void Put(MQMessage *message*);
public void Put(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);
Genera MQException.

Pubblica un messaggio nell'argomento.

Le modifiche all'oggetto MQMessage dopo che la chiamata Put è stata effettuata non influiscono sul messaggio effettivo sulla coda IBM MQ o sull'argomento di pubblicazione.

Put aggiorna le proprietà MessageId e CorrelationId dell'oggetto MQMessage e non cancella i dati del messaggio. Ulteriori chiamate Put o Get fanno riferimento alle informazioni aggiornate nell'oggetto MQMessage . Ad esempio, nel seguente frammento di codice, il primo messaggio contiene a e il secondo ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

messaggio
Un MQMessage oggetto contenente i dati del descrittore del messaggio e il messaggio da inviare. Il descrittore del messaggio può essere modificato come conseguenza di questo metodo. I valori nel descrittore del messaggio immediatamente dopo il completamento di questo metodo sono i valori inseriti nella coda o pubblicati nell'argomento.

I seguenti codici di errore vengono restituiti ad un client ricollegabile:

- MQRC_CALL_INTERRUPTED se la connessione viene interrotta durante l'esecuzione di una chiamata Put su un messaggio persistente e la riconnessione ha esito positivo.
- MQRC_NONE se la connessione ha esito positivo durante l'esecuzione di una chiamata Put su un messaggio non persistente (consultare Ripristino applicazione).

Opzioni putMessage
Opzioni che controllano l'azione dell'inserimento.

Se *putMessageOptions* non è specificato, viene utilizzata l'istanza predefinita di MQPutMessageOptons .

Se si utilizza l'opzione MQPMO_LOGICAL_ORDER in un client ricollegabile, viene restituito il codice di errore MQRC_RECONNECT_INCOMPATIBLE .

Nota: Per semplicità e prestazioni, se si desidera inserire un singolo messaggio in una coda, utilizzare l'oggetto MQQueueManager.Put . Si dovrebbe avere un oggetto MQQueue per questo.

public void Get(MQMessage *message*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*, int *MaxMsgSize*);
Genera MQException.

Richiama un messaggio dall'argomento.

Questo metodo utilizza un'istanza predefinita di MQGetMessageOptions per eseguire il comando get. L'opzione del messaggio utilizzata è MQGMO_NOWAIT.

Se il richiamo ha esito negativo, l'oggetto MQMessage non viene modificato. Se ha esito positivo, il descrittore del messaggio e le parti di dati del messaggio di MQMessage vengono sostituiti con il descrittore del messaggio e i dati del messaggio dal messaggio in arrivo.

Tutte le chiamate a IBM MQ da un particolare MQQueueManager sono sincrone. Pertanto, se si esegue un richiamo con attesa, a tutti gli altri thread che utilizzano lo stesso MQQueueManager viene impedito di effettuare ulteriori chiamate IBM MQ fino a quando non viene eseguita la chiamata Get. Se sono necessari più thread per accedere simultaneamente a IBM MQ , ogni thread deve creare il proprio oggetto MQQueueManager .

messaggio
Contiene il descrittore del messaggio e i dati del messaggio restituiti. Alcuni dei campi nel descrittore del messaggio sono parametri di input. È importante assicurarsi che i parametri di input MessageId e CorrelationId vengano impostati come richiesto.

Un client ricollegabile restituisce il codice di errore MQRC_BACKED_OUT dopo una riconnessione riuscita, per i messaggi ricevuti in MQGM_SYNCPOINT.

Opzioni getMessage
Opzioni che controllano l'azione del get.

L'utilizzo dell'opzione MQC.MQGMO_CONVERT potrebbe causare un'eccezione con codice di errore MQC.MQRC_CONVERTED_STRING_TOO_BIG durante la conversione da codici di caratteri a byte singolo in codici a byte doppio. In tal caso, il messaggio viene copiato nel buffer senza conversione.

Se *getMessageOptions* non è specificato, l'opzione del messaggio utilizzata è MQGMO_NOWAIT.

Se si utilizza l'opzione MQGMO_LOGICAL_ORDER in un client ricollegabile, viene restituito il codice di errore MQRC_RECONNECT_INCOMPATIBLE .

Dimensione MaxMsg
Il messaggio più grande che questo oggetto messaggio deve ricevere. Se il messaggio sulla coda è più grande di questa dimensione, si verifica una delle due seguenti situazioni:

- Se l'indicatore MQGMO_ACCEPT_TRUNCATED_MSG è impostato nell'oggetto MQGetMessageOptions , il messaggio viene riempito con il maggior numero possibile di dati del messaggio. Viene generata un'eccezione con il codice di completamento MQCC_WARNING e il codice motivo MQRC_TRUNCATED_MSG_ACCEPTED .
- Se l'indicatore MQGMO_ACCEPT_TRUNCATED_MSG non è impostato, il messaggio rimane nella coda. Viene generata un'eccezione con il codice di completamento MQCC_WARNING e il codice motivo MQRC_TRUNCATED_MSG_FAILED .

Se *MaxMsgSize* non è specificato, viene richiamato l'intero messaggio.

Costruttori

```
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options);
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId);
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName);
public MQTopic(MQQueueManager queueManager, MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName, System.Collections.Hashtable properties);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int openAs, int options);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int openAs, int options, string alternateUserId);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName);
public MQTopic(MQQueueManager queueManager, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName,
System.Collections.Hashtable properties);
```

Accedere a un argomento su *queueManager*.

Gli oggetti MQTopic sono strettamente correlati agli oggetti argomento di gestione, a volte denominati oggetti argomento. Nell'input, topicObject punta a un oggetto argomento di gestione. Il costruttore MQTopic ottiene una stringa di argomenti dall'oggetto argomento e la combina con topicName per creare un nome argomento. topicObject o topicName possono essere null. Il nome dell'argomento viene associato alla struttura ad albero dell'argomento e il nome dell'oggetto dell'argomento di gestione corrispondente più vicino viene restituito in topicObject.

Gli argomenti associati all'oggetto MQTopic sono il risultato della combinazione di due stringhe di argomenti. La prima stringa di argomento è definita dall'oggetto argomento di amministrazione identificato da *topicObject*. La seconda stringa di argomento è *topicString*. La stringa di argomenti risultante associata all'oggetto MQTopic può identificare più argomenti includendo caratteri jolly.

A seconda che l'argomento sia aperto per la pubblicazione o la sottoscrizione, è possibile utilizzare i metodi MQTopic.Put per la pubblicazione sugli argomenti o i metodi MQTopic.Get per ricevere le pubblicazioni sugli argomenti. Se si desidera pubblicare e sottoscrivere lo stesso argomento, è necessario accedere all'argomento due volte, una per la pubblicazione e una per la sottoscrizione.

Se si crea un oggetto MQTopic per la sottoscrizione, senza fornire un oggetto MQDestination , viene utilizzata una sottoscrizione gestita. Se si passa una coda come un oggetto MQDestination , viene utilizzata una sottoscrizione non gestita. È necessario assicurarsi che le opzioni di sottoscrizione impostate siano congruenti con la sottoscrizione gestita o non gestita.

queueManager
Gestore code su cui accedere a un argomento.

destinazione
destination è un'istanza MQQueue . Fornendo *destination*, MQTopic viene aperto come sottoscrizione non gestita. Le pubblicazioni sull'argomento vengono consegnate alla coda a cui si accede come *destination*.

topicName
Una stringa argomento che è la seconda parte del nome argomento. *topicName* è concatenato con la stringa argomento definita nell'oggetto argomento di amministrazione *topicObject* . È possibile impostare *topicName* su un valore null, nel qual caso il nome dell'argomento è definito dalla stringa di argomenti in *topicObject*.

topicObject
Nell'input, *topicObject* è il nome dell'oggetto argomento che contiene la stringa argomento che costituisce la prima parte del nome argomento. La stringa di argomenti in *topicObject* è concatena con *topicName*. Le regole per la costruzione di stringhe di argomento sono definite in Combinazione di stringhe di argomento.

Nell'output, *topicObject* contiene il nome dell'oggetto dell'argomento di gestione che è la corrispondenza più vicina nella struttura ad albero dell'argomento all'argomento identificato dalla stringa dell'argomento.

openAs
Accedere all'argomento per la pubblicazione o la sottoscrizione. Il parametro può contenere solo una delle seguenti opzioni:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

opzioni

Combinare le opzioni che controllano l'apertura dell'argomento per la pubblicazione o la sottoscrizione. Utilizzare le costanti MQC.MQSO_* per accedere a un argomento per la sottoscrizione e le costanti MQC.MQOO_* per accedere a un argomento per la pubblicazione.

Se è richiesta più di un'opzione, aggiungere i valori insieme o combinare i valori dell'opzione utilizzando l'operatore OR bit per bit.

AlternateUserid
Specificare l'ID utente alternativo utilizzato per controllare l'autorizzazione richiesta per completare l'operazione. È necessario specificare *alternateUserId*, se MQC.MQOO_ALTERNATE_USER_AUTHORITY o MQC.MQSO_ALTERNATE_USER_AUTHORITY è impostato nel parametro delle opzioni.

subscriptionName
subscriptionName è richiesto se vengono fornite le opzioni MQC.MQSO_DURABLE o MQC.MQSO_ALTER . In entrambi i casi, MQTopic viene aperto implicitamente per la sottoscrizione.

Viene generata un'eccezione se MQC.MQSO_DURABLE è impostato e la sottoscrizione esiste oppure se MQC.MQSO_ALTER è impostato e la sottoscrizione non esiste.

Proprietà

Impostare le proprietà della sottoscrizione speciale elencate utilizzando una tabella hash. Le voci specificate nella tabella hash vengono aggiornate con i valori di output. Le voci non vengono aggiunte alla tabella hash per riportare i valori di output.

- MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID
- MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY
- MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA
- MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID
- MQC.MQSUB_PROP_PUBLICATION_PRIORITY
- MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN
- MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA

```
public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options);
public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId);
public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName);
public MQTopic MQQueueManager.AccessTopic(MQDestination destination, string
topicName, string topicObject, int options, string alternateUserId, string
subscriptionName, System.Collections.Hashtable properties);
public MQTopic MQQueueManager.AccessTopic(string topicName, string topicObject,
int openAs, int options);
public MQTopic MQQueueManager.AccessTopic(string topicName, string topicObject,
int openAs, int options, string alternateUserId);
public MQTopic MQQueueManager.AccessTopic(string topicName, string topicObject,
int options, string alternateUserId, string subscriptionName);
public MQTopic MQQueueManager.AccessTopic(string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName,
System.Collections.Hashtable properties);
```

Accedere a un argomento su questo gestore code.

Gli oggetti MQTopic sono strettamente correlati agli oggetti argomento di gestione, a volte denominati oggetti argomento. Nell'input, topicObject punta a un oggetto argomento di gestione. Il costruttore MQTopic ottiene una stringa di argomenti dall'oggetto argomento e la combina con topicName per creare un nome argomento. topicObject o topicName possono essere null. Il nome dell'argomento viene associato alla struttura ad albero dell'argomento e il nome dell'oggetto dell'argomento di gestione corrispondente più vicino viene restituito in topicObject.

Gli argomenti associati all'oggetto MQTopic sono il risultato della combinazione di due stringhe di argomenti. La prima stringa di argomento è definita dall'oggetto argomento di amministrazione identificato da *topicObject*. La seconda stringa di argomento è *topicString*. La stringa di argomenti risultante associata all'oggetto MQTopic può identificare più argomenti includendo caratteri jolly.

A seconda che l'argomento sia aperto per la pubblicazione o la sottoscrizione, è possibile utilizzare i metodi MQTopic.Put per la pubblicazione sugli argomenti o i metodi MQTopic.Get per ricevere le pubblicazioni sugli argomenti. Se si desidera pubblicare e sottoscrivere lo stesso argomento, è necessario accedere all'argomento due volte, una per la pubblicazione e una per la sottoscrizione.

Se si crea un oggetto MQTopic per la sottoscrizione, senza fornire un oggetto MQDestination , viene utilizzata una sottoscrizione gestita. Se si passa una coda come un oggetto MQDestination ,

viene utilizzata una sottoscrizione non gestita. È necessario assicurarsi che le opzioni di sottoscrizione impostate siano congruenti con la sottoscrizione gestita o non gestita.

destinazione
destination è un'istanza MQQueue . Fornendo *destination*, MQTopic viene aperto come sottoscrizione non gestita. Le pubblicazioni sull'argomento vengono consegnate alla coda a cui si accede come *destination*.

topicName
Una stringa argomento che è la seconda parte del nome argomento. *topicName* è concatenato con la stringa argomento definita nell'oggetto argomento di amministrazione *topicObject* . È possibile impostare *topicName* su un valore null, nel qual caso il nome dell'argomento è definito dalla stringa di argomenti in *topicObject*.

topicObject
Nell'input, *topicObject* è il nome dell'oggetto argomento che contiene la stringa argomento che costituisce la prima parte del nome argomento. La stringa di argomenti in *topicObject* è concatena con *topicName*. Le regole per la costruzione di stringhe di argomento sono definite in Combinazione di stringhe di argomento.

Nell'output, *topicObject* contiene il nome dell'oggetto dell'argomento di gestione che è la corrispondenza più vicina nella struttura ad albero dell'argomento all'argomento identificato dalla stringa dell'argomento.

openAs
Accedere all'argomento per la pubblicazione o la sottoscrizione. Il parametro può contenere solo una delle seguenti opzioni:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

opzioni

Combinare le opzioni che controllano l'apertura dell'argomento per la pubblicazione o la sottoscrizione. Utilizzare le costanti MQC.MQSO_* per accedere a un argomento per la sottoscrizione e le costanti MQC.MQOO_* per accedere a un argomento per la pubblicazione.

Se è richiesta più di un'opzione, aggiungere i valori insieme o combinare i valori dell'opzione utilizzando l'operatore OR bit per bit.

AlternateUserid
Specificare l'ID utente alternativo utilizzato per controllare l'autorizzazione richiesta per completare l'operazione. È necessario specificare *alternateUserId*, se MQC.MQOO_ALTERNATE_USER_AUTHORITY o MQC.MQSO_ALTERNATE_USER_AUTHORITY è impostato nel parametro delle opzioni.

subscriptionName
subscriptionName è richiesto se vengono fornite le opzioni MQC.MQSO_DURABLE o MQC.MQSO_ALTER . In entrambi i casi, MQTopic viene aperto implicitamente per la sottoscrizione. Viene generata un'eccezione se MQC.MQSO_DURABLE è impostato e la sottoscrizione esiste oppure se MQC.MQSO_ALTER è impostato e la sottoscrizione non esiste.

Proprietà
Impostare le proprietà della sottoscrizione speciale elencate utilizzando una tabella hash. Le voci specificate nella tabella hash vengono aggiornate con i valori di output. Le voci non vengono aggiunte alla tabella hash per riportare i valori di output.

- MQC.MQSUB_PROP_ALTERNATE_SECURITY_ID
- MQC.MQSUB_PROP_SUBSCRIPTION_EXPIRY
- MQC.MQSUB_PROP_SUBSCRIPTION_USER_DATA
- MQC.MQSUB_PROP_SUBSCRIPTION_CORRELATION_ID
- MQC.MQSUB_PROP_PUBLICATION_PRIORITY
- MQC.MQSUB_PROP_PUBLICATION_ACCOUNTING_TOKEN

- MQC.MQSUB_PROP_PUBLICATION_APPLICATIONID_DATA

Interfaccia IMQObjectTrigger.NET

Implementare IMQObjectTrigger per elaborare i messaggi trasmessi dal monitor **runmqdnm**.NET .

Interfaccia

```
public interface IBM.WMQMonitor.IMQObjectTrigger();
```

A seconda che il controllo del punto di sincronizzazione sia specificato nel comando **runmqdnm** , il messaggio viene rimosso dalla coda prima o dopo la restituzione del metodo Execute .

Metodi

void Execute (MQQueueManager *queueManager*, MQQueue *queue*, MQMessage *message*, string *param*);

queueManager
Gestore code che ospita la coda monitorata.

coda
Coda monitorata.

messaggio
Messaggio letto dalla coda.

param
Dati passati da UserParameter.

Interfaccia MQC.NET

Fare riferimento ad una costante MQI anteponendo il nome della costante con MQC.. MQC definisce tutte le costanti utilizzate da MQI.

Interfaccia

```
System.Object
        │
        └─ IBM.WMQ.MQC
```

```
public interface IBM.WMQ.MQC extends System.Object;
```

Esempio

```
MQQueue queue;
queue.closeOptions = MQC.MQCO_DELETE;
```

Identificativi della serie di caratteri per applicazioni .NET .

Descrizioni delle serie di caratteri che è possibile selezionare per codificare i messaggi .NET IBM MQ

Serie di caratteri	Descrizione
280	ibm037
437	ibm437 / PC originale
500	ibm500

Serie di caratteri	Descrizione
819	iso-8859-1 / latin1 / ibm819
1200	Unicode
1208	UTF-8
273	ibm273
277	ibm277
278	ibm278
280	ibm280
284	ibm284
285	ibm285
297	ibm297
420	ibm420
424	ibm424
737	ibm737 / PC Greco
775	ibm775 / PC Baltico
813	iso-8859-7 / Greco / ibm813
838	ibm838
850	ibm850 / PC Latino 1
852	ibm852 / PC Latin 2
855	ibm855 / PC Cirillico
856	ibm856
857	ibm857 / PC Turco
860	ibm860 / PC Portoghese
861	ibm861 / PC Islandese
862	ibm862 / PC Ebraico
863	ibm863 / PC Francese canadese
864	ibm864 / PC Arabo
865	ibm865 / PC Nordico
866	ibm866 / PC Russo
868	ibm868
869	ibm869 / PC Greco Moderno
870	ibm870
871	ibm871
874	ibm874
875	ibm875
912	iso-8859-2 / latin2 / ibm912

Serie di caratteri	Descrizione
913	iso-8859-3 / latin3 / ibm913
914	iso-8859-4 / latin4 / ibm914
915	iso-8859-5 / cirillico / ibm915
916	iso-8859-8 / ebraico / ibm916
918	ibm918
920	iso-8859-9 / latin5 / ibm920
921	ibm921
922	ibm922
930	ibm930
932	PC Giapponese
933	ibm933
935	ibm935
937	ibm937
939	ibm939
942	ibm942
943	ibm943
948	ibm948
949	ibm949
950	ibm950 / Big 5 Cinese tradizionale
954	eucjis
964	ibm964 / CNS 11643 Cinese tradizionale
970	ibm970
1006	ibm1006
1025	ibm1025
1026	ibm1026
1089	iso-8859-6 / arabico / ibm1089
1097	ibm1097
1098	ibm1098
1112	ibm1112
1122	ibm1122
1123	ibm1123
1124	ibm1124
1250	Windows Latino 2
1251	Windows Cirillico
1252	Windows Latino 1

Serie di caratteri	Descrizione
1253	Windows Greco
1254	Windows Turco
1255	Windows Ebraico
1256	Windows Arabo
1257	Windows Baltico
1258	Windows Vietnamita
1381	ibm1381
1383	ibm1383
2022	JIS
5601	ksc-5601 Coreano
33722	ibm33722

Classi C++ IBM MQ

Le classi IBM MQ C+ + incapsulano l'interfaccia MQI (Message Queue Interface) IBM MQ . Esiste un singolo file di intestazione C + +, **imqi.hpp**, che copre tutte queste classi.

Per ogni classe, vengono visualizzate le seguenti informazioni:

Diagramma gerarchia di classi
Un diagramma di classe che mostra la classe nella sua relazione di eredità con le sue classi principali immediate, se presenti.

Altre classi pertinenti
Collegamenti di documenti ad altre classi rilevanti, come le classi principali e le classi di oggetti utilizzate nelle firme del metodo.

Attributi oggetto
Attributi della classe. Questi sono in aggiunta agli attributi definiti per le classi principali. Molti attributi riflettono i membri della struttura dati IBM MQ (consultare “Riferimento incrociato C++ e MQI” a pagina 1834). Per descrizioni dettagliate, consultare “Attributi degli oggetti” a pagina 818.

Costruttori
Firme dei metodi speciali utilizzati per creare un oggetto della classe.

Metodi oggetto (pubblico)
Firme di metodi che richiedono un'istanza della classe per il loro funzionamento e che non hanno limitazioni di utilizzo.

Quando si applica, vengono visualizzate anche le seguenti informazioni:

Metodi di classe (pubblico)
Firme di metodi che non richiedono un'istanza della classe per il loro funzionamento e che non hanno limitazioni di utilizzo.

Metodi sovraccaricati (classe parent)
Le firme dei metodi virtuali che sono definiti nelle classi parent, ma che presentano un comportamento polimorfico diverso per questa classe.

Metodi oggetto (protetti)
Firme di metodi che richiedono un'istanza della classe per il loro funzionamento e sono riservati per l'utilizzo da parte delle implementazioni di classi derivate. Questa sezione è di interesse solo per gli scrittori di classe, in contrapposizione agli utenti di classe.

Dati oggetto (protetti)
Dettagli di implementazione per i dati di istanza oggetto disponibili per le implementazioni delle classi derivate. Questa sezione è di interesse solo per gli scrittori di classe, in contrapposizione agli utenti di classe.

Codici di origine
Valori MQRC_ * (vedere Codici di completamento API e motivo) che possono essere previsti da questi metodi che non riescono. Per un elenco completo dei codici di errore che possono verificarsi per un oggetto di una classe, consultare la documentazione della classe parent. L'elenco documentato di codici di errore per una classe non include i codici di errore per le classi principali.

Nota:

1. Gli oggetti di queste classi non sono thread - safe. Ciò garantisce prestazioni ottimali, ma fare attenzione a non accedere ad alcun oggetto da più di un thread.
2. Si consiglia, per un programma a più sottoprocessi, di utilizzare un oggetto ImqQueueManager separato per ciascun thread. Ogni oggetto gestore deve avere la propria raccolta indipendente di altri oggetti, garantendo che gli oggetti in thread diversi siano isolati l'uno dall'altro.

Le classi sono:

Riferimento incrociato C++ e MQI

Questa raccolta di argomenti contiene informazioni relative a C++ per MQI.

Leggere queste informazioni insieme a "Tipi di dati utilizzati in MQI" a pagina 236.

Questa tabella mette in relazione le strutture dati MQI con le classi C + + e i file di inclusione. I seguenti argomenti mostrano informazioni sui riferimenti incrociati per ciascuna classe C + +. Questi riferimenti incrociati si riferiscono all'utilizzo delle interfacce procedurali IBM MQ sottostanti. Le classi ImqBinary, ImqDistributionList e ImqString non hanno attributi che rientrano in questa categoria e sono esclusi.

Tabella 845. Riferimento incrociato file di inclusione, classe e struttura dati

Struttura dei dati	**Classe**	**Includi file**
MQAIR	Record ImqAuthentication	imqair.hpp
	ImqBinary	imqbin.hpp
	ImqCache	imqcac.hpp
MQCD	ImqChannel	imqchl.hpp
MQCIH	ImqCICSBridgeHeader	imqcih.hpp
MQDLH	ImqDeadLetterHeader	imqdlh.hpp
MQOR	Elenco ImqDistribution	imqdst.hpp
	ImqError	imqerr.hpp
MQGMO	ImqGetMessageOptions	imqgmo.hpp
	ImqHeader	imqhdr.hpp
MQIIH	ImqIMSBridgeHeader	imqiih.hpp
	ImqItem	imqitm.hpp
MQMD	ImqMessage	imqmsg.hpp
	Programma di traccia ImqMessage	imqmtr.hpp
	ImqNamelist	imqnml.hpp
MQOD, MQRR	ImqObject	imqobj.hpp
MQPMO, MQPMR, MQRR	ImqPutMessageOptions	imqpmo.hpp
	ImqProcess	imqpro.hpp
	ImqQueue	imqque.hpp
MQBO, MQCNO, MQCSP	Gestore ImqQueue	imqmgr.hpp
MQRMH	Intestazione ImqReference	imqrfh.hpp
	ImqString	imqstr.hpp
MQM	ImqTrigger	imqtrg.hpp
MMQTMC		
MQTMC2	ImqTrigger	imqtrg.hpp
QQMQX		
MQWIH	Intestazione ImqWork	imqwih.hpp

Riferimento incrociato record ImqAuthentication

Riferimento incrociato di attributi, strutture di dati, campi e chiamate per la classe C++ del record ImqAuthentication.

Tabella 846. Attributi, strutture dati, campi e chiamate			
Attributo	**Struttura dei dati**	**Campo**	**Chiamata**
nome connessione	MQAIR	AuthInfoConnName	MQCONNX
password	MQAIR	LDAPPassword	MQCONNX
il tipo	MQAIR	AuthInfoType	MQCONNX
nome utente	MQAIR	LDAPUserNamePtr	MQCONNX
	MQAIR	Offset LDAPUserName	MQCONNX
	MQAIR	LDAPUserNameLunghezza	MQCONNX

Riferimento incrociato ImqCache

Riferimento incrociato di attributi e chiamate per la classe C++ ImqCache .

Tabella 847. Attributi e chiamate	
Attributo	**Chiamata**
buffer automatico	MQGET
lunghezza del buffer	MQGET
puntatore buffer	MQGET, MQPUT
Lunghezza dati	MQGET
Offset dati	MQGET
puntatore dati	MQGET
lunghezza messaggio	MQGET, MQPUT

Riferimento incrociato ImqChannel

Riferimenti incrociati di attributi, strutture dati, campi e chiamate per la classe C++ ImqChannel .

Tabella 848. Attributi, strutture dati, campi e chiamate			
Attributo	**Struttura dei dati**	**Campo**	**Chiamata**
battito cardiaco batch	MQCD	BatchHeartbeat	MQCONNX
nome canale	MQCD	ChannelName	MQCONNX
nome connessione	MQCD	ConnectionName	MQCONNX
	MQCD	Nome ShortConnection	MQCONNX
Compressione intestazione	MQCD	Elenco HdrComp	MQCONNX
intervallo heartbeat	MQCD	HeartbeatInterval	MQCONNX
Intervallo keep alive	MQCD	KeepAliveInterval	MQCONNX
indirizzo locale	MQCD	LocalAddress	MQCONNX
Lunghezza massima dei messaggi	MQCD	MaxMsgLength	MQCONNX
Compressione messaggi	MQCD	Elenco MsgComp	MQCONNX
nome modaltà	MQCD	ModeName	MQCONNX

Tabella 848. Attributi, strutture dati, campi e chiamate (Continua)			
Attributo	**Struttura dei dati**	**Campo**	**Chiamata**
password	MQCD	Password	MQCONNX
conteggio uscite ricezione	MQCD		MQCONNX
nomi delle uscite di ricezione	MQCD	ReceiveExit	MQCONNX
	MQCD	Definite ReceiveExits	MQCONNX
	MQCD	Ptr ReceiveExit	MQCONNX
ricevere i dati utente	MQCD	ReceiveUserData	MQCONNX
	MQCD	ReceiveUserDataPtr	MQCONNX
Nome uscita di sicurezza	MQCD	SecurityExit	MQCONNX
dati utente di protezione	MQCD	SecurityUserDati	MQCONNX
conteggio uscite di invio	MQCD		MQCONNX
nomi uscita di invio	MQCD	SendExit	MQCONNX
	MQCD	SendExitsDefinito	MQCONNX
	MQCD	Ptr SendExit	MQCONNX
invia dati utente	MQCD	SendUserData	MQCONNX
	MQCD	SendUserDataPtr	MQCONNX
SSL CipherSpec	MQCD	Specifica sslCipher	MQCONNX
Tipo di autenticazione client SSL	MQCD	Autenticazione sslClient	MQCONNX
Nome peer SSL	MQCD	SSLPeerName	MQCONNX
Nome programma transazioni	MQCD	TpName	MQCONNX
tipo di trasporto	MQCD	TransportType	MQCONNX
identificativo utente	MQCD	UserIdentifier	MQCONNX

Riferimento incrociato ImqCICSBridgeHeader

Riferimento incrociato di attributi, strutture dati e campi per la classe ImqCICSBridgeHeader C++.

Tabella 849. Associazione di attributi, strutture dati e campi		
Attributo	**Struttura dei dati**	**Campo**
codice di interruzione bridge	MQCIH	AbendCode
Descrittore ADS	MQCIH	AdsDescriptor
AID (attention identifier)	MQCIH	AttentionId
programma di autenticazione	MQCIH	Programma di autenticazione
codice di completamento bridge	MQCIH	Codice BridgeCompletion
offset errore bridge	MQCIH	ErrorOffset
codice motivo bridge	MQCIH	BridgeReason
codice di annullamento bridge	MQCIH	CancelCode

Tabella 849. Associazione di attributi, strutture dati e campi (Continua)		
Attributo	**Struttura dei dati**	**Campo**
attività di conversazione	MQCIH	ConversationalTask
posizione cursore	MQCIH	CursorPosition
token funzione	MQCIH	Funzione
tempo di conservazione facility	MQCIH	FacilityKeepTime
struttura come	MQCIH	FacilityLike
funzione	MQCIH	Funzione
ottieni intervallo di attesa	MQCIH	GetWaitInterval
tipo di link	MQCIH	LinkType
identificativo transazione successiva	MQCIH	NextTransactionId
lunghezza dati di emissione	MQCIH	OutputDataLength
formato reply - to	MQCIH	ReplyToFormat
codice di ritorno bridge	MQCIH	ReturnCode
codice di avvio	MQCIH	StartCode
stato fine attività	MQCIH	TaskEndStatus
identificativo transazione	MQCIH	TransactionId
controllo uow	MQCIH	UowControl
versione	MQCIH	Versione

Riferimento incrociato ImqDeadLetterHeader

Riferimento incrociato di attributi, strutture di dati e campi per la classe C++ ImqDeadLetterHeader .

Tabella 850. Associazione di attributi, strutture dati e campi		
Attributo	**Struttura dei dati**	**Campo**
codice motivo lettera non recapitabile	MQDLH	Motivo
Nome gestore code destinazione	MQDLH	DestQMgrName
Nome coda di destinazione	MQDLH	DestQName
Nome applicazione Put	MQDLH	PutApplName
Tipo di applicazione Put	MQDLH	PutApplType
data di collocazione	MQDLH	PutDate
Ora Put	MQDLH	PutTime

Riferimento incrociato ImqError

Riferimento incrociato di attributi e chiamate per la classe C++ ImqError .

Tabella 851. Attributi e chiamate	
Attributo	**Chiamata**
codice di completamento	MQBACK, MQBEGIN, MQCLOSE, MQ, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET
codice motivo	MQBACK, MQBEGIN, MQCLOSE, MQ, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET

Riferimento incrociato ImqGetMessageOptions

Riferimento incrociato di attributi, strutture di dati e campi per la classe C++ ImqGetMessageOptions .

Tabella 852. Associazione di attributi, strutture dati e campi		
Attributo	**Struttura dei dati**	**Campo**
stato del gruppo	MQGMO	GroupStatus
opzioni di corrispondenza	MQGMO	MatchOptions
token del messaggio	MQGMO	MessageToken
opzioni	MQGMO	Opzioni
Nome coda risolto	MQGMO	ResolvedQName
lunghezza restituita	MQGMO	ReturnedLength
segmentazione	MQGMO	Segmentazione
stato segmento	MQGMO	SegmentStatus
	MQGMO	Signal1
	MQGMO	Signal2
partecipazione punto di sincronizzazione	MQGMO	Opzioni
intervallo di attesa	MQGMO	WaitInterval

Riferimento incrociato ImqHeader

Riferimento incrociato di attributi, strutture dati e campi per la classe C++ ImqHeader .

Tabella 853. Associazione di attributi, strutture dati e campi		
Attributo	**Struttura dei dati**	**Campo**
character set	MQDLH, MQIIH	CodedCharSetId
codifica	MQDLH, MQIIH	Codifica
formato	MQDLH, MQIIH	Formato
indicatori intestazione	MQIIH, MQRMH	Indicatori

Riferimento incrociato ImqIMSBridgeHeader

Riferimento incrociato di attributi, strutture dati e campi per la classe C++ del record ImqAuthentication.

Tabella 854. Associazione di attributi, strutture dati e campi		
Attributo	**Struttura dei dati**	**Campo**
programma di autenticazione	MQIIH	Programma di autenticazione

Tabella 854. Associazione di attributi, strutture dati e campi (Continua)		
Attributo	**Struttura dei dati**	**Campo**
Modalità di commit	MQIIH	CommitMode
Sovrascrittura terminale logico	MQIIH	LTermOverride
Nome associazione di servizi del formato messaggio	MQIIH	MFSMapName
formato reply - to	MQIIH	ReplyToFormat
ambito della sicurezza	MQIIH	SecurityScope
id istanza transazione	MQIIH	TranInstanceId
STATO TRANSAZIONE	MQIIH	TranState

Riferimento incrociato ImqItem

Riferimento incrociato di attributi e chiamate per la classe C++ ImqItem .

Tabella 855. Attributi e chiamate	
Attributo	**Chiamata**
id struttura	MQGET

Riferimento incrociato ImqMessage

Riferimento incrociato di attributi, strutture dati, campi e chiamate per la classe C++ ImqMessage .

Tabella 856. Attributi, strutture dati, campi e chiamate			
Attributo	**Struttura dei dati**	**Campo**	**Chiamata**
Dati ID applicazione	MQMD	ApplIdentityData	
Dati di origine dell'applicazione	MQMD	ApplOriginData	
conteggio backout	MQMD	BackoutCount	
character set	MQMD	CodedCharSetId	
codifica	MQMD	Codifica	
scadenza	MQMD	Scadenza	
formato	MQMD	Formato	
Indicatori di messaggio	MQMD	MsgFlags	
tipo di messaggio	MQMD	MsgType	
scostamento	MQMD	Offset	
Lunghezza originale	MQMD	OriginalLength	
persistenza	MQMD	Persistenza	
priorità	MQMD	Priorità	
Nome applicazione Put	MQMD	PutApplName	
Tipo di applicazione Put	MQMD	PutApplType	
data di collocazione	MQMD	PutDate	

Tabella 856. Attributi, strutture dati, campi e chiamate (Continua)			
Attributo	**Struttura dei dati**	**Campo**	**Chiamata**
Ora Put	MQMD	PutTime	
nome gestore code reply - to	MQMD	ReplyToQMgr	
Nome delle repliche alla coda	MQMD	ReplyToQ	
report	MQMD	Prospetto	
numero di sequenza	MQMD	MsgSeqNumber	
lunghezza totale messaggio		DataLength	MQGET
identificativo utente	MQMD	UserIdentifier	

Riferimento incrociato ImqMessageTracker

Riferimento incrociato di attributi, strutture di dati e campi per la classe C++ di ImqMessageTracker.

Tabella 857. Associazione di attributi, strutture dati e campi		
Attributo	**Struttura dei dati**	**Campo**
token di accounting	MQMD	AccountingToken
ID correlazione	MQMD	CorrelId
il feedback	MQMD	Feedback
ID gruppo	MQMD	GroupId
ID messaggio	MQMD	MsgId

Riferimento incrociato ImqNamelist

Riferimento incrociato di attributi, richieste e chiamate per la classe C++ ImqNamelist .

Tabella 858. Attributi, richieste e chiamate		
Attributo	**Interrogazione**	**Chiamata**
Numero nomi	CONTEGGIO_NAME_MQI	MQINQ
nome namelist	NOME MQCA_NAMELIST_NAME	MQINQ

Riferimento incrociato ImqObject

Riferimenti incrociati di attributi, strutture dati, campi, richieste e chiamate per la classe C++ ImqObject .

Tabella 859. Attributi, strutture dati, campi, richieste e chiamate				
Attributo	**Struttura dei dati**	**Campo**	**Interrogazione**	**Chiamata**
Data della modifica			MQCA_ALTERATION_DATE	MQINQ
Ora della modifica			MQCA_ALTERATION_TIME	MQINQ
ID utente alternativo	MQOD	AlternateUserid		
ID di sicurezza alternativo				

Tabella 859. Attributi, strutture dati, campi, richieste e chiamate (Continua)				
Attributo	**Struttura dei dati**	**Campo**	**Interrogazione**	**Chiamata**
opzioni di chiusura				MQCLOSE
descrizione			MQCA_Q_DESC, MQCA_Q_MGR_DESC, MQCA_PROCESS_DESC	MQINQ
nome	MQOD	ObjectName	MQCA_Q_MGR_NAME, MQCQ_Q_NAME, MQCA_PROCESS_NAME	MQINQ
Opzioni visualizzate				MQOPEN
stato aperto				MQOPEN, MQCLOSE
Identificativo gestore code	Identificativo gestore code		IDENTIFICATIVO_Q_MGR_MQCA_	MQINQ

Riferimento incrociato ImqProcess

Riferimento incrociato di attributi, richieste e chiamate per la classe C++ del record ImqAuthentication.

Tabella 860. Attributi, richieste e chiamate		
Attributo	**Interrogazione**	**Chiamata**
ID applicazione	ID_APPL_MQCA	MQINQ
tipo di applicazione	TIPO_APPL_MQI	MQINQ
Dati ambiente	DATI ENV MQCA	MQINQ
dati utente	DATI_USER_MQCA	MQINQ

Riferimento incrociato ImqPutMessageOptions

Riferimento incrociato di attributi, strutture dati e campi per la classe C++ del record ImqAuthentication.

Tabella 861. Associazione di attributi, strutture dati e campi		
Attributo	**Struttura dei dati**	**Campo**
riferimento contesto	MQPMO	Contesto
	MQPMO	InvalidDestCount
	MQPMO	KnownDestCount
opzioni	MQPMO	Opzioni
campi record	MQPMO	PutMsgRecFields
Nome del gestore code risolto	MQPMO	Nome ResolvedQMgr
Nome coda risolto	MQPMO	ResolvedQName
	MQPMO	Timeout
	MQPMO	UnknownDestCount
partecipazione punto di sincronizzazione	MQPMO	Opzioni

Riferimento incrociato ImqQueue

Riferimento incrociato di attributi, strutture dati, campi, richieste e chiamate per la classe C++ ImqQueue .

Tabella 862. Riferimento incrociato ImqQueue

Attributo	Struttura dei dati	Campo	Interrogazione	Chiamata
Nome di riaccodamento di backout			MQCA_BACKOUT_REQ_Q_NAME	MQINQ
soglia di backout			SOGLIA di backout_mqia_	MQINQ
nome coda di base			MQCA_BASE_Q_NAME	MQINQ
nome del cluster			NOME MQCA_CLUSTER	MQINQ
Nome elenco nomi cluster			ELENCO NOMI COMANDO MQCA_CLUSTER_	MQINQ
Classif. carico lavoro cluster			MQIA_CLWL_Q_RANK	MQINQ
Priorità carico lavoro cluster			MQIA_CLWL_Q_PRIORITY	MQINQ
Coda utilizzo carico di lavoro cluster			MQIA_CLWL_UTENTE	MQINQ
data di creazione			DATA CREAZIONE MQCA	MQINQ
ora di creazione			ORA_MQCA_CREAZIONE	MQINQ
Profondità corrente			MQIA_CURRENT_Q_DEPTH	MQINQ
bind predefinito			MQIA_DEF_BIND	MQINQ
Opzione predefinita di apertura in immissione			MQIA_DEF_INPUT_OPEN_OPTION	MQINQ
Persistenza predefinita			MQIA_DEF_PERSISTENZA	MQINQ
Priorità predefinita			PRIORITÀ_DEF_MQIA_	MQINQ
tipo di definizione			TIPO_DI_DEFINIZIONE MQIA	MQINQ
evento profondità elevata			MQIA_Q_DEPTH_HIGH_EVENT	MQINQ
limite massimo profondità			MQIA_Q_DEPTH_HIGH_LIMIT	MQINQ
evento profondità bassa			Q_MQIA_DEPTH_LOW_EVENT	MQINQ
limite minimo profondità			MQIA_Q_DEPTH_LOW_LIMIT	MQINQ
evento profondità massima			MQIA_Q_DEPTH_MAX_LIMIT	MQINQ

Tabella 862. Riferimento incrociato ImqQueue (Continua)

Attributo	Struttura dei dati	Campo	Interrogazione	Chiamata
elenchi di distribuzione			MQIA_DIST_LISTS	MQINQ, MQSET
Nome coda dinamica	MQOD	DynamicQName		
Ripristino forzato get			MQIA_HARDEN_GET_BACKOUT	MQINQ
Tipo di indice			TIPO_INDEX_MQI	MQINQ
inibisci get			MQIA_INIBITORI_GET	MQINQ, MQSET
inibisci inserimento			MQIA_INIB_PUT	MQINQ, MQSET
Nome coda di attivazione			NOME_Q_INIZIALIZZAZIONE_MQCA	MQINQ
Profondità massima			Q_DEPTH MQIA_MAX_	MQINQ
Lunghezza massima dei messaggi			MQIA_MAX_MSG_LENGTH	MQINQ
Sequenza di consegna messaggi			MQIA_MSG_DELIVERY_SEQUENCE	MQINQ
coda distribuita successiva				
Classe messaggi non permanente			CLASS_NPM_MQI	MQINQ
Conteggio input aperti			CONTEGGIO_INPUT_OPEN_MQIA_	MQINQ
Conteggio output aperti			MQIA_OPEN_OUTPUT_COUNT	MQINQ
coda distribuita precedente				
nome del processo			NOME_PROCESSO_MQCA	MQINQ
Conto coda			MQIA_XX_ENCODE_CASE_ONE conteggio_coda	MQINQ
Nome gestore code	MQOD	ObjectQMgrName		
Controllo coda			MQIA_MONITORING_Q	MQINQ
Informazioni coda			MQIA_STATISTICS_Q	MQINQ
tipo di coda			TIPO_Q_MQI	MQINQ
Nome gestore code remoto			MQCA_REMOTE_Q_MGR_NAME	MQINQ
Nome coda remota			MQCA_REMOTE_Q_NAME	MQINQ
Nome del gestore code risolto	MQOD	Nome ResolvedQMgr		
Nome coda risolto	MQOD	ResolvedQName		

Tabella 862. Riferimento incrociato ImqQueue (Continua)				
Attributo	**Struttura dei dati**	**Campo**	**Interrogazione**	**Chiamata**
intervallo di comservazione			INTERVAL MQIA_RETENTION_	MQINQ
ambito			SCOPE MQI	MQINQ
intervallo di servizio			INTERVALLO_Q_SERVIZIO_MQIA_	MQINQ
evento intervallo di servizio			EVENTO_INTERVALLO_QIA_SERVIZIO_ MQIA_	MQINQ
Condivisione			MQIA_CONDIVISIONABILITÀ	MQINQ
classe di memorizzazione			MQCA_STORAGE_CLASSE	MQINQ
Nome coda di trasmissione			MQCA_XMIT_Q_NAME	MQINQ
Controllo trigger			CONTROL MQIA_TRIGGER_	MQINQ, MQSET
dati del trigger			DATI MQCA_TRIGGER_	MQINQ, MQSET
Capacità di Trigger			PROFONDITÀ trigger MQIA_	MQINQ, MQSET
Priorità messaggio trigger			MQIA_TRIGGER_MSG_PRIORITY	MQINQ, MQSET
tipo trigger			TIPO_TRIGGER_MQI	MQINQ, MQSET
utilizzo			USAGO_MQI	MQINQ

Riferimento incrociato ImqQueueManager

Riferimenti incrociati di attributi, strutture dati, campi, richieste e chiamate per la classe C++ del gestore ImqQueue.

Tabella 863. Attributi, strutture dati, campi, richieste e chiamate				
Attributo	**Struttura dei dati**	**Campo**	**Interrogazione**	**Chiamata**
sovrascrittura connessioni di account			MQIA_ACCOUNTING_CONN_OVERRIDE	MQINQ
Intervallo account			INTERVALLO_ACCOUNT_MQIA_	MQINQ
registrazione attività			REGISTRAZIONE_ATTIVITÀ_MQIA_	MQINQ
Utilizza nuova verifica MCA			MQIA_ADOPTNEWMCA_CHECK	MQINQ
Adotta nuovo tipo MCA			MQIA_ADOPTNEWMCA_TIPO	MQINQ

Tabella 863. Attributi, strutture dati, campi, richieste e chiamate (Continua)

Attributo	Struttura dei dati	Campo	Interrogazione	Chiamata
tipo autenticazione	MQCSP	AuthenticationType		MQCONNX
evento autorizzazione			AUTORIZZAZIONE EVENTO MQIA_AUTHORITY_EVENT	MQINQ
Opzioni di inizio	MQBO	Opzioni		MQBEGIN
evento bridge			EVENTO_BRIDGE_MQIA_	MQINQ
Definizione automatica canale			DEF AUTOMAZIONE MQIA_CHANNEL_	MQINQ
evento di definizione automatica del canale			MQIA_CHANNEL_AUTO_EVENT	MQIA
Uscita definizione automatica canale			MQIA_CHANNEL_AUTO_EXIT	MQIA
evento del canale			EVENTO MQIA_CHANNEL_EVENT	MQINQ
Adattatori dell'iniziatore di canali			Adattatori MQIA_CHINIT_ADAPTERS	MQINQ
Controllo programma di avvio canale			CONTROL MQIA_CHINIT_	MQINQ
Dispatcher dell'iniziatore di canali			MQIA_CHINIT_DISPATCHER	MQINQ
Avvio automatico della traccia dell'iniziatore di canali			MQIA_CHINIT_TRACE_AUTO_START	MQINQ
Dimensione tabella di traccia dell'iniziatore di canali			DIMENSIONE_TABELLA_TRACCIA MQIA_CHINIT_	MQINQ
Controllo canale			CANALE_MONITORAGGIO_MQIA_	MQINQ
riferimento canale	MQCD	ChannelType		MQCONNX
Statistiche canale			MQIA_STATISTICS_CHALLEGATO	MQINQ
character set			ID_MQIA_CODED_CHAR_SET_	MQINQ
Controllo mittente cluster			MQIA_MONITORING_AUTO_CLUSSDR	MQINQ
Informazioni mittente cluster			MQIA_STATISTICS_AUTO_CLUSSDR	MQINQ
Dati carico di lavoro cluster			DATI CARICO DI LAVORO MQCA_CLUSTER_	MQINQ

Tabella 863. Attributi, strutture dati, campi, richieste e chiamate (Continua)

Attributo	Struttura dei dati	Campo	Interrogazione	Chiamata
Uscita carico di lavoro cluster			MQCA_CLUSTER_WORKLOAD_EXIT	MQINQ
Lunghezza carico di lavoro cluster			LUNGHEZZA_CARICO_LAVORO_CLUSTER_ MQIA_	MQINQ
mru carico di lavoro cluster			MQIA_CLWL_MRU_CHANNELS	MQINQ
Coda utilizzo carico di lavoro cluster			MQIA_CLWL_UTENTE	MQINQ
evento di comando			MQIA_COMMAND_EVENT	MQINQ
Nome coda di input comandi			MQCA_COMMAND_INPUT_Q_NAME	MQINQ
livello comando			LIVELLO_COMMAND_MQI	MQINQ
Controllo server di comandi			MQIA_CMD_SERVER_CONTROL	MQINQ
Opzioni di connessione	MQCNO	Opzioni		MQCONN, MQCONNX
id connessione	MQCNO	ConnectionId		MQCONNX
stato della connessione				MQCONN, MQCONNX, MQDISC
tag di connessione	MQCD	ConnTag		MQCONNX
hardware di crittografia	MQSCO	CryptoHardware		MQCONNX
nome coda di messaggi non instradabili			MQCA_DEAD_LETTER_Q_NAME	MQINQ
nome coda di trasmissione predefinito			MQCA_DEF_XMIT_Q_NAME	MQINQ
elenchi di distribuzione			MQIA_DIST_LISTS	MQINQ
gruppo dns			GRUPPO_DNS_MQCA	MQINQ
wlm dns			MQIA_DNS_WLM	MQINQ
primo record di autenticazione	MQSCO	AuthInfoRecOffset		MQCONNX
	MQSCO	AuthInfoRecPtr		MQCONNX
evento di inibizione			EVENTO_INIBITORI_MQIA_	MQINQ
Versione indirizzo IP			MQIA_IP_ADDRESS_VERSIONE	MQINQ

Tabella 863. Attributi, strutture dati, campi, richieste e chiamate (Continua)

Attributo	Struttura dei dati	Campo	Interrogazione	Chiamata
repository delle chiavi	MQSCO	KeyRepository		MQCONNX
conteggio reimpostazioni chiave	MQSCO	Conteggio KeyReset		MQCONNX
Timer listener			TIMER MQIA_LISTENER_	MQINQ
evento locale			EVENTO LOCALE MQI	MQINQ
evento logger			EVENTO LOGGER_MQI	MQINQ
Nome gruppo LU			MQCA_LU_GROUP_NAME	MQINQ
Nome LU			NOME_LU_MQCA	MQINQ
Suffisso braccio lu62			MQCA_LU62_ARM_SUFFIX	MQINQ
lu62 canali			MQIA_LU62_CHANNELS	MQINQ
numero massimo canali attivi			MQIA_ATTIVA_CHANNELS	MQINQ
Numero massimo di canali			MQIA_MAX_CHANNELS	MQINQ
numero massimo di handle			MQIA_MAX_HANDLES	MQINQ
Lunghezza massima dei messaggi			MQIA_MAX_MSG_LENGTH	MQINQ
Priorità massima			MQIA_MAX_PRIORITY	MQINQ
Num. mass. mess. non sincronizzati			MQIA_MAX_UNCOMMITTED_MSGS	MQINQ
Account MQI			MQIA_XX_ENCODE_CASE_ONE conteggio_MQI	MQINQ
Statistiche MQI			MQIA_STATISTICS_MQI	MQINQ
numero massimo di porte in uscita			MQIA_OUTBOUND_PORTA_MAX	MQINQ
minimo porta in uscita			MQIA_OUTBOUND_PORT_MIN	MQINQ
password	MQCSP	CSPPasswordPtr		MQCONNX
	MQCSP	CSPPasswordOffset		MQCONNX
	MQCSP	CSPPasswordLength		MQCONNX
evento delle prestazioni			MQIA_EVENTO_PRESTAZIONI	MQINQ

Tabella 863. Attributi, strutture dati, campi, richieste e chiamate (Continua)

Attributo	Struttura dei dati	Campo	Interrogazione	Chiamata
enterprise			PIATFORM MQIA	MQINQ
Conto coda			MQIA_XX_ENCODE_CASE_ONE conteggio_coda	MQINQ
Controllo coda			MQIA_MONITORING_Q	MQINQ
Informazioni coda			MQIA_STATISTICS_Q	MQINQ
timeout di ricezione			MQIA_RECEIVE_TIMEOUT	MQINQ
minimo timeout di ricezione			MQIA_RECEIVE_TIMEOUT_MIN	MQINQ
Tipo di timeout di ricezione			TIPO_ORA_MQIA_RECEIVE_TIMEOUT_	MQINQ
evento remoto			MQIA_REMOTE_EVENT	MQINQ
nome repository			MQCA_REPOSITORY_NAME	MQINQ
Elenco nomi repository			MQCA_REPOSITORY_NAMELIST	MQINQ
nome gestore code condiviso			MQIA_SHARED_Q_Q_MGR_NAME	MQINQ
evento ssl			EVENT_SSL_MQI	MQINQ
fip ssl			MQIA_SSL_FIPS_REQUIRED	MQINQ
Conteggio reimpostazioni chiave SSL			MQIA_SSL_RESET_COUNT	MQINQ
evento start - stop			MQIA_START_STOP_EVENT	MQINQ
intervallo statistico			STATISTICHE MQIA_INTERVAL	MQINQ
Disponibilità Syncpoint			SYNCPOINT MQI	MQINQ
canali tcp			MQIA_TCP_CHANNELS	MQINQ
Keepalive TCP			MQIA_TCP_KEEP_ALIVE	MQINQ
Nome TCP			NOME TCP_MQCA	MQINQ
Tipo di stack TCP			TIPO_STACK_TCP_MQI	MQINQ
Registrazione instradamento traccia			MQIA_TRACE_ROUTE_RECORDING	MQINQ
Intervallo trigger			INTERVALLO_TRIGGER_MQIA_	MQINQ
identificativo utente	MQCSP	Ptr CSPUserId		MQCONNX
	MQCSP	Offset CSPUserId		MQCONNX

Tabella 863. Attributi, strutture dati, campi, richieste e chiamate (Continua)				
Attributo	**Struttura dei dati**	**Campo**	**Interrogazione**	**Chiamata**
	MQCSP	CSPUserIdLunghezza		MQCONNX

Riferimento incrociato intestazione ImqReference

Riferimento incrociato di attributi, strutture dati e campi per la classe C++ del record ImqAuthentication.

Tabella 864. Associazione di attributi, strutture dati e campi		
Attributo	**Struttura dei dati**	**Campo**
ambiente di destinazione	MQRMH	DestEnvLunghezza, DestEnvOffset
nome destinazione	MQRMH	DestNameLunghezza, Offset DestName
ID istanza	MQRMH	ObjectInstanceId
lunghezza logica	MQRMH	DataLogicalLength
offset logico	MQRMH	DataLogicalOffset
offset logico 2	MQRMH	DataLogicalOffset2
tipo di riferimento	MQRMH	ObjectType
ambiente di origine	MQRMH	SrcEnvLunghezza, SrcEnvOffset
nome dell'origine	MQRMH	SrcNameLunghezza, offset SrcName

Riferimento incrociato ImqTrigger

Riferimento incrociato di attributi, strutture dati e campi per la classe C++ del record ImqAuthentication.

Tabella 865. Associazione di attributi, strutture dati e campi		
Attributo	**Struttura dei dati**	**Campo**
ID applicazione	MQM	ApplId
tipo di applicazione	MQM	ApplType
Dati ambiente	MQM	EnvData
nome del processo	MQM	ProcessName
nome coda	MQM	QName
dati del trigger	MQM	TriggerData
dati utente	MQM	UserData

Riferimento incrociato intestazione ImqWork

Riferimento incrociato di attributi, strutture dati e campi per la classe C++ del record ImqAuthentication.

Tabella 866. Associazione di attributi, strutture dati e campi		
Attributo	**Struttura dei dati**	**Campo**
token del messaggio	MQWIH	MessageToken

Tabella 866. Associazione di attributi, strutture dati e campi (Continua)		
Attributo	**Struttura dei dati**	**Campo**
nome servizio	MQWIH	ServiceName
fase di servizio	MQWIH	ServiceStep

Classe ImqAuthenticationRecord C++

Questa classe incapsula un record di informazioni di autenticazione (MQAIR) da utilizzare durante l'esecuzione del metodo ImqQueueManager: :connect, per le connessioni client TLS personalizzate.

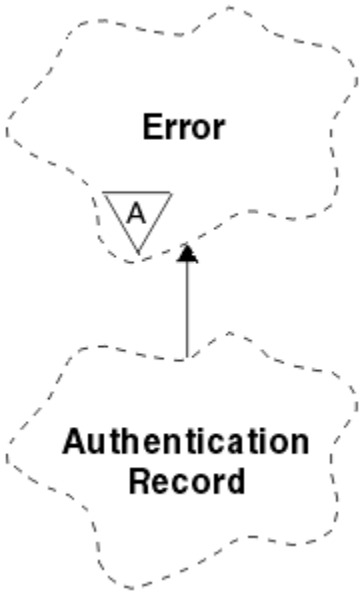


Figura 14. Classe record ImqAuthentication

Per ulteriori informazioni, consultare la descrizione del metodo ImqQueueManager: :connect. Questa classe non è disponibile sulla piattaforma z/OS .

- "Attributi oggetto" a pagina 1851
- "Costruttori" a pagina 1852
- "Metodi oggetto (pubblico)" a pagina 1852
- "Metodi oggetto (protetti)" a pagina 1853

Attributi oggetto

nome connessione
Il nome della connessione al server CRL LDAP. È l'indirizzo IP o il nome DNS, seguito facoltativamente dal numero di porta, tra parentesi.

riferimento connessione
Un riferimento a un oggetto gestore ImqQueueche fornisce la connessione richiesta a un gestore code (locale). Il valore iniziale è zero. Non confondere questo nome con il nome del gestore code che identifica un gestore code (possibilmente remoto) per una coda denominata.

record di autenticazione successivo
Oggetto successivo di questa classe, in nessun ordine particolare, con lo stesso **riferimento di connessione** di questo oggetto. Il valore iniziale è zero.

password
Una password fornita per l'autenticazione della connessione al server CRL LDAP.

record di autenticazione precedente
Oggetto precedente di questa classe, in nessun ordine particolare, con lo stesso **riferimento di connessione** di questo oggetto. Il valore iniziale è zero.

il tipo
Il tipo di informazioni di autenticazione contenute nel record.

nome utente
Un identificatore utente fornito per l'autorizzazione al server CRL LDAP.

Costruttori

ImqAuthenticationRecord ();
Il costruttore predefinito.

Metodi oggetto (pubblico)

void operator = (const ImqAuthenticationRecord &*air*);
Copia i dati di istanza da *air*, sostituendo i dati di istanza esistenti.

const ImqString & connectionName () const;
Restituisce il **nome connessione**.

void setConnectionName (const ImqString &*name*);
Imposta il **Nome connessione**.

void setConnectionNome (const char **nome* = 0);
Imposta il **Nome connessione**.

ImqQueueImqQueue * connectionReference () const;
Restituisce il **riferimento connessione**.

void setConnectionRiferimento (ImqQueueGestore &*gestore*);
Imposta il **riferimento connessione**.

void setConnectionReference (ImqQueueGestore **gestore* = 0);
Imposta il **riferimento connessione**.

void copyOut (MQAIR **pAir*);
Copia i dati di istanza in *pAir*, sostituendo i dati di istanza esistenti. Ciò potrebbe comportare l'allocazione di memoria dipendente.

void clear (MQAIR **pAir*);
Elimina la struttura e rilascia la memoria dipendente a cui fa riferimento *pAir*.

ImqAuthenticationRecord * nextAuthenticationRecord () const;
Restituisce il **record di autenticazione successivo**.

const ImqString & password () const;
Restituisce la **password**.

void setPassword (const ImqString &*password*);
Imposta la **password**.

void setPassword (const char +*password* = 0);
Imposta la **password**.

ImqAuthenticationRecord * previousAuthenticationRecord () const;
Restituisce il **record di autenticazione precedente**.

Tipo MQLONG () const;
Restituisce il **tipo**.

void setType (const MQLONG *tipo*);
Imposta il **tipo**.

const ImqString & userName () const;
Restituisce il **nome utente**.

void setUserName (const ImqString & *name*);
Imposta il **nome utente**.

void setUserNome (const char * *nome* = 0);
Imposta il **nome utente**.

Metodi oggetto (protetti)

void setNextAuthenticationRecord (ImqAuthenticationRecord * *pAir* = 0);
Imposta il **record di autenticazione successivo**.

Attenzione: utilizzare questa funzione solo se si è certi che non interromperà l'elenco di record di autenticazione.

void setPreviousAuthenticationRecord (ImqAuthenticationRecord * *pAir* = 0);
Imposta il **record di autenticazione precedente**.

Attenzione: utilizzare questa funzione solo se si è certi che non interromperà l'elenco di record di autenticazione.

classe C++ ImqBinary

Questa classe comprende un array di byte binari che può essere utilizzato per i valori ImqMessage **accounting token**, **ID correlazione**e **id messaggio** . Consente facile assegnazione, copia e confronto.

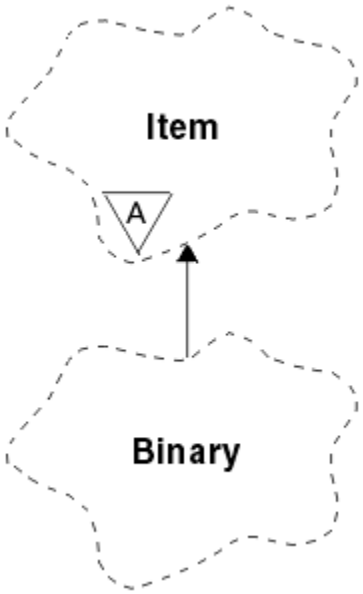


Figura 15. classe ImqBinary

Attributi oggetto

dati
Un array di byte di dati binari. Il valore iniziale è null.

Lunghezza dati
Il numero di byte. Il valore iniziale è zero.

puntatore dati
L'indirizzo del primo byte dei **dati**. Il valore iniziale è zero.

Costruttori

ImqBinary( );
Il costruttore predefinito.

ImqBinary(const ImqBinary & ***binario*** **);**
Il costruttore di copia.

ImqBinary(const void * ***dati*****, const size_t** ***lunghezza*** **);**
Copia *lunghezza* byte da *dati*.

Metodi ImqItem sovraccaricati

ImqBoolean copyOut (ImqMessage & ***msg*** **);**
Copia i **dati** nel buffer di messaggi, sostituendo qualsiasi contenuto esistente. Imposta il formato *msg* su MQFMT_NONE.

Consultare la descrizione del metodo della classe ImqItem per ulteriori dettagli.

ImqBoolean pasteIn (ImqMessage & ***msg*** **);**
Imposta i **dati** trasferendo i dati rimanenti dal buffer di messaggio, sostituendo i **dati**esistenti.

Per avere esito positivo, il formato di ImqMessage deve essere MQFMT_NONE.

Consultare la descrizione del metodo della classe ImqItem per ulteriori dettagli.

Metodi oggetto (pubblico)

void operator = (const ImqBinary & ***binario*** **);**
Copia byte da *binario*.

ImqBoolean operatore == (const ImqBinary & ***binario*** **);**
Confronta questo oggetto con *binario*. Restituisce FALSE se non è uguale e TRUE in caso contrario. Gli oggetti sono uguali se hanno la stessa **lunghezza dati** e i byte corrispondono.

ImqBoolean copyOut (void * ***buffer*****, const size_t** ***length*****, const char** ***pad*** **= 0);**
Copia fino a *lunghezza* byte dal **puntatore dati** al *buffer*. Se la **lunghezza dei dati** non è sufficiente, lo spazio rimanente nel *buffer* viene riempito con *riempimento* byte. *buffer* può essere zero se anche *lunghezza* è zero. *length* non deve essere negativo. Restituisce TRUE in caso di esito positivo.

dimensione_t dataLength () const ;
Restituisce la **lunghezza dei dati**.

ImqBoolean setDataLunghezza (const size_t ***lunghezza*** **);**
Imposta la **lunghezza dati**. Se la **lunghezza dei dati** viene modificata come risultato di questo metodo, i dati nell'oggetto non vengono inizializzati. Restituisce TRUE in caso di esito positivo.

void * dataPointer () const ;
Restituisce il **puntatore dati**.

ImqBoolean isNull () const ;
Restituisce TRUE se la **lunghezza dati** è zero o se tutti i **dati** byte sono zero. Altrimenti restituisce FALSE.

ImqBoolean set (const void * ***buffer*****, const size_t** ***lunghezza*** **);**
Copia *lunghezza* byte da *buffer*. Restituisce TRUE in caso di esito positivo.

Metodi oggetto (protetti)

vuoto clear ();
Riduce la **lunghezza dati** a zero.

Codici di origine

- MQRC_NO_BUFFER
- MQRC_STORAGE_NON_DISPONIBILE
- MQRC_INCONSISTENT_FORMAT

Classe C++ ImqCache

Utilizzare questa classe per conservare o eseguire il marshalling dei dati in memoria.

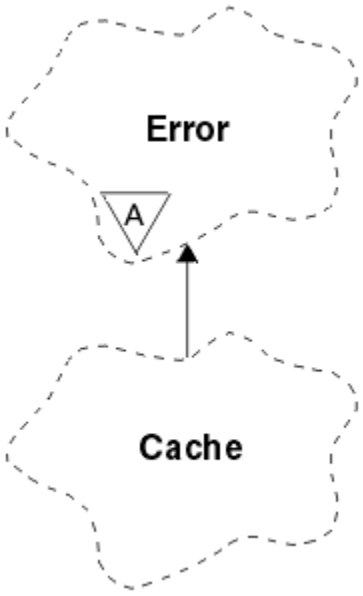


Figura 16. Classe ImqCache

Utilizzare questa classe per conservare o eseguire il marshalling dei dati in memoria. È possibile denominare un buffer di memoria di dimensione fissa oppure il sistema può fornire automaticamente una quantità di memoria flessibile. Questa classe è relativa alle chiamate MQI elencate in "Riferimento incrociato ImqCache" a pagina 1836.

Attributi oggetto

buffer automatico
Indica se la memoria di buffer viene gestita automaticamente dal sistema (TRUE) o viene fornita dall'utente (FALSE). Inizialmente è impostato su TRUE.

Questo attributo non è impostato direttamente. Viene impostato indirettamente utilizzando il metodo **useEmptyBuffer** o il metodo **useFullBuffer** .

Se viene fornita memoria utente, questo attributo è FALSE, la memoria del buffer non può crescere e possono verificarsi errori di overflow del buffer. L'indirizzo e la lunghezza del buffer rimangono costanti.

Se la memoria utente non viene fornita, questo attributo è TRUE e la memoria di buffer può crescere in modo incrementale per contenere una quantità arbitraria di dati del messaggio. Tuttavia, quando il buffer cresce, l'indirizzo del buffer potrebbe cambiare, quindi prestare attenzione quando si utilizzano il **puntatore del buffer** e il **puntatore dei dati**.

lunghezza del buffer
Il numero di byte di memoria nel buffer. Il valore iniziale è zero.

puntatore buffer
L'indirizzo della memoria buffer. Il valore iniziale è null.

Lunghezza dati
Il numero di byte successivi al **puntatore dati**. Deve essere uguale o inferiore alla **lunghezza del messaggio**. Il valore iniziale è zero.

Offset dati
Il numero di byte che precedono il **puntatore dati**. Deve essere uguale o inferiore alla **lunghezza del messaggio**. Il valore iniziale è zero.

puntatore dati
L'indirizzo della parte del buffer che deve essere scritta o letta dal successivo. Il valore iniziale è null.

lunghezza messaggio
Il numero di byte di dati significativi nel buffer. Il valore iniziale è zero.

Costruttori

ImqCache( );
Il costruttore predefinito.

ImqCache(const ImqCache & ***cache*** **);**
Il costruttore di copia.

Metodi oggetto (pubblico)

void operatore = (const ImqCache & ***cache*** **);**
Copia fino a **lunghezza messaggio** byte di dati dall'oggetto *cache* all'oggetto. Se **buffer automatico** è FALSE, la **lunghezza buffer** deve essere già sufficiente per contenere i dati copiati.

ImqBoolean automaticBuffer () const ;
Restituisce il valore **buffer automatico** .

dimensione_t bufferLength () const ;
Restituisce la **lunghezza del buffer**.

char * bufferPointer () const ;
Restituisce il **puntatore del buffer**.

void clearMessage ();
Imposta la **lunghezza del messaggio** e l' **offset dati** su zero.

dimensione_t dataLength () const ;
Restituisce la **lunghezza dei dati**.

dimensione_t dataOffset () const ;
Restituisce l' **offset di dati**.

ImqBoolean setDataOffset (const size_t ***offset*** **);**
Imposta l' **offset di dati**. La **lunghezza del messaggio** viene aumentata se necessario per garantire che non sia inferiore all' **offset di dati**. Questo metodo restituisce TRUE se ha esito positivo.

char * dataPointer () const ;
Restituisce una copia del **puntatore dati**.

size_t messageLength () const ;
Restituisce la **lunghezza del messaggio**.

ImqBoolean setMessageLunghezza (const size_t ***lunghezza*** **);**
Imposta la **lunghezza del messaggio**. Aumenta la **lunghezza del buffer** se necessario per garantire che la **lunghezza del messaggio** non sia maggiore della **lunghezza buffer**. Riduce l' **offset dei dati** se necessario per garantire che non sia maggiore della **lunghezza del messaggio**. Restituisce TRUE in caso di esito positivo.

ImqBoolean moreBytes (const size_t *byte - obbligatorio*);
Assicura che siano disponibili *byte - richiesti* in più byte (per la scrittura) tra il **puntatore dati** e la fine del buffer. Restituisce TRUE in caso di esito positivo.

Se **buffer automatico** è TRUE, viene acquisita ulteriore memoria come richiesto; altrimenti, la **lunghezza del buffer** deve essere già adeguata.

ImqBoolean read (const size_t *lunghezza*, char * & *buffer esterno*);
Copia i byte di *lunghezza* , dal buffer a partire dalla posizione **puntatore dati** , nel *buffer esterno*. Una volta copiati i dati, l' **offset di dati** viene aumentato di *lunghezza*. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean resizeBuffer (const_size_t *lunghezza*);
Varia la **lunghezza del buffer**, purché il **buffer automatico** sia TRUE. Ciò si ottiene riassegnando la memoria del buffer. Fino a **lunghezza del messaggio** byte di dati dal buffer esistente vengono copiati nel nuovo buffer. Il numero massimo copiato è *lunghezza* byte. Il **puntatore del buffer** viene modificato. La **lunghezza del messaggio** e lo **scostamento dati** vengono conservati il più possibile entro i limiti del nuovo buffer. Restituisce TRUE in caso di esito positivo e FALSE se il **buffer automatico** è FALSE.

Nota: Questo metodo può non riuscire con MQRC_STORAGE_NOT_AVAILABLE se si verifica un problema con le risorse di sistema.

ImqBoolean useEmptyBuffer (const char * *external - buffer*, const size_t *lunghezza*);
Identifica un buffer utente vuoto, impostando il **puntatore del buffer** in modo che punti al *buffer esterno*, la **lunghezza del buffer** in *lunghezza*e la **lunghezza del messaggio** su zero. Esegue un **clearMessage**. Se il buffer è completamente pieno di dati, utilizzare invece il metodo **useFullBuffer** . Se il buffer è parzialmente pieno di dati, utilizzare il metodo **setMessageLength** per indicare la quantità corretta. Questo metodo restituisce TRUE se ha esito positivo.

Questo metodo può essere utilizzato per identificare una quantità fissa di memoria, come descritto precedentemente (*external - buffer* non è null e *length* non è zero), nel qual caso **automatic buffer** è impostato su FALSE o può essere utilizzato per ripristinare la memoria flessibile gestita dal sistema (*external - buffer* è null e *length* è zero), nel qual caso **automatic buffer** è impostato su TRUE.

ImqBoolean useFullBuffer (const char * *externalBuffer*, const size_t *lunghezza*);
Come per **useEmptyBuffer**, con la differenza che la **lunghezza del messaggio** è impostata su *length*. Restituisce TRUE in caso di esito positivo.

ImqBoolean write (size_t const *lunghezza*, const char * *buffer esterno*);
Copia *lunghezza* byte, dal *buffer esterno*, nel buffer a partire dalla posizione **puntatore dati** . Dopo che i dati sono stati copiati, l' **offset di dati** viene aumentato di *lunghezza*e la **lunghezza del messaggio** viene aumentata se necessario per garantire che non sia inferiore al nuovo valore di **offset di dati** . Questo metodo restituisce TRUE se ha esito positivo.

Se il **buffer automatico** è TRUE, viene garantita una quantità di memoria adeguata; altrimenti, l' **offset di dati** finale non deve superare la **lunghezza del buffer**.

Codici di origine

- MQRC_BUFFER_NOT_AUTOMATIC
- DATA_TRUNCATED MQRC_
- MQRC_BUFFER insufficiente
- DATI MQRC_INSUFFICIENT_
- NULL_POINTER MQRC
- MQRC_STORAGE_NON_DISPONIBILE
- LENGTH - ZERO_MQRC

Classe ImqChannel C++

Questa classe incapsula una definizione di canale (MQCD) da utilizzare durante l'esecuzione del metodo Manager: :connect, per connessioni client personalizzate.

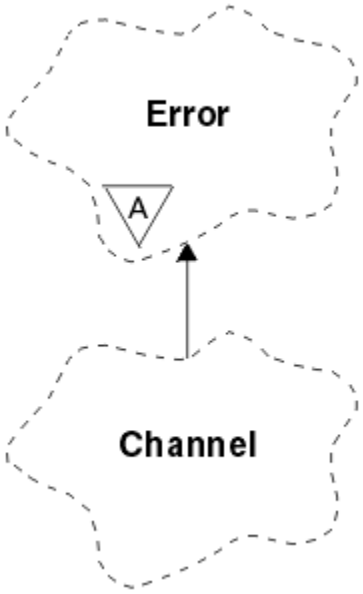


Figura 17. Classe ImqChannel

Per ulteriori informazioni, consultare la descrizione del metodo Manager: :connect e Programma di esempio HELLO WORLD (imqwrld.cpp).

Non tutti i metodi elencati sono applicabili a tutte le piattaforme. Per ulteriori informazioni, consultare le descrizioni dei comandi DEFINE CHANNEL e ALTER CHANNEL .

La classe ImqChannel non è supportata su z/OS.

- "Attributi oggetto" a pagina 1858
- "Costruttori" a pagina 1859
- "Metodi oggetto (pubblico)" a pagina 1860
- "Codici di origine" a pagina 1863

Attributi oggetto

battito cardiaco batch
Il numero di millesimi di secondo tra le verifiche che un canale remoto è attivo. Il valore iniziale è 0.

nome canale
Il nome del canale. Il valore iniziale è null.

nome connessione
Il nome della connessione. Ad esempio, l'indirizzo IP di un computer host. Il valore iniziale è null.

Compressione intestazione
L'elenco delle tecniche di compressione dei dati di intestazione supportate dal canale. I valori iniziali sono tutti impostati su MQCOMPRESS_NOT_AVAILABLE.

intervallo heartbeat
Il numero di secondi tra le verifiche che una connessione è ancora in funzione. Il valore iniziale è 300.

Intervallo keep alive
Il numero di secondi passati allo stack di comunicazione specificando il tempo keep alive per il canale. Il valore iniziale è MQKAI_AUTO.

indirizzo locale
L'indirizzo di comunicazione locale per il canale.

Lunghezza massima dei messaggi
La lunghezza massima del messaggio supportato dal canale in una singola comunicazione. Il valore iniziale è 4 194 304.

Compressione messaggi
L'elenco delle tecniche di compressione dei dati di dei messaggi supportate dal canale. I valori iniziali sono tutti impostati su MQCOMPRESS_NOT_AVAILABLE.

nome modaltà
Il nome della modalità. Il valore iniziale è null.

password
Una password fornita per l'autenticazione della connessione. Il valore iniziale è null.

conteggio uscite ricezione
Il numero di uscite di ricezione. Il valore iniziale è zero. Questo attributo è di sola lettura.

nomi delle uscite di ricezione
I nomi delle uscite di ricezione.

ricevere i dati utente
Dati associati alle uscite di ricezione.

Nome uscita di sicurezza
Il nome di un'uscita di sicurezza da richiamare sul server della connessione. Il valore iniziale è null.

dati utente di protezione
Dati da trasmettere all'exit di sicurezza. Il valore iniziale è null.

conteggio uscite di invio
Il numero di uscite di invio. Il valore iniziale è zero. Questo attributo è di sola lettura.

nomi uscita di invio
I nomi delle uscite di invio.

invia dati utente
Dati associati alle uscite di invio.

SSL CipherSpec
CipherSpec per l'utilizzo con TLS.

Tipo di autenticazione client SSL
Tipo di autenticazione client da utilizzare con TLS.

Nome peer SSL
Nome peer da utilizzare con TLS.

Nome programma transazioni
Il nome del programma di transazione. Il valore iniziale è null.

tipo di trasporto
Il tipo di trasporto della connessione. Il valore iniziale è MQXPT_LU62.

identificativo utente
Un identificativo utente fornito per l'autorizzazione. Il valore iniziale è null.

Costruttori

ImqChannel( ) ;
Il costruttore predefinito.

ImqChannel(const ImqChannel & ***canale*** **);**
Il costruttore di copia.

Metodi oggetto (pubblico)

operatore void = (const ImqChannel & *canale*);
Copia i dati di istanza dal *canale*, sostituendo i dati di istanza esistenti.

MQLONG batchHeartBeat () const;
Restituisce l' **heartbeat batch**.

ImqBoolean setBatchHeartBeat(const MQLONG *heartbeat* = 0L);
Imposta l' **heartbeat batch**. Questo metodo restituisce TRUE se ha esito positivo.

ImqString channelName() const;
Restituisce il **nome canale**.

ImqBoolean setChannelNome (const char * *nome* = 0);
Imposta il **nome canale**. Questo metodo restituisce TRUE se ha esito positivo.

ImqString connectionName() const;
Restituisce il **nome connessione**.

ImqBoolean setConnectionNome (const char * *nome* = 0);
Imposta il **Nome connessione**. Questo metodo restituisce TRUE se ha esito positivo.

size_t headerCompressionConteggio () const;
Restituisce il conteggio delle tecniche di compressione dei dati di intestazione supportate.

ImqBoolean headerCompression(conteggio const size_t, MQLONG compress []) const;
Restituisce le copie delle tecniche di compressione dei dati di intestazione supportate in **compress**. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setHeaderCompressione (conteggio const size_t, const MQLONG compress []);

Imposta le tecniche di compressione dei dati di intestazione supportate su **compress**.

Imposta il conteggio delle tecniche di compressione dei dati di intestazione supportate su **count**.

Questo metodo restituisce TRUE se ha esito positivo.

Intervallo heartBeatMQLONG () const;
Restituisce l' **intervallo heartbeat**.

ImqBoolean setHeartBeatInterval(const MQLONG *interval* = 300L);
Imposta l' **intervallo heartbeat**. Questo metodo restituisce TRUE se ha esito positivo.

Intervallo () MQLONG keepAlive const;
Restituisce l' **intervallo keep alive**.

ImqBoolean setKeepAliveInterval(const MQLONG *intervallo* = MQKAI_AUTO);
Imposta l' **intervallo keep alive**. Questo metodo restituisce TRUE se ha esito positivo.

ImqString localAddress() const;
Restituisce l' **indirizzo locale**.

ImqBoolean setLocalAddress (const char * *address* = 0);
Imposta l' **indirizzo locale**. Questo metodo restituisce TRUE se ha esito positivo.

MQLONG maximumMessageLunghezza () const;
Restituisce la **lunghezza massima del messaggio**.

ImqBoolean setMaximumMessageLength(const MQLONG *lunghezza* = 4194304L);
Imposta la **lunghezza massima del messaggio**. Questo metodo restituisce TRUE se ha esito positivo.

size_t messageCompressionConteggio () const;
Restituisce il conteggio delle tecniche di compressione dei dati dei messaggi supportate.

ImqBoolean messageCompression(conteggio const size_t, MQLONG compress []) const;
Restituisce copie delle tecniche di compressione dei dati dei messaggi supportate in **compress**. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setMessageCompressione (const size_t count, const MQLONG compress []);

Imposta le tecniche di compressione dei dati dei messaggi supportate da comprimere.

Imposta il conteggio delle tecniche di compressione dei dati dei messaggi supportate.

Questo metodo restituisce TRUE se ha esito positivo.

ImqString modeName() const;
Restituisce il **nome modalità**.

ImqBoolean setModeNome (const char * *nome* = 0);
Imposta il **nome modalità**. Questo metodo restituisce TRUE se ha esito positivo.

Password ImqString () const;
Restituisce la **password**.

ImqBoolean setPassword(const char * *password* = 0);
Imposta la **password**. Questo metodo restituisce TRUE se ha esito positivo.

size_t receiveExitConteggio () const;
Restituisce il **conteggio uscite di ricezione**.

ImqString receiveExitNome ();
Restituisce il primo dei **nomi delle uscite di ricezione**, se presenti. Se il **conteggio uscite di ricezione** è zero, restituisce una stringa vuota.

ImqBoolean receiveExitNomi (const size_t *count*, ImqString * *names* []);
Restituisce copie dei **nomi delle uscite di ricezione** in *names*. Imposta i *nomi* in eccesso di **conteggio uscite di ricezione** su stringhe null. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setReceiveExitName(const char * *name* = 0);
Imposta i **nomi di uscita ricezione** sul singolo *nome*. *nome* può essere vuoto o null. Imposta il **conteggio uscite di ricezione** su 1 o zero. Cancella i dati utente di **ricezione**. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setReceiveExitNames(const size_t *count*, const char * *names* []);
Imposta i **nomi delle uscite di ricezione** su *nomi*. I singoli valori *nomi* non devono essere vuoti o null. Imposta il **conteggio uscite di ricezione** su *conteggio*. Cancella i dati utente di **ricezione**. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setReceiveExitNames(const size_t *count*, const ImqString * *names* []);
Imposta i **nomi delle uscite di ricezione** su *nomi*. I singoli valori *nomi* non devono essere vuoti o null. Imposta il **conteggio uscite di ricezione** su *conteggio*. Cancella i dati utente di **ricezione**. Questo metodo restituisce TRUE se ha esito positivo.

ImqString receiveUserData ();
Restituisce il primo degli elementi **receive user data** , se presenti. Se il **conteggio uscite di ricezione** è zero, restituisce una stringa vuota.

Dati ImqBoolean receiveUser(const size_t *count*, ImqString * *data* []);
Restituisce copie degli elementi **receive user data** in *data*. Imposta tutti i *dati* in eccesso di **conteggio uscite di ricezione** su stringhe null. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setReceiveUserData(const char * *data* = 0);
Imposta **ricezione dati utente** sui dati *elemento singolo*. Se *data* non è null, **conteggio uscite di ricezione** deve essere almeno 1. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setReceiveUserData(const size_t *count*, const char * *data* []);
Imposta **receive user data** su *data*. *count* non deve essere maggiore del **conteggio uscite di ricezione**. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setReceiveUserData(const size_t *count*, const ImqString * *data* []);
Imposta **receive user data** su *data*. *count* non deve essere maggiore del **conteggio uscite di ricezione**. Questo metodo restituisce TRUE se ha esito positivo.

ImqString securityExitNome () const;
Restituisce il **nome dell'uscita di sicurezza**.

ImqBoolean setSecurityExitName(const char * *name* = 0);
Imposta il **nome dell'uscita di sicurezza**. Questo metodo restituisce TRUE se ha esito positivo.

Dati ImqString securityUser() const;
Restituisce i **dati utente di sicurezza**.

ImqBoolean setSecurityUserData(const char * *data* = 0);
Imposta i **dati utente di sicurezza**. Questo metodo restituisce TRUE se ha esito positivo.

size_t sendExitConteggio () const;
Restituisce il **conteggio uscite di invio**.

ImqString sendExitNome ();
Restituisce il primo dei **nomi delle uscite di invio**, se presenti. Restituisce una stringa vuota se il **conteggio uscite di invio** è zero.

ImqBoolean sendExitNames (const size_t *count*, ImqString * *names* []);
Restituisce copie dei **nomi di uscita di invio** in *names*. Imposta tutti i *nomi* in eccesso di **conteggio uscite di invio** su stringhe null. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setSendExitName(const char * *name* = 0);
Imposta i **nomi di uscita di invio** sul singolo *nome*. *nome* può essere vuoto o null. Imposta il **conteggio uscite di invio** su 1 o zero. Cancella i **dati utente di invio**. Questo metodo restituisce TRUE se ha esito positivo

ImqBoolean setSendExitNames(const size_t *count*, const char * *names* []);
Imposta i **nomi di uscita invio** su *nomi*. I singoli valori *nomi* non devono essere vuoti o null. Imposta il **conteggio uscite di invio** su *conteggio*. Cancella i **dati utente di invio**. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setSendExitNames(const size_t *count*, const ImqString * *names* []);
Imposta i **nomi di uscita invio** su *nomi*. I singoli valori *nomi* non devono essere vuoti o null. Imposta il **conteggio uscite di invio** su *conteggio*. Cancella i **dati utente di invio**. Questo metodo restituisce TRUE se ha esito positivo.

ImqString sendUserData ();
Restituisce il primo degli elementi **send user data** , se presenti. Restituisce una stringa vuota se il **conteggio uscite di invio** è zero.

ImqBoolean sendUserData (const size_t *count*, ImqString * *data* []);
Restituisce copie degli elementi **send user data** in *data*. Imposta i *dati* in eccesso di **conteggio uscite di invio** su stringhe null. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setSendUserData(const char * *data* = 0);
Imposta l'opzione **invia dati utente** ai *dati*dell'elemento singolo. Se *data* non è null, il **conteggio uscite di invio** deve essere almeno 1. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setSendUserData(const size_t *count*, const char + *data* []);
Imposta **send user data** su *data*. *count* non deve essere maggiore del **conteggio uscite di invio**. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setSendUserData(const size_t *count*, const ImqString * *data* []);
Imposta **send user data** su *data*. *count* non deve essere maggiore del **conteggio uscite di invio**. Questo metodo restituisce TRUE se ha esito positivo.

ImqString sslCipherSpecifica () const;
Restituisce la specifica di cifratura TLS.

ImqBoolean setSslCipherSpecification(const char * *name* = 0);
Imposta la specifica di cifratura TLS. Questo metodo restituisce TRUE se ha esito positivo.

Autenticazione MQLONG sslClient() const;
Restituisce il tipo di autenticazione client TLS.

ImqBoolean setSslClientAuthentication(const MQLONG *auth* = MQSCA_REQUIRED);
Imposta il tipo di autenticazione client TLS. Questo metodo restituisce TRUE se ha esito positivo.

ImqString sslPeerNome () const;
Restituisce il nome peer TLS.

ImqBoolean setSslPeerName(const char * *name* = 0);
Imposta il nome peer TLS. Questo metodo restituisce TRUE se ha esito positivo.

ImqString transactionProgramNome () const;
Restituisce il **nome del programma di transazione**.

ImqBoolean setTransactionProgramName(const char * *name* = 0);
Imposta il **Nome programma di transazioni**. Questo metodo restituisce TRUE se ha esito positivo.

MQLONG transportType() const;
Restituisce il **tipo di trasporto**.

ImqBoolean setTransportTipo (const MQLONG *tipo* = MQXPT_LU62);
Imposta il **tipo di trasporto**. Questo metodo restituisce TRUE se ha esito positivo.

ImqString userId() const;
Restituisce l' **ID utente**.

ImqBoolean setUserId (const char * *id* = 0);
Imposta l' **id utente**. Questo metodo restituisce TRUE se ha esito positivo.

Codici di origine

- ERRORE MQRC_DATA_LENGTH
- ERRORE CONTEGGIO_ERRORI MQRC_IT
- NULL_POINTER MQRC
- ERRORE_ORIGINE_RISORSE MQRC

Classe ImqCICSBridgeHeader C++

Questa classe incapsula funzioni specifiche della struttura dati MQCIH.

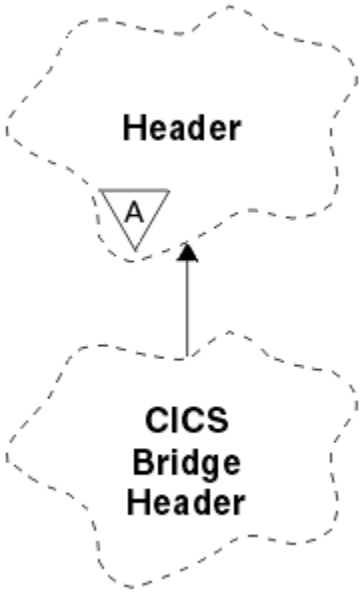


Figura 18. classe ImqCICSBridgeHeader

Gli oggetti di questa classe vengono utilizzati dalle applicazioni che inviano messaggi a CICS bridge tramite IBM MQ for z/OS.

- "Attributi oggetto" a pagina 1864
- "Costruttori" a pagina 1866
- "Metodi ImqItem sovraccaricati" a pagina 1866
- "Metodi oggetto (pubblico)" a pagina 1866
- "Dati oggetto (protetti)" a pagina 1868
- "Codici di origine" a pagina 1869
- "Codici di ritorno" a pagina 1869

Attributi oggetto

Descrittore ADS
Descrittore ADS di invio / ricezione. Viene impostato utilizzando MQCADSD_NONE. Il valore iniziale è MQCADSD_NONE. Sono possibili i seguenti valori aggiuntivi:

- MQCADSD_NONE
- MQCADSD_INVIA
- MQCADSD_RECV
- MQCADSD_MSGFORMATO

AID (attention identifier)
Tasto AID. Il campo deve essere di lunghezza MQ_ATTENION_ID_LENGTH.

programma di autenticazione
RACF password o passticket. Il valore iniziale contiene spazi, di lunghezza MQ_AUTHENTICATOR_LENGTH.

codice di interruzione bridge
Codice di fine anomala bridge, di lunghezza MQ_ABEND_CODE_LENGTH. Il valore iniziale è di quattro caratteri vuoti. Il valore restituito in questo campo dipende dal codice di ritorno. Per ulteriori dettagli, vedere Tabella 867 a pagina 1869.

codice di annullamento bridge
Codice transazione di fine anomala del bridge. Il campo è riservato, deve contenere spazi vuoti e deve essere di lunghezza MQ_CANCEL_CODE_LENGTH.

codice di completamento bridge
Codice di completamento, che può contenere il codice di completamento IBM MQ o il valore EIBRESP CICS . Il campo ha il valore iniziale MQCC_OK. Il valore restituito in questo campo dipende dal codice di ritorno. Per ulteriori dettagli, vedere Tabella 867 a pagina 1869.

offset errore bridge
Scostamento errore bridge. Il valore iniziale è zero. Questo attributo è di sola lettura.

codice motivo bridge
Codice di errore. Questo campo può contenere il motivo IBM MQ o il valore CICS EIBRESP2 . Il campo ha il valore iniziale di MQRC_NONE. Il valore restituito in questo campo dipende dal codice di ritorno. Per ulteriori dettagli, vedere Tabella 867 a pagina 1869.

codice di ritorno bridge
Codice di ritorno da CICS bridge. Il valore iniziale è MQCRC_OK.

attività di conversazione
Se l'attività può essere conversazionale. Il valore iniziale è MQCCT_NO. Sono possibili i seguenti valori aggiuntivi:

- SÌ MQCC
- MQCCT_NO

posizione cursore
Posizione del cursore. Il valore iniziale è zero.

tempo di conservazione facility
Ora di rilascio della funzione CICS bridge .

struttura come
Attributo emulato del terminale. Il campo deve essere di lunghezza MQ_FACILITY_LIKE_LENGTH.

token funzione
Valore token BVT. Il campo deve essere di lunghezza MQ_FACILITY_LENGTH. Il valore iniziale è MQCFAC_NONE.

funzione
Funzione, che può contenere il nome della chiamata IBM MQ o la funzione CICS EIBFN. Il campo ha il valore iniziale di MQCFUNC_NONE, con lunghezza MQ_FUNCTION_LENGTH. Il valore restituito in questo campo dipende dal codice di ritorno. Per ulteriori dettagli, vedere Tabella 867 a pagina 1869.

I seguenti valori aggiuntivi sono possibili quando **funzione** contiene un nome chiamata IBM MQ :

- MQCFUN_MQCONN
- MQCFUN_MQGET
- MQCFUN_MQINQ
- MQCFUN_NONE
- MQCFUN_MQOPEN
- MQCFUN_PUT
- MQCFUNC_MQPUT1

ottieni intervallo di attesa
Intervallo di attesa per una chiamata MQGET emessa dall'attività CICS bridge . Il valore iniziale è MQCGWI_DEFAULT. Il campo viene applicato solo quando **uow control** ha il valore MQCUOWC_FIRST. Sono possibili i seguenti valori aggiuntivi:

- MQCGWI_DEFAULT
- MQWI_ILLIMITATO

tipo di link
Tipo di collegamento. Il valore iniziale è MQCLT_PROGRAM. Sono possibili i seguenti valori aggiuntivi:

- PROGRAMMA_MQCL
- TRANSAZIONE MQCLT

identificativo transazione successiva
ID della transazione successiva da allegare. Il campo deve essere di lunghezza MQ_TRANSACTION_ID_LENGTH.

lunghezza dati di emissione
Lunghezza dati COMMAREA. Il valore iniziale è MQCODL_AS_INPUT.

formato reply - to
Nome formato del messaggio di risposta. Il valore iniziale è MQFMT_NONE con lunghezza MQ_FORMAT_LENGTH.

codice di avvio
Codice di inizio transazione. Il campo deve essere di lunghezza MQ_START_CODE_LENGTH. Il valore iniziale è MQCSC_NONE. Sono possibili i seguenti valori aggiuntivi:

- INIZIO_MQCSC
- DATI STAR MQCSC
- MQCSC_TERMINPUT
- MQCSC_NONE

stato fine attività
Stato di fine attività. Il valore iniziale è MQCTES_NOSYNC. Sono possibili i seguenti valori aggiuntivi:

- COMMIT MQCTES
- BACKOUT MQCTES
- ENDTASK MQCTES
- NOSYNC MQCTES

identificativo transazione
ID della transazione da allegare. Il valore iniziale deve contenere spazi e deve essere di lunghezza MQ_TRANSACTION_ID_LENGTH. Il campo si applica solo quando **uow control** ha il valore MQCUOWC_FIRST o MQCUOWC_ONLY.

Controllo UOW
Controllo UOW. Il valore iniziale è MQCUOWC_ONLY. Sono possibili i seguenti valori aggiuntivi:

- MQCUOWC_FIRST

- MQCUOWC_MEDIO
- LAST MQCUOWC
- SOLO MQCUOWC_
- COMMIT MQCUOWC
- BACKOUT MQCUOWC
- MQCUOWC_CONTINUA

versione
Il numero di versione MQCIH. Il valore iniziale è MQCIH_VERSION_2. L'unico altro valore supportato è MQCIH_VERSION_1.

Costruttori

ImqCICSBridgeHeader();
Il costruttore predefinito.

ImqCICSBridgeHeader(CICSBridgeHeader & *intestazione const*);
Il costruttore di copia.

Metodi ImqItem sovraccaricati

ImqBoolean copyOutvirtuale (ImqMessage & *msg*);
Inserisce una struttura dati MQCIH nel buffer dei messaggi all'inizio, spostando ulteriormente i dati dei messaggi esistenti e impostandone il formato su MQFMT_CICS.

Consultare la descrizione del metodo della classe parent per ulteriori dettagli.

ImqBoolean pasteInvirtuale (ImqMessage & *msg*);
Legge una struttura dati MQCIH dal buffer di messaggio. Per avere esito positivo, la codifica dell'oggetto *msg* deve essere MQENC_NATIVE. Richiamare i messaggi con MQGMO_CONVERT in MQENC_NATIVE. Per avere esito positivo, il formato di ImqMessage deve essere MQFMT_CICS.

Consultare la descrizione del metodo della classe parent per ulteriori dettagli.

Metodi oggetto (pubblico)

operatore void = (const ImqCICSBridgeHeader & *intestazione*);
Copia i dati di istanza dall'intestazione , sostituendo i dati di istanza esistenti.

MQLONG ADSDescriptor () const;
Restituisce una copia del **descrittore ADS**.

void setADSDescriptor(const MQLONG *descrittore* = MQCADSD_NONE);
Imposta il **descrittore ADS**.

ImqString attentionIdentifier() const;
Restituisce una copia dell' **identificativo di attenzione**, riempito con spazi finali di lunghezza MQ_ATTENION_ID_LENGTH.

void setAttentionIdentifier (const char * *data* = 0);
Imposta l' **identificativo di attenzione**, riempito con spazi finali di lunghezza MQ_ATTENION_ID_LENGTH. Se non viene fornito alcun *dato* , reimposta l' **identificativo di attenzione** sul valore iniziale.

ImqString authenticator () const;
Restituisce una copia del **programma di autenticazione**, riempito con spazi vuoti finali fino alla lunghezza MQ_AUTHENTICATOR_LENGTH.

void setAuthenticator(const char * *dati* = 0);
Imposta il **programma di autenticazione**, riempito con spazi vuoti finali per la lunghezza MQ_AUTHENTICATOR_LENGTH. Se non viene fornito alcun *dato* , reimposta **authenticator** sul valore iniziale.

Codice ImqString bridgeAbend() const;
Restituisce una copia del **codice di abend bridge**, riempito con spazi vuoti finali fino alla lunghezza MQ_ABEND_CODE_LENGTH.

ImqString bridgeCancelCodice () const;
Restituisce una copia del **codice di annullamento bridge**, riempito con spazi vuoti finali fino alla lunghezza MQ_CANCEL_CODE_LENGTH.

void setBridgeCancelCode(const char * *data* = 0);
Imposta il **codice di annullamento bridge**, riempito con spazi finali di lunghezza MQ_CANCEL_CODE_LENGTH. Se non vengono forniti *dati* , reimposta il **codice di annullamento bridge** sul valore iniziale.

Codice MQLONG bridgeCompletion() const;
Restituisce una copia del **codice di completamento bridge**.

Offset bridgeErrorMQLONG () const;
Restituisce una copia dell' **offset di errore bridge**.

Codice MQLONG bridgeReason() const;
Restituisce una copia del **codice motivo del bridge**.

Codice MQLONG bridgeReturn() const;
Restituisce il **codice di ritorno bridge**.

MQLONG conversationalTask() const;
Restituisce una copia dell' **attività di conversazione**.

void setConversationalTask (const MQLONG *task* = MQCCT_NO);
Imposta l' **attività di conversazione**.

MQLONG cursorPosition() const;
Restituisce una copia della **posizione del cursore**.

void setCursorPosizione (const MQLONG *posizione* = 0);
Imposta la **posizione del cursore**.

Tempo facilityKeepMQLONG () const;
Restituisce una copia del **tempo di conservazione funzione**.

void setFacilityKeepTime(const MQLONG *time* = 0);
Imposta il **tempo di conservazione funzione**.

ImqString facilityLike() const;
Restituisce una copia della funzione **come**, riempita con spazi vuoti finali fino alla lunghezza MQ_FACILITY_LIKE_LENGTH.

void setFacilityLike (const char * *nome* = 0);
Imposta la funzione **come**, riempita con spazi vuoti finali per la lunghezza MQ_FACILITY_LIKE_LENGTH. Se non viene fornito alcun *nome* , reimposta la funzione **come** il valore iniziale.

ImqBinary facilityToken() const;
Restituisce una copia del **token funzione**.

ImqBoolean setFacilityToken (const ImqBinary & *token*);
Imposta il **token funzione**. La **lunghezza dati** di *token* deve essere zero o MQ_FACILITY_LENGTH. Restituisce TRUE in caso di esito positivo.

void setFacilityToken (const MQBYTE8 *token* = 0);
Imposta il **token funzione**. *token* può essere zero, che equivale a specificare MQCFAC_NONE. Se *token* è diverso da zero, deve indirizzare i byte MQ_FACILITY_LENGTH dei dati binari. Quando si utilizzano valori predefiniti come MQCFAC_NONE, potrebbe essere necessario eseguire un cast per garantire una corrispondenza della firma. Ad esempio, (MQBYTE *) MQCFAC_NONE.

Funzione ImqString () const;
Restituisce una copia della **funzione**, riempita con spazi vuoti finali fino alla lunghezza MQ_FUNCTION_LENGTH.

Intervallo getWaitMQLONG () const;
Restituisce una copia dell' **intervallo di attesa get**.

void setGetWaitInterval(const MQLONG *interval* = MQCGWI_DEFA
Imposta l' **intervallo di attesa get**.

MQLONG linkType() const;
Restituisce una copia del **tipo di collegamento**.

void setLinkType (const MQLONG *type* = MQCLT_PROGRAM);
Imposta il **tipo di link**.

ImqString nextTransactionIdentifier () const;
Restituisce una copia dei dati **identificativo della transazione successiva** , riempita con spazi finali fino alla lunghezza MQ_TRANSACTION_ID_LENGTH.

MQLONG outputDataLunghezza () const;
Restituisce una copia della **lunghezza dati di output**.

void setOutputDataLength(const MQLONG *length* = MQCODL_AS_INPUT);
Imposta la **lunghezza dei dati di output**.

ImqString replyToFormato () const;
Restituisce una copia del nome del **formato reply - to** , riempito con spazi finali di lunghezza MQ_FORMAT_LENGTH.

void setReplyToFormat(const char * *nome* = 0);
Imposta il **formato reply - to**, riempito con spazi finali di lunghezza MQ_FORMAT_LENGTH. Se non viene fornito alcun *nome* , reimposta il **formato reply - to** sul valore iniziale.

ImqString startCode() const;
Restituisce una copia del **codice di inizio**, riempita con spazi vuoti finali fino alla lunghezza MQ_START_CODE_LENGTH.

void setStartCode (const char + *data* = 0);
Imposta i dati del **codice iniziale** , riempiti con spazi finali di lunghezza MQ_START_CODE_LENGTH. Se non viene fornito alcun *dato* , reimposta il **codice di avvio** sul valore iniziale.

Stato MQLONG taskEnd() const;
Restituisce una copia dello **stato di fine attività**.

ImqString transactionIdentifier() const;
Restituisce una copia dei dati dell' **identificativo della transazione** , riempita con spazi vuoti finali fino alla lunghezza MQ_TRANSACTION_ID_LENGTH.

void setTransactionIdentifier (const char * *data* = 0);
Imposta l' **identificativo della transazione**, riempito con spazi finali di lunghezza MQ_TRANSACTION_ID_LENGTH. Se non vengono forniti *dati* , reimposta **identificativo transazione** sul valore iniziale.

UOWControl MQLONG () const;
Restituisce una copia del **controllo UOW**.

void setUOWControl(const MQLONG *control* = MQCUOWC_ONLY);
Imposta il **controllo UOW**.

MQLONG versione () const;
Restituisce il numero **versione** .

ImqBoolean setVersion(const MQLONG *versione* = MQCIH_VERSION_2);
Imposta il numero di **versione** . Restituisce TRUE in caso di esito positivo.

Dati oggetto (protetti)

MQLONG *olVersion*
Il numero massimo di versione MQCIH che può trovarsi nell'archivio assegnato per *opcih*.

***opcih* PMQCIH**
L'indirizzo di una struttura dati MQCIH. La quantità di memoria assegnata è indicata da *olVersion*.

Codici di origine

- MQRC_BINARY_DATA_LENGTH_ERROR
- VERSIONE MQRC_WRONG_

Codici di ritorno

Tabella 867. Codici di ritorno della classe ImqCICSBridgeHeader

Codice di ritorno	Funzione	CompCode	Motivo	Codice di abend
OK MQCRC				
ERRORE MQCRC_BRIDGE_			MQFB_CICS	
ERRORE MQCR_MQ_API	Nome chiamata IBM MQ	IBM MQ CompCode	IBM MQ Motivo	
MQCRC_BRIDGE_TIMEOUT	Nome chiamata IBM MQ	IBM MQ CompCode	IBM MQ Motivo	
ERRORE MQCRC_CICS_EXEC	EIBFN CICS	CICS EIBRESP	CICS EIBRESP2	
ERRORE MQCR_SECURITY_ERROR	EIBFN CICS	CICS EIBRESP	CICS EIBRESP2	
PROGRAMMA_MQCR_NOT_AVAILABLE	EIBFN CICS	CICS EIBRESP	CICS EIBRESP2	
MQCRC_TRANSID_NOT_AVAILABLE	EIBFN CICS	CICS EIBRESP	CICS EIBRESP2	
MQCRC_BRIDGE_ABEND				CICS CODICE
FINE ANOMALA APPLICAZIONE MQCR				CICS CODICE

Classe C++ ImqDeadLetterHeader

Questa classe incapsula le funzioni della struttura dati MQDLH.

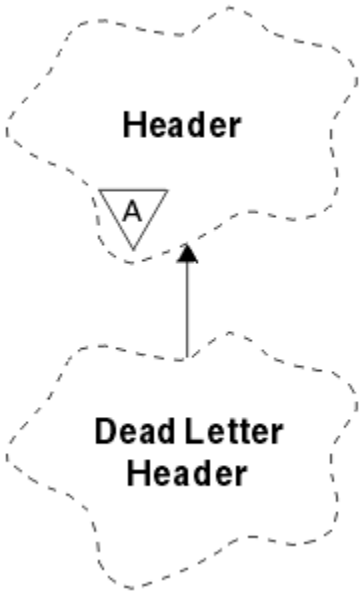


Figura 19. Classe ImqDeadLetterHeader

Gli oggetti di questa classe sono generalmente utilizzati da una applicazione che rileva un messaggio che non può essere elaborato. Un nuovo messaggio che comprende un'intestazione di messaggi non instradabili e il contenuto del messaggio viene inserito nella coda di messaggi non instradabili e il messaggio viene eliminato.

Attributi oggetto

codice motivo lettera non recapitabile
Il motivo per cui il messaggio è arrivato sulla coda di messaggi non instradabili. Il valore iniziale è MQRC_NONE.

Nome gestore code destinazione
Il nome del gestore code di destinazione originale. Il nome è una stringa di lunghezza MQ_Q_MGR_NAME_LENGTH. Il valore iniziale è null.

Nome coda di destinazione
Il nome della coda di destinazione originale. Il nome è una stringa di lunghezza MQ_Q_NAME_LENGTH. Il valore iniziale è null.

Nome applicazione Put
Il nome dell'applicazione che ha inserito il messaggio nella coda di messaggi non instradabili. Il nome è una stringa di lunghezza MQ_PUT_APPL_NAME_LENGTH. Il valore iniziale è null.

Tipo di applicazione Put
Il tipo di applicazione che inserisce il messaggio nella coda di messaggi non recapitabili. Il valore iniziale è zero.

data di collocazione
La data in cui il messaggio è stato inserito nella coda di messaggi non instradabili. La data è una stringa di lunghezza MQ_PUT_DATE_LENGTH. Il valore iniziale è una stringa nulla.

Ora Put
L'ora in cui il messaggio è stato inserito nella coda di messaggi non instradabili. L'ora è una stringa di lunghezza MQ_PUT_TIME_LENGTH. Il valore iniziale è una stringa nulla.

Costruttori

ImqDeadLetterHeader( );
Il costruttore predefinito.

ImqDeadLetterHeader(const ImqDeadLetterHeader & *intestazione*);
Il costruttore di copia.

Metodi ImqItem sovraccaricati

ImqBoolean copyOut virtuale (ImqMessage & *msg*);
Inserisce una struttura dati MQDLH nel buffer dei messaggi all'inizio, spostando ulteriormente i dati dei messaggi esistenti. Imposta il formato *msg* su MQFMT_DEAD_LETTER_HEADER.

Consultare la descrizione del metodo della classe ImqHeader a pagina "Classe ImqHeader C++" a pagina 1877 per ulteriori dettagli.

ImqBoolean pasteIn virtuale (ImqMessage & *msg*);
Legge una struttura dati MQDLH dal buffer di messaggio.

Per avere esito positivo, il formato ImqMessage deve essere MQFMT_DEAD_LETTER_HEADER.

Consultare la descrizione del metodo della classe ImqHeader a pagina “Classe ImqHeader C++” a pagina 1877 per ulteriori dettagli.

Metodi oggetto (pubblico)

operatore vuoto = (const ImqDeadLetterHeader & *intestazione*);
Copia i dati dell'istanza da *intestazione*, sostituendo i dati dell'istanza esistenti.

MQLONG deadLetterReasonCode () const;
Restituisce il codice motivo della lettera non recapitabile.

void setDeadLetterReasonCode (const MQLONG *motivo*);
Imposta il codice di errore della lettera non recapitabile.

ImqString destinationQueueManagerName () const;
Restituisce il nome del gestore code di destinazione, senza spazi finali.

void setDestinationQueueManagerNome (const char * *nome*);
Imposta il nome del gestore code di destinazione. Tronca i dati più lunghi di MQ_Q_MGR_NAME_LENGTH (48 caratteri).

ImqString destinationQueueNome () const;
Restituisce una copia del nome della coda di destinazione, senza spazi vuoti finali.

void setDestinationQueueName (const char * *nome*);
Imposta il nome della coda di destinazione. Tronca i dati più lunghi di MQ_Q_NAME_LENGTH (48 caratteri).

ImqString putApplicationNome () const;
Restituisce una copia del nome dell'applicazione di inserimento, senza spazi vuoti finali.

void setPutApplicationName (const char * *name* = 0);
Imposta il nome dell'applicazione di inserimento. Tronca i dati più lunghi di MQ_PUT_APPL_NAME_LENGTH (28 caratteri).

MQLONG putApplicationTipo () const;
Restituisce il tipo di applicazione put.

void setPutApplicationType (const MQLONG *type* = MQAT_NO_CONTEXT);
Imposta il tipo di applicazione di inserimento.

ImqString putDate () const;
Restituisce una copia della data di inserimento, senza spazi vuoti finali.

void setPutDate (const char * *date* = 0);
Imposta la data di inserimento. Tronca i dati più lunghi di MQ_PUT_DATE_LENGTH (8 caratteri).

ImqString putTime () const;
Restituisce una copia dell'ora di inserimento, senza spazi vuoti finali.

void setPutTime (const char * *ora* = 0);
Imposta l'ora di inserimento. Tronca i dati più lunghi di MQ_PUT_TIME_LENGTH (8 caratteri).

Dati oggetto (protetti)

MQDLH *omqdlh*
La struttura dati MQDLH.

Codici di origine

- MQRC_INCONSISTENT_FORMAT
- ERRORE MQRC_STRUC_ID
- ERRORE MQRC_ENCODING_

Classe ImqDistributionList C++

Questa classe incapsula un elenco di distribuzione dinamico che fa riferimento a una o più code allo scopo di inviare uno o più messaggi a più destinazioni.

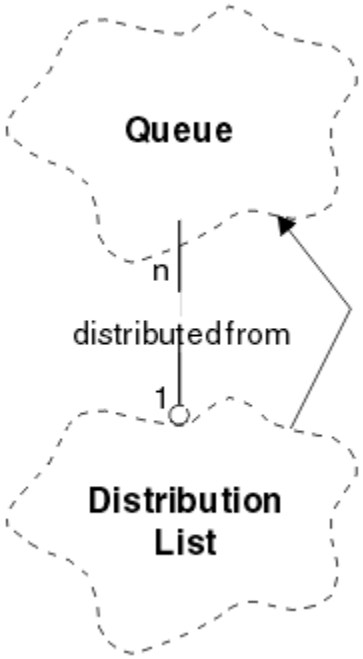


Figura 20. Classe elenco ImqDistribution

Attributi oggetto

prima coda distribuita
Il primo di uno o più oggetti della classe, in nessun ordine particolare, in cui il **riferimento dell'elenco di distribuzione** si riferisce a questo oggetto.

Inizialmente non esistono oggetti di questo tipo. Per aprire correttamente un elenco ImqDistribution, deve esservi almeno un oggetto di questo tipo.

Nota: Quando viene aperto un oggetto Elenco ImqDistribution, tutti gli oggetti aperti che fanno riferimento ad esso vengono automaticamente chiusi.

Costruttori

ImqDistributionList ();
Il costruttore predefinito.

Elenco ImqDistribution(const ImqDistributionList & ***list*** **);**
Il costruttore di copia.

Metodi oggetto (pubblico)

void operator = (const ImqDistributionList & ***list*** **);**
Tutti gli oggetti che fanno riferimento a **questo** oggetto vengono annullati prima della copia. Nessun oggetto farà riferimento a **questo** oggetto dopo il richiamo di questo metodo.

*** firstDistributedQueue () const ;**
Restituisce la **prima coda distribuita**.

Metodi oggetto (protetti)

void setFirstDistributedQueue (* *queue* = 0);
Imposta la **prima coda distribuita**.

Classe ImqError C++

Questa classe astratta fornisce informazioni sugli errori associati ad un oggetto.

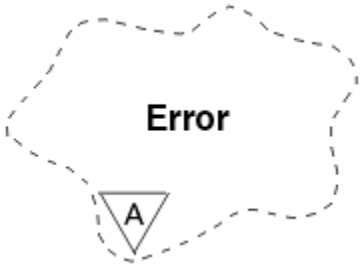


Figura 21. classe ImqError

Attributi oggetto

codice di completamento
Il codice di completamento più recente. Il valore iniziale è zero. Sono possibili i seguenti valori aggiuntivi:

- MQCC_OK
- MQCC_AVVERTENZA
- MQCC_NON RIUSCITO

codice motivo
Il codice di errore più recente. Il valore iniziale è zero.

Costruttori

ImqError( );
Il costruttore predefinito.

ImqError(const ImqError & *errore*);
Il costruttore di copia.

Metodi oggetto (pubblico)

void operatore = (const ImqError & *errore*);
Copia i dati di istanza da *errore*, sostituendo i dati di istanza esistenti.

void clearErrorCodici ();
Imposta il **codice di completamento** e il **codice motivo** entrambi su zero.

MQLONG completionCode () const ;
Restituisce il **codice di completamento**.

MQLONG reasonCode () const ;
Restituisce il **codice motivo**.

Metodi oggetto (protetti)

ImqBoolean checkReadPuntatore (const void * *pointer*, const size_t *length*);
Verifica che la combinazione di puntatore e lunghezza sia valida per l'accesso di sola lettura e restituisce TRUE in caso di esito positivo.

ImqBoolean checkWritePointer (const void * *pointer*, const size_t *length*);
Verifica che la combinazione di puntatore e lunghezza sia valida per l'accesso lettura - scrittura e restituisce TRUE in caso di esito positivo.

void setCompletionCode (const MQLONG *code* = 0);
Imposta il **codice di completamento**.

void setReasonCode (const MQLONG *code* = 0);
Imposta il **codice di errore**.

Codici di origine

- ERRORE MQRC_BUFFER_

Classe C++ ImqGetMessageOptions

Questa classe incapsula la struttura dati MQGMO

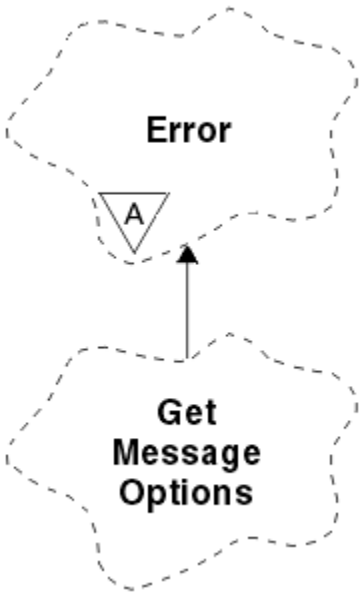


Figura 22. Classe ImqGetMessageOptions

Attributi oggetto

stato del gruppo
Stato di un messaggio per un gruppo di messaggi. Il valore iniziale è MQGS_NOT_IN_GROUP. Sono possibili i seguenti valori aggiuntivi:

- MQGS_MSG_IN_GROUP
- MQGS_LAST_MSG_IN_GROUP

opzioni di corrispondenza
Opzioni per selezionare i messaggi in arrivo. Il valore iniziale è MQMO_MATCH | MQMO_MATCH_CORREL_ID. Sono possibili i seguenti valori aggiuntivi:

- ID_GROUP_MQMO
- NUMERO SEQ MQMO_MATCH_MSG_
- MQMO_MATCH_OFFSET
- MQMO_MSG_TOKEN
- MQMO_NONE

token del messaggio
Token messaggio. Un valore binario (MQBYTE16) di lunghezza MQ_MSG_TOKEN_LENGTH. Il valore iniziale è MQMTOK_NONE.

opzioni
Opzioni applicabili a un messaggio. Il valore iniziale è MQGMO_NO_WAIT. Sono possibili i seguenti valori aggiuntivi:

- MQGMO_WAIT
- SYNCPOINT MQGMO
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_NO_SYNCPOINT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_SUCCESSIVO
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MSG_UNDER_CURSOR
- LOCK_MQGMO
- MQGMO_UNLOCK
- MQGMO_ACCEPT_TRUNCATED_MSG
- MQGMO_SET_SIGNAL
- MQGMO_FAIL_IF_QUIESCING
- MQGMO_CONVERT
- ORDER LOGICAL_MQGMO_
- MQGMO_COMPLETE_MSG
- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_NONE

Nome coda risolto
Nome coda risolto. Questo attributo è di sola lettura. I nomi non sono mai più lunghi di 48 caratteri e possono essere riempiti con valori null. Il valore iniziale è una stringa null.

lunghezza restituita
Lunghezza restituita. Il valore iniziale è MQRL_UNDEFINED. Questo attributo è di sola lettura.

segmentazione
La capacità di segmentare un messaggio. Il valore iniziale è MQSEG_INIITED. È possibile utilizzare il valore aggiuntivo MQSEG_ALLOWED.

stato segmento
Lo stato di segmentazione di un messaggio. Il valore iniziale è MQSS_NOT_A_SEGMENT. Sono possibili i seguenti valori aggiuntivi:

- ISCRIZIONE MQSS_SEGMENT
- ISCRIZIONE MQSS_LAST_SEGMENT

partecipazione punto di sincronizzazione
TRUE quando i messaggi vengono richiamati sotto il controllo del punto di sincronizzazione.

intervallo di attesa
L'intervallo di tempo durante il quale il metodo get della classe viene sospeso durante l'attesa dell'arrivo di un messaggio appropriato, se non è già disponibile. Il valore iniziale è zero, che influisce su un'attesa indefinita. Il valore aggiuntivo, MQWI_UNLIMITED, è possibile. Questo attributo viene ignorato a meno che le opzioni non includano MQGMO_WAIT.

Costruttori

ImqGetMessageOptions( );
Il costruttore predefinito.

ImqGetMessageOptions(const ImqGetMessageOptions & ***gmo*** **);**
Il costruttore di copia.

Metodi oggetto (pubblico)

operatore void = (const ImqGetMessageOptions & ***gmo*** **);**
Copia i dati dell'istanza da *gmo*, sostituendo i dati dell'istanza esistenti.

MQCHAR groupStatus () const;
Restituisce lo stato del gruppo.

void setGroupStatus (const MQCHAR ***status*** **);**
Imposta lo stato del gruppo.

MQLONG matchOptions () const;
Restituisce le opzioni di corrispondenza.

void setMatchOptions (const MQLONG ***options*** **);**
Imposta le opzioni di corrispondenza.

ImqBinary messageToken() const;
Restituisce il token del messaggio.

ImqBoolean setMessageToken (const ImqBinary & ***token*** **);**
Imposta il token del messaggio. La lunghezza dei dati di *token* deve essere zero o MQ_MSG_TOKEN_LENGTH. Questo metodo restituisce TRUE se ha esito positivo.

void setMessageToken (const MQBYTE16 ***token*** **= 0);**
Imposta il token del messaggio. *token* può essere zero, che equivale a specificare MQMTOK_NONE. Se *token* è diverso da zero, allora deve indirizzare i byte MQ_MSG_TOKEN_LENGTH dei dati binari.

Quando si utilizzano valori predefiniti, come ad esempio MQMTOK_NONE, potrebbe non essere necessario eseguire un cast per garantire una corrispondenza della firma, ad esempio (MQBYTE *) MQMTOK_NONE.

Opzioni MQLONG () const;
Restituisce le opzioni.

void setOptions (const MQLONG ***opzioni*** **);**
Imposta le opzioni, incluso il valore di partecipazione del punto di sincronizzazione.

ImqString resolvedQueueNome () const;
Restituisce una copia del nome della coda risolta.

MQLONG returnedLength() const;
Restituisce la lunghezza restituita.

Segmentazione MQCHAR () const;
Restituisce la segmentazione.

void setSegmentation (const MQCHAR *valore*);
Imposta la segmentazione.

MQCHAR segmentStatus () const;
Restituisce lo stato del segmento.

void setSegmentStatus (const MQCHAR *status*);
Imposta lo stato del segmento.

ImqBoolean syncPointPartecipazione () const;
Restituisce il valore di partecipazione del punto di sincronizzazione, che è TRUE se le opzioni includono MQGMO_SYNCPOINT o MQGMO_SYNCPOINT_IF_PERSISTENT.

void setSyncPointParticipation (const ImqBoolean *sync*);
Imposta il valore di partecipazione del punto di sincronizzazione. Se *sync* è TRUE, modifica le opzioni per includere MQGMO_SYNCPOINT e per escludere sia MQGMO_NO_SYNCPOINT che MQGMO_SYNCPOINT_IF_PERSISTENT. Se *sync* è FALSE, modifica le opzioni per includere MQGMO_NO_SYNCPOINT e per escludere sia MQGMO_SYNCPOINT che MQGMO_SYNCPOINT_IF_PERSISTENT.

MQLONG waitInterval () const;
Restituisce l'intervallo di attesa.

void setWaitInterval (const MQLONG *intervallo*);
Imposta l'intervallo di attesa.

Metodi oggetto (protetti)

vuoto statico setVersionsupportato (const MQLONG);
Imposta la versione MQGMO. Il valore predefinito è MQGMO_VERSION_3.

Dati oggetto (protetti)

MQGMO *mqgmo*
Una struttura dati MQGMO Versione 2. Accedere ai campi MQGMO supportati solo per MQGMO_VERSION_2 .

PMQGMO *opgmo*
L'indirizzo di una struttura dati MQGMO. Il numero di versione per questo indirizzo è indicato in *olVersion*. Esaminare il numero di versione prima di accedere ai campi MQGMO, per assicurarsi che siano presenti.

MQLONG *olVersion*
Il numero di versione della struttura dati MQGMO a cui si rivolge *opgmo*.

Codici di origine

- MQRC_BINARY_DATA_LENGTH_ERROR

Classe ImqHeader C++

Questa classe astratta incapsula le funzioni comuni della struttura dati MQDLH.

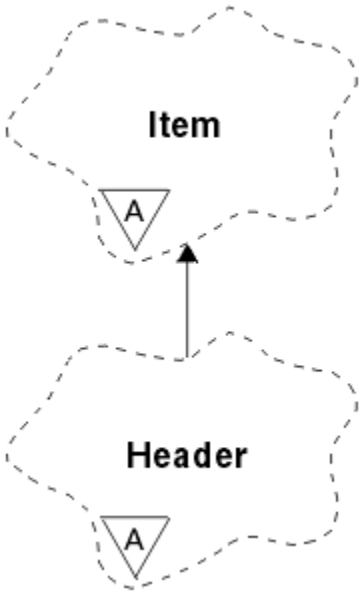


Figura 23. Classe ImqHeader

- "Attributi oggetto" a pagina 1878
- "Costruttori" a pagina 1878
- "Metodi oggetto (pubblico)" a pagina 1878

Attributi oggetto

character set
Il CCSID (coded character set identifier) originale. Inizialmente MQCCSI_Q_MGR.

codifica
La codifica originale. Inizialmente MQENC_NATIVE.

formato
Il formato originale. Inizialmente MQFMT_NONE.

indicatori intestazione
I valori iniziali sono:

- Zero per gli oggetti della classeLetterHeader ImqDead
- MQIIH_NONE per gli oggetti della classe ImqIMSBridgeHeader
- MQRMHF_LAST per gli oggetti della classe di intestazione ImqReference
- MQCIH_NONE per gli oggetti della classe ImqCICSBridgeHeader
- MQWIH_NONE per gli oggetti della classe di intestazione ImqWork

Costruttori

ImqHeader( );
Il costruttore predefinito.

ImqHeader(const ImqHeader & *intestazione*);
Il costruttore di copia.

Metodi oggetto (pubblico)

void operator = (const ImqHeader & *header*);
Copia i dati di istanza da *intestazione*, sostituendo quelli esistenti.

MQLONG virtuale characterSet () const ;
Restituisce la **serie di caratteri**.

vuoto virtuale setCharacterSet (const MQLONG *ccsid* = MQCCSI_Q_MGR);
Imposta la **serie di caratteri**.

MQLONG virtuale codifica () const ;
Restituisce la **codifica**.

virtual void setEncoding (const MQLONG *encoding* = MQENC_NATIVE);
Imposta la **codifica**.

ImqString formato () const ;
Restituisce una copia del **formato**, inclusi gli spazi finali.

setFormat (const char * *name* = 0);
Imposta il **formato**, riempito con 8 caratteri con spazi finali.

MQLONG virtuale headerFlags () const ;
Restituisce gli **indicatori di intestazione**.

virtual void setHeaderIndicatori (const MQLONG *indicatori* = 0);
Imposta gli **indicatori di intestazione**.

Classe ImqIMSBridgeHeader C++

Questa classe incapsula le funzioni della struttura dati MQIIH.

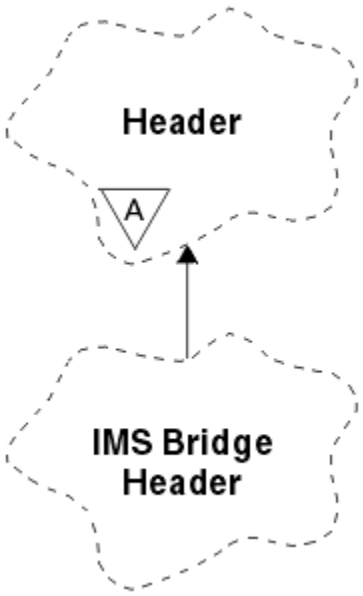


Figura 24. classe ImqIMSBridgeHeader

Gli oggetti di questa classe vengono utilizzati da applicazioni che inviano messaggi al bridge IMS tramite IBM MQ for z/OS.

Nota: La serie di caratteri e la codifica ImqHeader devono avere valori predefiniti e non devono essere impostati su altri valori.

- "Attributi oggetto" a pagina 1880
- "Costruttori" a pagina 1880
- "Metodi ImqItem sovraccaricati" a pagina 1880
- "Metodi oggetto (pubblico)" a pagina 1880
- "Dati oggetto (protetti)" a pagina 1881
- "Codici di origine" a pagina 1881

Attributi oggetto

programma di autenticazione
RACF password o passticket, di lunghezza MQ_AUTHENTICATOR_LENGTH. Il valore iniziale è MQIAUT_NONE.

Modalità di commit
Modalità di commit. Consultare il manuale *OTMA User's Guide* per ulteriori informazioni sulle modalità di commit IMS . Il valore iniziale è MQICM_COMMIT_THEN_SEND. Il valore aggiuntivo, MQICM_SEND_THEN_COMMIT, è possibile.

Sovrascrittura terminale logico
Sovrascrittura terminale logico, di lunghezza MQ_LTERM_OVERRIDE_LENGTH. Il valore iniziale è una stringa null.

Nome associazione di servizi del formato messaggio
Il nome della mappa MFS, di lunghezza MQ_MFS_MAP_NAME_LENGTH. Il valore iniziale è una stringa null.

formato reply - to
Formato di qualsiasi risposta, di lunghezza MQ_FORMAT_LENGTH. Il valore iniziale è MQFMT_NONE.

ambito della sicurezza
Ambito dell'elaborazione della sicurezza IMS . Il valore iniziale è MQISS_CHECK. Il valore aggiuntivo, MQISS_FULL, è possibile.

id istanza transazione
Identità dell'istanza della transazione, un valore binario (MQBYTE16) di lunghezza MQ_TRAN_INSTANCE_ID_LENGTH. Il valore iniziale è MQITII_NONE.

STATO TRANSAZIONE
Stato della conversazione IMS . Il valore iniziale è MQITS_NOT_IN_CONVERSATION. Il valore aggiuntivo, MQITS_IN_CONVERSATION, è possibile.

Costruttori

ImqIMSBridgeHeader();
Il costruttore predefinito.

ImqIMSBridgeHeader(IMSBridgeHeader & ***intestazione const*** **);**
Il costruttore di copia.

Metodi ImqItem sovraccaricati

ImqBoolean copyOut virtuale (ImqMessage & ***msg*** **);**
Inserisce una struttura di dati MQIIH nel buffer di messaggi all'inizio, spostando ulteriormente i dati del messaggio esistenti. Imposta il formato *msg* su MQFMT_IMS.

Consultare la descrizione del metodo della classe parent per ulteriori dettagli.

ImqBoolean pasteIn virtuale (ImqMessage & ***msg*** **);**
Legge una struttura dati MQIIH dal buffer di messaggio.

Per avere esito positivo, la codifica dell'oggetto *msg* deve essere MQENC_NATIVE. Richiamare i messaggi con MQGMO_CONVERT in MQENC_NATIVE.

Per avere esito positivo, il formato di ImqMessage deve essere MQFMT_IMS.

Consultare la descrizione del metodo della classe parent per ulteriori dettagli.

Metodi oggetto (pubblico)

operatore void = (const ImqIMSBridgeHeader & ***intestazione*** **);**
Copia i dati di istanza da *intestazione*, sostituendo quelli esistenti.

ImqString authenticator () const;
Restituisce una copia del programma di autenticazione, riempita con spazi finali fino alla lunghezza MQ_AUTHENTICATOR_LENGTH.

void setAuthenticator (const char * *nome*);
Imposta l'autenticatore.

MQCHAR commitMode () const;
Restituisce la modalità di commit.

void setCommitMode (const MQCHAR *mode*);
Imposta la modalità di commit.

ImqString logicalTerminalOverride () const;
Restituisce una copia della sovrascrittura del terminale logico.

void setLogicalTerminalOverride (const char * *override*);
Imposta la sostituzione del terminale logico.

ImqString messageFormatServicesMapName () const;
Restituisce una copia del nome della mappa dei servizi di formato del messaggio.

void setMessageFormatServicesMapName (const char * *name*);
Imposta il nome della mappa dei servizi di formato del messaggio.

ImqString replyToFormato () const;
Restituisce una copia del formato reply - to, riempita con spazi finali di lunghezza MQ_FORMAT_LENGTH.

void setReplyToFormat (const char * *formato*);
Imposta il formato reply - to, riempito con spazi finali di lunghezza MQ_FORMAT_LENGTH.

MQCHAR securityScope () const;
Restituisce l'ambito di sicurezza.

void setSecurityAmbito (const MQCHAR *ambito*);
Imposta l'ambito di sicurezza.

ID ImqBinary transactionInstance() const;
Restituisce una copia dell'ID istanza della transazione.

ImqBoolean setTransactionInstanceId (const ImqBinary & *id*);
Imposta l'ID istanza della transazione. La lunghezza dei dati di *token* deve essere zero o MQ_TRAN_INSTANCE_ID_LENGTH. Questo metodo restituisce TRUE se ha esito positivo.

void setTransactionInstanceId (const MQBYTE16 *ID* = 0);
Imposta l'ID istanza della transazione. *id* può essere zero, che equivale a specificare MQITII_NONE. Se *id* è diverso da zero, deve indirizzare i byte MQ_TRAN_INSTANCE_ID_LENGTH dei dati binari. Quando si utilizzano valori predefiniti come MQITII_NONE, potrebbe essere necessario eseguire un cast per garantire una corrispondenza di firma, ad esempio (MQBYTE *) MQITII_NONE.

MQCHAR transactionState () const;
Restituisce lo stato della transazione.

void setTransactionState (const MQCHAR *stato*);
Imposta lo stato della transazione.

Dati oggetto (protetti)

MQIIH *omqiih*
La struttura dati MQIIH.

Codici di origine

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_INCONSISTENT_FORMAT
- ERRORE MQRC_ENCODING_

- ERRORE MQRC_STRUC_ID

Classe ImqItem C++

Questa classe astratta rappresenta un elemento, forse uno dei diversi, all'interno di un messaggio.

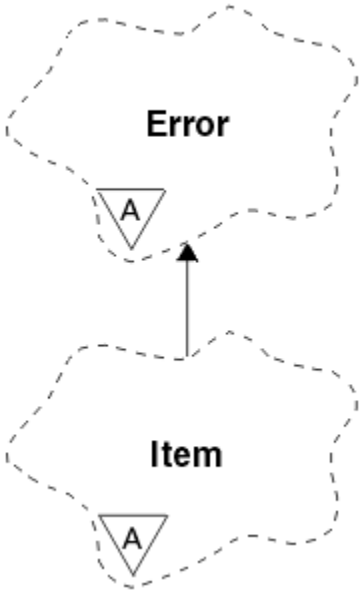


Figura 25. classe ImqItem

Le voci sono concatenate in un buffer di messaggi. Ogni specializzazione è associata a una particolare struttura dati che inizia con un ID struttura.

I metodi polimorfici in questa classe astratta consentono di copiare gli elementi da e verso i messaggi. I metodi ImqMessage class **readItem** e **writeItem** forniscono un altro stile di richiamo di questi metodi polimorfici che è più naturale per i programmi applicativi.

- "Attributi oggetto" a pagina 1882
- "Costruttori" a pagina 1882
- "Metodi di classe (pubblico)" a pagina 1882
- "Metodi oggetto (pubblico)" a pagina 1883
- "Codici di origine" a pagina 1883

Attributi oggetto

id struttura
Una stringa di quattro caratteri all'inizio della struttura dati. Questo attributo è di sola lettura. Considerare questo attributo per le classi derivate. Non è incluso automaticamente.

Costruttori

ImqItem( );
Il costruttore predefinito.

ImqItem(const ImqItem & ***item*** **);**
Il costruttore di copia.

Metodi di classe (pubblico)

static ImqBoolean structureIdIs (const char * ***structure - id - to - test*****, const ImqMessage &** ***msg*** **);**
Restituisce TRUE se l' **ID struttura** del successivo ImqItem nel *msg* in entrata è uguale a *structure - id - to - test*. La voce successiva viene identificata come quella parte del buffer di messaggi attualmente

indirizzata dal **puntatore dati** ImqCache . Questo metodo si basa sull' **id struttura** e, pertanto, non è garantito che funzioni per tutte le classi derivate ImqItem .

Metodi oggetto (pubblico)

void operator = (const ImqItem & ***item*** **);**
Copia i dati di istanza da *elemento*, sostituendo i dati di istanza esistenti.

ImqBoolean copyOut virtuale (ImqMessage & ***msg*** **) = 0;**
Scrive questo oggetto come elemento successivo in un buffer di messaggi in uscita, accodandolo a qualsiasi elemento esistente. Se l'operazione di scrittura ha esito positivo, aumenta la **lunghezza dati** ImqCache . Questo metodo restituisce TRUE se ha esito positivo.

Sovrascrivere questo metodo per gestire una sottoclasse specifica.

ImqBoolean pasteIn (ImqMessage & ***msg*** **) virtuale = 0;**
Legge questo oggetto *in modo distruttivo* dal buffer dei messaggi in entrata. La lettura è distruttiva in quanto il **puntatore dati** di ImqCache viene spostato. Tuttavia, il contenuto del buffer rimane lo stesso, quindi i dati possono essere riletti reimpostando il **puntatore dati** ImqCache .

La classe (secondaria) di questo oggetto deve essere congruente con l' **id struttura** trovato successivamente nel buffer di messaggi dell'oggetto *msg* .

La **codifica** dell'oggetto *msg* deve essere MQENC_NATIVE. Si consiglia di recuperare i messaggi con la **codifica** ImqMessage impostata su MQENC_NATIVE e con le opzioni ImqGetMessageOptions incluso MQGMO_CONVERT.

Se l'operazione di lettura ha esito positivo, la **lunghezza dati** di ImqCache viene ridotta. Questo metodo restituisce TRUE se ha esito positivo.

Sovrascrivere questo metodo per gestire una sottoclasse specifica.

Codici di origine

- ERRORE MQRC_ENCODING_
- ERRORE MQRC_STRUC_ID
- MQRC_INCONSISTENT_FORMAT
- MQRC_BUFFER insufficiente
- DATI MQRC_INSUFFICIENT_

classe C++ ImqMessage

Questa classe incapsula una struttura di dati MQMD e gestisce anche la costruzione e la ricostruzione dei dati del messaggio.

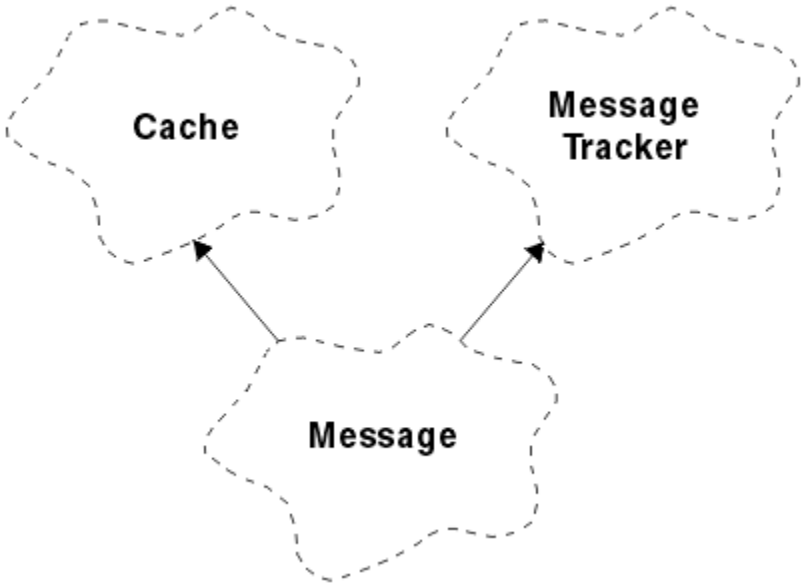


Figura 26. classe ImqMessage

Attributi oggetto

Dati ID applicazione
Informazioni sull'identità associate a un messaggio. Il valore iniziale è una stringa null.

Dati di origine dell'applicazione
Informazioni di origine associate a un messaggio. Il valore iniziale è una stringa null.

conteggio backout
Il numero di volte in cui un messaggio è stato richiamato provvisoriamente e successivamente ripristinato. Il valore iniziale è zero. Questo attributo è di sola lettura.

character set
CCSID (Coded Character Set Id). Il valore iniziale è MQCCSI_Q_MGR. Sono possibili i seguenti valori aggiuntivi:

- MQCCSI_INHERIT
- MQCCSI_EMBEDDED

È anche possibile utilizzare un CCSID (Coded Character Set Id) a scelta. Per informazioni, consultare "Conversione code page" a pagina 965.

codifica
La codifica macchina dei dati del messaggio. Il valore iniziale è MQENC_NATIVE.

scadenza
Una quantità dipendente dal tempo che controlla per quanto tempo IBM MQ conserva un messaggio non richiamato prima di eliminarlo. Il valore iniziale è MQEI_UNLIMITED.

formato
Il nome del formato (modello) che descrive il layout dei dati nel buffer. I nomi più lunghi di otto caratteri vengono troncati a otto. I nomi vengono sempre riempiti con spazi vuoti fino a otto caratteri. Il valore della costante iniziale è MQFMT_NONE. Sono possibili le seguenti costanti aggiuntive:

- MMQFMT_ADMIN
- MQFMT_CICS
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- INTESTAZIONE_DIST_MQFM
- EVENTO MQFMT
- IMS MQFMT
- MQFMT_IMS_VAR_STRING
- MQFMT_MD_EXTENSIONE
- MQFMT_PCF
- MQFMT_REF_MSG_HEADER
- MQFMT_RF_HEADER
- MQFMT_STRING
- MQFMT_TRIGGER
- Intestazione MQFMT_WORK_INFO_HEADER
- MQFMT_XMIT_Q_HEADER

È anche possibile utilizzare una stringa specifica dell'applicazione a scelta. Per ulteriori informazioni, consultare il campo "Formato (MQCHAR8) per MQMD" a pagina 458 del descrittore del messaggio (MQMD).

Indicatori di messaggio
Informazioni sul controllo della segmentazione. Il valore iniziale è MQMF_SEGMENTATION_INIITED. Sono possibili i seguenti valori aggiuntivi:

- MQMF_SEGMENTAZIONE_CONSENTITA
- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- ISCRIZIONE MQMF_SE
- MQMF_LAST_SEGMENT
- MQMF_NONE

tipo di messaggio
L'ampia categorizzazione di un messaggio. Il valore iniziale è MQMT_DATAGRAM. Sono possibili i seguenti valori aggiuntivi:

- MQMT_SYSTEM_FIRST
- SYSTEM_MQMT_LAST
- MQMT_DATAGRAM
- MQMT_REQUEST
- MQMT_REPLY
- REPORT MQMT
- MQMT_APPL_FIRST
- APPL_MQMT_LAST

È anche possibile utilizzare un valore specifico dell'applicazione a scelta. Per ulteriori informazioni, consultare il campo "MsgType (MQLONG) per MQMD" a pagina 447 del descrittore del messaggio (MQMD).

scostamento
Informazioni sull'offset. Il valore iniziale è zero.

Lunghezza originale
La lunghezza originale di un messaggio segmentato. Il valore iniziale è MQOL_UNDEFINED.

persistenza
Indica che il messaggio è importante e deve essere sempre sottoposto a backup utilizzando la memoria persistente. Questa opzione implica una penalizzazione delle prestazioni. Il valore iniziale è MQPER_PERSISTENCE_AS_Q_DEF. Sono possibili i seguenti valori aggiuntivi:

- PERSISTORA_MQPER_
- MQPER_NOT_PERSISTENT

priorità
La priorità relativa per la trasmissione e la consegna. I messaggi con la stessa priorità vengono solitamente consegnati nella stessa sequenza in cui sono stati forniti (anche se ci sono diversi criteri che devono essere soddisfatti per garantire ciò). Il valore iniziale è MQPRI_PRIORITY_AS_Q_DEF.

convalida proprietà
Specifica se la convalida delle proprietà deve essere eseguita quando viene impostata una proprietà del messaggio. Il valore iniziale è **MQCMHO_DEFAULT_VALIDATION**. Sono possibili i seguenti valori aggiuntivi:

- VALIDATE MQCMHO_
- MQCMHO_NO_VALIDATION

I seguenti metodi agiscono sulla **convalida proprietà**:

MQLONG propertyValidation() const;
Restituisce l'opzione **convalida proprietà** .

void setPropertyValidation (const MQLONG *opzione*);
Imposta l'opzione **convalida proprietà** .

Nome applicazione Put
Il nome dell'applicazione che ha inserito un messaggio. Il valore iniziale è una stringa null.

Tipo di applicazione Put
Il tipo di applicazione che inserisce un messaggio. Il valore iniziale è MQAT_NO_CONTEXT. Sono possibili i seguenti valori aggiuntivi:

- AIX MQAT
- MQAT_CICS
- BRIDGE - MQAT_CICS_BRIDGE
- DOS MQAT
- IMS MQAT
- MQAT_IM_Bridge
- MVS MQAT
- MQAT_NOTES_AGENT
- MQAT_OS2
- MQAT_OS390
- MQAT_OS400
- Gestore code MQAT
- UNIX MQAT
- WINDOWS MQAT
- MQAT_WINDOWS_NT
- XCF MQAT
- MQAT_PREDEFINITO
- MQAT_SCONOSCIUTO

- MQAT_USER_FIRST
- MQAT_USER_LAST

È anche possibile utilizzare una stringa specifica dell'applicazione a scelta. Per ulteriori informazioni, consultare il campo “PutApplTipo (MQLONG) per MQMD” a pagina 473 del descrittore del messaggio (MQMD).

data di collocazione
La data in cui è stato inserito un messaggio. Il valore iniziale è una stringa null.

Ora Put
L'ora in cui è stato inserito un messaggio. Il valore iniziale è una stringa null.

nome gestore code reply - to
Il nome del gestore code a cui deve essere inviata qualsiasi risposta. Il valore iniziale è una stringa null.

Nome delle repliche alla coda
Il nome della coda a cui deve essere inviata qualsiasi risposta. Il valore iniziale è una stringa null.

report
Informazioni di feedback associate a un messaggio. Il valore iniziale è MQRO_NONE. Sono possibili i seguenti valori aggiuntivi:

- MQRO_ECCEZIONE
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA *
- SCADENZA_MQRO
- MQRO_EXPIRATION_WITH_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA *
- COA MQRO
- DATA_COA_WITH_MQRO
- MQRO_COA_WITH_FULL_DATA *
- COD MQRO
- DATI MQRO_COD_WITH_
- MQRO_COD_WITH_FULL_DATA *
- MQRO_PAN
- MQRO_NAN
- ID_MSG_NEW_MQRO
- ID_CORREL_NEW_MQRO
- ID_COPY_MQRO_MSG_TO_CORREL_ID
- ID CORREL_PASS_MQRO_
- MQRO_DEAD_LETTER_Q
- MQRO_DISCARD_MSG

dove * indica valori non supportati in IBM MQ for z/OS.

numero di sequenza
Informazioni di sequenza che identificano un messaggio all'interno di un gruppo. Il valore iniziale è uno.

lunghezza totale messaggio
Il numero di byte disponibili durante il tentativo più recente di leggere un messaggio. Questo numero sarà maggiore della **lunghezza del messaggio** di ImqCache se l'ultimo messaggio è stato troncato o se l'ultimo messaggio non è stato letto perché si sarebbe verificato un troncamento. Questo attributo è di sola lettura. Il valore iniziale è zero.

Questo attributo può essere utile in qualsiasi situazione che coinvolga messaggi troncati.

identificativo utente
Un'identità utente associata a un messaggio. Il valore iniziale è una stringa null.

Costruttori

ImqMessage( );
Il costruttore predefinito.

ImqMessage(const ImqMessage & ***msg*** **);**
Il costruttore di copia. Per i dettagli, consultare il metodo **operator =** .

Metodi oggetto (pubblico)

void operator = (const ImqMessage e ***msg*** **);**
Copia i dati MQMD e del messaggio da *msg*. Se un buffer è stato fornito dall'utente per questo oggetto, la quantità di dati copiati è limitata alla dimensione buffer disponibile. In caso contrario, il sistema garantisce che venga reso disponibile un buffer di dimensione adeguata per i dati copiati.

ImqString applicationIdDati () const ;
Restituisce una copia dei **dati ID applicazione**.

void setApplicationIdData (const char * ***data*** **= 0);**
Imposta i **dati ID applicazione**.

ImqString applicationOriginDati () const ;
Restituisce una copia dei **dati di origine dell'applicazione**.

void setApplicationOriginData (const char * ***data*** **= 0);**
Imposta i **dati di origine dell'applicazione**.

MQLONG backoutCount () const ;
Restituisce il **conteggio backout**.

MQLONG characterSet () const ;
Restituisce la **serie di caratteri**.

void setCharacterSet (const MQLONG ***ccsid*** **= MQCCSI_Q_MGR);**
Imposta la **serie di caratteri**.

MQLONG codifica () const ;
Restituisce la **codifica**.

void setEncoding (const MQLONG ***encoding*** **= MQENC_NATIVE);**
Imposta la **codifica**.

MQLONG scadenza () const ;
Restituisce la **scadenza**.

void setExpiry (const MQLONG ***scadenza*** **);**
Imposta la **scadenza**.

ImqString formato () const ;
Restituisce una copia del **formato**, inclusi gli spazi finali.

ImqBoolean formatIs (const char * ***format - to - test*** **) const ;**
Restituisce TRUE se il **formato** è uguale a *format - to - test*.

void setFormat (const char + ***nome*** **= 0);**
Imposta il **formato**, riempito con otto caratteri con spazi finali.

MQLONG messageFlags () const ;
Restituisce gli **indicatori di messaggio**.

void setMessageFlags (const MQLONG ***flags*** **);**
Imposta gli **indicatori di messaggio**.

MQLONG messageType () const ;
Restituisce il **tipo di messaggio**.

void setMessageType (const MQLONG *type*);
Imposta il **tipo di messaggio**.

MQLONG offset () const ;
Restituisce l' **offset**.

void setOffset (const MQLONG *offset*);
Imposta l' **offset**.

MQLONG originalLength () const ;
Restituisce la **lunghezza originale**.

void setOriginalLength (const MQLONG *lunghezza*);
Imposta la **lunghezza originale**.

MQLONG persistenza () const ;
Restituisce la **persistenza**.

void setPersistence (const MQLONG *persistenza*);
Imposta la **persistenza**.

MQLONG priorità () const ;
Restituisce la **priorità**.

void setPriority (const MQLONG *priority*);
Imposta la **priorità**.

ImqString putApplicationNome () const ;
Restituisce una copia del **nome applicazione di inserimento**.

void setPutApplicationName (const char * *nome* = 0);
Imposta il **nome applicazione di inserimento**.

MQLONG putApplicationTipo () const ;
Restituisce il **tipo di applicazione put**.

void setPutApplicationType (const MQLONG *type* = MQAT_NO_CONTEXT);
Imposta il **tipo di applicazione put**.

ImqString putDate () const ;
Restituisce una copia della **data di inserimento**.

void setPutDate (const char + *date* = 0);
Imposta la **data di inserimento**.

ImqString putTime () const ;
Restituisce una copia dell' **ora di inserimento**.

void setPutTime (const char * *time* = 0);
Imposta l' **ora di inserimento**.

ImqBoolean readItem (ImqItem & *item*);
Legge nell'oggetto *item* dal buffer di messaggi, utilizzando il metodo ImqItem **pasteIn** . Restituisce TRUE in caso di esito positivo.

ImqString replyToQueueManagerNome () const ;
Restituisce una copia del **nome gestore code reply - to**.

void setReplyToQueueManagerName (const char * *name* = 0);
Imposta il **nome del gestore code di risposta**.

ImqString replyToQueueName () const ;
Restituisce una copia del **nome della coda di risposta**.

void setReplyToQueueNome (const char + *name* = 0);
Imposta il **nome della coda di risposta**.

MQLONG report () const ;
Restituisce il **prospetto**.

void setReport (const MQLONG *report*);
Imposta il **report**.

MQLONG sequenceNumber () const ;
Restituisce il **numero di sequenza**.

void setSequenceNumber (const MQLONG *numero*);
Imposta il **numero di sequenza**.

size_t totalMessageLunghezza () const ;
Restituisce la **lunghezza totale del messaggio**.

ImqString userId () const ;
Restituisce una copia dell' **ID utente**.

void setUserID (const char * *id* = 0);
Imposta l' **id utente**.

ImqBoolean writeItem (ImqItem e *item*);
Scrive dall'oggetto *item* nel buffer di messaggi, utilizzando il metodo ImqItem **copyOut** . La scrittura può assumere la forma di inserimento, sostituzione o aggiunta: ciò dipende dalla classe dell'oggetto *item* . Questo metodo restituisce TRUE se ha esito positivo.

Metodi oggetto (protetti)

static void setVersionSupported (const MQLONG);
Imposta la **versione MQMD**. Il valore predefinito è **MQMD_VERSION_2**.

Dati oggetto (protetti)

z/OS **MQMD1 *omqmd***
La struttura dei dati MQMD su z/OS.

Multi **MQMD2 *idomq***
La struttura dei dati MQMD su Multipiattaforme.

Classe ImqMessageTracker C++

Questa classe incapsula gli attributi di un oggetto ImqMessage o ImqQueue che possono essere associati a entrambi gli oggetti.

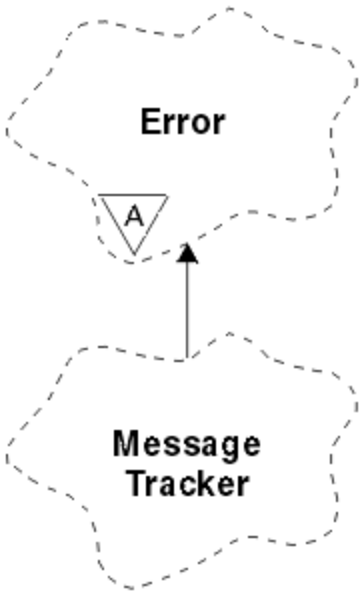


Figura 27. Classe Tracker ImqMessage

Questa classe è relativa alle chiamate MQI elencate in “Riferimento incrociato ImqMessageTracker” a pagina 1841.

Attributi oggetto

token di accounting
Un valore binario (MQBYTE32) di lunghezza MQ_ACCOUNTING_TOKEN_LENGTH. Il valore iniziale è MQACT_NONE.

ID correlazione
Un valore binario (MQBYTE24) di lunghezza MQ_CORREL_ID_LENGTH assegnato per correlare i messaggi. Il valore iniziale è MQCI_NONE. Il valore aggiuntivo, MQCI_NEW_SESSION, è possibile.

il feedback
Informazioni di feedback da inviare con un messaggio. Il valore iniziale è MQFB_NONE. Sono possibili i seguenti valori aggiuntivi:

- MQFB_SYSTEM_FIRST
- MQFB_SYSTEM_LAST
- MQFB_APPL_FIRST
- LAST APPL_MQFB
- COA MQFB
- COD MQFB
- SCADENZA_MQFB
- PAN MQFB
- NAN_MQFB
- MQFB_QUIT
- LENGTH_ZERO MQFB_DATA_
- MQFB_DATA_LENGTH_NEGATIVE
- LENGTH_TOO_BIG MQFB_DATA_BIG
- MQFB_BUFFER_OVERFLOW
- MQFB_LENGTH_OFF_BY_ONE
- ERRORE MQFB_IIH
- MQFB_NOT_AUTHORIZED_FOR_IMS
- ERRORE MQFB_IMS
- MQFB_IMS_FIRST
- MQFB_IMS_LAST
- MQFB_CICS_APPL_ABENDED
- MQFB_CICS_APPL_NOT_STARTED
- MQFB_CICS_BRIDGE_FAILURE
- ERRORE CCSID MQFB_CICS_
- ERRORE MQFB_CICS_CIH
- ERRORE MQFB_CICS_COMMAREA_
- ERRORE MQFB_CICS_CORREL_ID_
- ERRORE MQFB_CICS_DLQ
- ERRORE MQFB_CICS_ENCODING_ERROR
- ERRORE INTERNO MQFB_CICS_

- MQFB_CICS_NOT_AUTHORIZED
- MQFB_CICS_UOW_BACKED_OUT
- ERRORE MQFB_CICS_UOW_

È anche possibile utilizzare una stringa specifica dell'applicazione a scelta. Per ulteriori informazioni, consultare il campo "Feedback (MQLONG) per MQMD" a pagina 452 del descrittore del messaggio (MQMD).

ID gruppo
Un valore binario (MQBYTE24) di lunghezza MQ_GROUP_ID_LENGTH univoco in una coda. Il valore iniziale è MQGI_NONE.

ID messaggio
Un valore binario (MQBYTE24) di lunghezza MQ_MSG_ID_LENGTH univoco all'interno di una coda. Il valore iniziale è MQMI_NONE.

Costruttori

ImqMessageTracker ();
Il costruttore predefinito.

ImqMessageTracker (const ImqMessageTracker & *tracker*);
Il costruttore di copia. Per i dettagli, consultare il metodo **operator =** .

Metodi oggetto (pubblico)

void operator = (const ImqMessageTracker & *tracker*);
Copia i dati di istanza da *tracker*, sostituendo i dati di istanza esistenti.

ImqBinary accountingToken () const ;
Restituisce una copia del **token di account**.

ImqBoolean setAccountingToken (const ImqBinary & *token*);
Imposta il **token di account**. La **lunghezza dei dati** di *token* deve essere zero o MQ_ACCOUNTING_TOKEN_LENGTH. Questo metodo restituisce TRUE se ha esito positivo.

void setAccountingToken (const MQBYTE32 *token* = 0);
Imposta il **token di account**. *token* può essere zero, che equivale a specificare MQACT_NONE. Se *token* è diverso da zero, deve indirizzare i byte MQ_ACCOUNTING_TOKEN_LENGTH dei dati binari. Quando si utilizzano valori predefiniti come MQACT_NONE, potrebbe essere necessario eseguire un cast per garantire una corrispondenza della firma; ad esempio, (MQBYTE *) MQACT_NONE.

ImqBinary correlationId () const ;
Restituisce una copia dell' **ID correlazione**.

ImqBoolean setCorrelationId (const ImqBinary & *token*);
Imposta l' **ID correlazione**. La **lunghezza dati** di *token* deve essere zero o MQ_CORREL_ID_LENGTH. Questo metodo restituisce TRUE se ha esito positivo.

void setCorrelationID (const MQBYTE24 *id* = 0);
Imposta l' **ID correlazione**. *id* può essere zero, che equivale a specificare MQCI_NONE. Se *id* è diverso da zero, deve indirizzare i byte MQ_CORREL_ID_LENGTH dei dati binari. Quando si utilizzano valori predefiniti come MQCI_NONE, potrebbe essere necessario eseguire un cast per garantire una corrispondenza della firma; ad esempio, (MQBYTE *) MQCI_NONE.

feedback MQLONG () const ;
Restituisce il **feedback**.

void setFeedback (const MQLONG *feedback*);
Imposta il **feedback**.

ImqBinary groupId () const ;
Restituisce una copia dell' **ID gruppo**.

ImqBoolean setGroupId (const ImqBinary & *token*);
Imposta l' **id gruppo**. La **lunghezza dati** di *token* deve essere zero o MQ_GROUP_ID_LENGTH. Questo metodo restituisce TRUE se ha esito positivo.

void setGroupId (const MQBYTE24 *id* = 0);
Imposta l' **id gruppo**. *id* può essere zero, che equivale a specificare MQGI_NONE. Se *id* è diverso da zero, deve indirizzare i byte MQ_GROUP_ID_LENGTH dei dati binari. Quando si utilizzano valori predefiniti come MQGI_NONE, potrebbe essere necessario eseguire un cast per garantire una corrispondenza della firma, ad esempio (MQBYTE *) MQGI_NONE.

ImqBinary messageId () const ;
Restituisce una copia di **id messaggio**.

ImqBoolean setMessageId (const ImqBinary e *token*);
Imposta l' **ID messaggio**. La **lunghezza dati** di *token* deve essere zero o MQ_MSG_ID_LENGTH. Questo metodo restituisce TRUE se ha esito positivo.

void setMessageId (const MQBYTE24 *id* = 0);
Imposta l' **ID messaggio**. *id* può essere zero, che equivale a specificare MQMI_NONE. Se *id* è diverso da zero, deve indirizzare i byte MQ_MSG_ID_LENGTH dei dati binari. Quando si utilizzano valori predefiniti come MQMI_NONE, potrebbe essere necessario eseguire un cast per garantire una corrispondenza di firma, ad esempio (MQBYTE *) MQMI_NONE.

Codici di origine

- MQRC_BINARY_DATA_LENGTH_ERROR

Classe C++ ImqNamelist

Questa classe incapsula un elenco nomi.

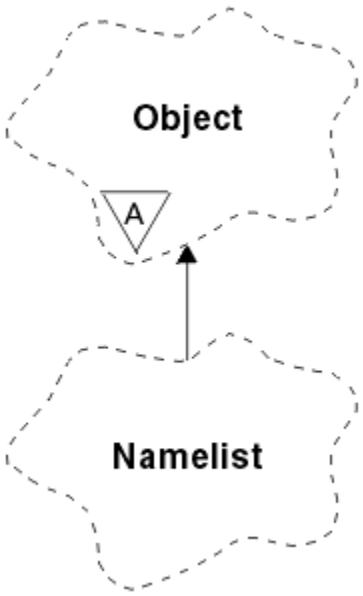


Figura 28. Classe ImqNamelist

Questa classe è relativa alle chiamate MQI elencate in “Riferimento incrociato ImqNamelist” a pagina 1841.

- “Attributi oggetto” a pagina 1894
- “Costruttori” a pagina 1894
- “Metodi oggetto (pubblico)” a pagina 1894
- “Codici di origine” a pagina 1894

Attributi oggetto

Numero nomi
Il numero di nomi oggetto in **nomi elenco nomi**. Questo attributo è di sola lettura.

nomi elenco nomi
I nomi oggetto, il cui numero è indicato dal **conteggio nomi**. Questo attributo è di sola lettura.

Costruttori

ImqNamelist( );
Il costruttore predefinito.

ImqNamelist(const ImqNamelist & *elenco*);
Il costruttore di copia. Lo **stato di apertura** di ImqObject è false.

ImqNamelist(const char * *nome*);
Imposta il nome ImqObject su **name**.

Metodi oggetto (pubblico)

void operator = (const ImqNamelist & *elenco*);
Copia i dati istanza da *elenco*, sostituendo i dati istanza esistenti. Lo **stato di apertura** di ImqObject è false.

ImqBoolean nameCount(MQLONG & *conteggio*);
Fornisce una copia del **conteggio nomi**. Restituisce TRUE in caso di esito positivo.

MQLONG nameCount ();
Restituisce il **conteggio nomi** senza alcuna indicazione di possibili errori.

ImqBoolean namelistName (const MQLONG *index*, ImqString & *name*);
Fornisce una copia di uno dei **nomi elenco nomi** in base all'indice basato su zero. Restituisce TRUE in caso di esito positivo.

ImqString namelistName (const MQLONG *indice*);
Restituisce uno dei **nomi elenco nomi** per indice basato su zero senza alcuna indicazione di possibili errori.

Codici di origine

- ERRORE MQRC_INDEX
- MQRC_INDEX_NOT_PRESENT

Classe ImqObject C++

Questa classe è astratta. Quando un oggetto di questa classe viene eliminato, viene chiuso automaticamente e la connessione del gestore ImqQueueviene interrotta.

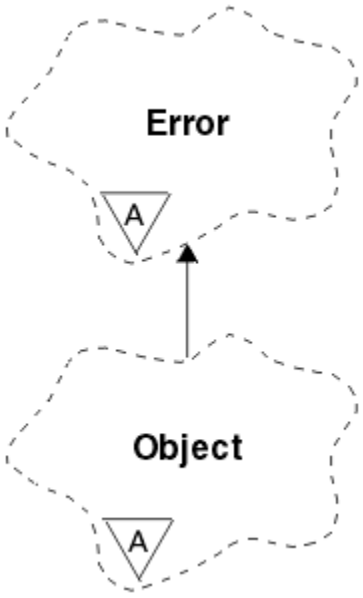


Figura 29. classe ImqObject

Questa classe è relativa alle chiamate MQI elencate in "Riferimento incrociato ImqObject" a pagina 1841.

Attributi classe

comportamento
Controlla il funzionamento dell'apertura implicita.

IMQ_IMPL_OPEN (8L)
L'apertura implicita è consentita. Questa è l'opzione predefinita.

Attributi oggetto

Data della modifica
La data di modifica. Questo attributo è di sola lettura.

Ora della modifica
Il tempo di alterazione. Questo attributo è di sola lettura.

ID utente alternativo
L'ID utente alternativo, con un massimo di MQ_USER_ID_LENGTH caratteri. Il valore iniziale è una stringa null.

ID di sicurezza alternativo
L'ID di sicurezza alternativo. Un valore binario (MQBYTE40) di lunghezza MQ_SECURITY_ID_LENGTH. Il valore iniziale è MQSID_NONE.

opzioni di chiusura
Opzioni che si applicano quando un oggetto viene chiuso. Il valore iniziale è MQCO_NONE. Questo attributo viene ignorato durante le operazioni di riapertura implicite, dove viene utilizzato sempre un valore di MQCO_NONE.

riferimento connessione
Un riferimento a un oggetto gestore ImqQueueche fornisce la connessione richiesta a un gestore code (locale). Per un oggetto Gestore ImqQueue, è l'oggetto stesso. Il valore iniziale è zero.

Nota: Non confondere questo nome con il nome del gestore code che identifica un gestore code (possibilmente remoto) per una coda denominata.

descrizione
Il nome descrittivo (fino a 64 caratteri) del gestore code, della coda, dell'elenco nomi o del processo. Questo attributo è di sola lettura.

nome
Il nome (fino a 48 caratteri) del gestore code, della coda, dell'elenco nomi o del processo. Il valore iniziale è una stringa null. Il nome di una coda modello cambia dopo un **open** nel nome della coda dinamica risultante.

Nota: Un gestore ImqQueuepuò avere un nome null, che rappresenta il gestore code predefinito. Il nome cambia nel gestore code effettivo dopo una corretta apertura. Un elenco ImqDistributionè dinamico e deve avere un nome null.

successivo oggetto gestito
Questo è l'oggetto successivo di questa classe, in nessun ordine particolare, con lo stesso riferimento di connessione di questo oggetto. Il valore iniziale è zero.

Opzioni visualizzate
Le opzioni che vengono applicate quando un oggetto viene aperto. Il valore iniziale è MQOO_INQUIRE. Esistono due modi per impostare i valori appropriati:

1. Non impostare le opzioni di apertura e non utilizzare il metodo di apertura. IBM MQ regola automaticamente le opzioni di apertura e apre, riapre e chiude automaticamente gli oggetti come richiesto. Ciò può comportare operazioni di riapertura non necessarie, perché IBM MQ utilizza il metodo openFor e aggiunge le opzioni di apertura solo in modo incrementale.
2. Impostare le opzioni di apertura prima di utilizzare i metodi che risultano in una chiamata MQI (consultare “Riferimento incrociato C++ e MQI” a pagina 1834). Ciò garantisce che non si verifichino operazioni di riapertura non necessarie. Impostare esplicitamente le opzioni di apertura se è probabile che si verifichi uno dei potenziali problemi di riapertura (consultare Riapri).

   Se si utilizza il metodo di apertura, è necessario prima assicurarsi che le opzioni di apertura siano appropriate. Tuttavia, l'uso del metodo aperto non è obbligatorio; IBM MQ presenta ancora lo stesso comportamento del caso 1, ma in questa circostanza, il comportamento è efficiente.

Zero non è un valore valido; impostare il valore appropriato prima di tentare di aprire l'oggetto. Questa operazione può essere eseguita utilizzando **`setOpenOptions`** (*lOpenOptions*) seguito da **open** () o **`openFor`** (*lRequiredOpenOption*).

Nota:

1. MQOO_OUTPUT viene sostituito con MQOO_INQUIRE durante il metodo **open** per un elenco di distribuzione, poiché MQOO_OUTPUT è l'unica **`open option`** valida in questo momento. Tuttavia, è consigliabile impostare sempre MQOO_OUTPUT esplicitamente nei programmi applicativi che utilizzano il metodo **open** .
2. Specificare MQOO_RESOLVE_NAMES se si desidera utilizzare gli attributi **`resolved queue manager name`** e **`resolved queue name`** della classe.

stato aperto
Se l'oggetto è aperto (TRUE) o chiuso (FALSE). Il valore iniziale è FALSE. Questo attributo è di sola lettura.

oggetto gestito precedente
L'oggetto precedente di questa classe, in nessun ordine particolare, che ha lo stesso riferimento di connessione di questo oggetto. Il valore iniziale è zero.

identificativo gestore code
L'identificativo del gestore code. Questo attributo è di sola lettura.

Costruttori

ImqObject( );
Il costruttore predefinito.

ImqObject(const ImqObject & ***oggetto*** **);**
Il costruttore di copia. Lo stato di apertura sarà FALSE.

Metodi di classe (pubblico)

comportamento statico MQLONG ();
Restituisce il comportamento.

void setBehavior(const MQLONG ***comportamento*** **= 0);**
Imposta il comportamento.

Metodi oggetto (pubblico)

operatore void = (const ImqObject & ***object*** **);**
Esegue una chiusura se necessario e copia i dati dell'istanza da *oggetto*. Lo stato di apertura sarà FALSE.

ImqBoolean alterationDate(ImqString & ***data*** **);**
Fornisce una copia della data di modifica. Restituisce TRUE in caso di esito positivo.

ImqString alterationDate( );
Restituisce la data di modifica senza alcuna indicazione di possibili errori.

ImqBoolean alterationTime(ImqString & ***ora*** **);**
Fornisce una copia del tempo di modifica. Restituisce TRUE in caso di esito positivo.

ImqString alterationTime( );
Restituisce il tempo di modifica senza alcuna indicazione di possibili errori.

ImqString alternateUserId () const;
Restituisce una copia dell'ID utente alternativo.

ImqBoolean setAlternateUserId (const char * ***id*** **);**
Imposta l'ID utente alternativo. l'ID utente alternativo può essere impostato solo quando lo stato di apertura è FALSE. Questo metodo restituisce TRUE se ha esito positivo.

ImqBinary alternateSecurityId () const;
Restituisce una copia dell'ID di sicurezza alternativo.

ImqBoolean setAlternateSecurityId(const ImqBinary & ***token*** **);**
Imposta l'ID di sicurezza alternativo. L'id di sicurezza alternativo può essere impostato solo quando lo stato aperto è FALSE. La lunghezza dei dati di *token* deve essere zero o MQ_SECURITY_ID_LENGTH. Restituisce TRUE in caso di esito positivo.

ImqBoolean setAlternateSecurityId(const MQBYTE* ***token*** **= 0);**
Imposta l'ID di sicurezza alternativo. *token* può essere zero, che equivale a specificare MQSID_NONE. Se *token* è diverso da zero, deve indirizzare i byte MQ_SECURITY_ID_LENGTH dei dati binari. Quando si utilizzano valori predefiniti come MQSID_NONE, potrebbe essere necessario eseguire un cast per garantire la corrispondenza della firma; ad esempio, (MQBYTE *) MQSID_NONE.

L'ID di protezione alternativo può essere impostato solo quando lo stato aperto è TRUE. Restituisce TRUE in caso di esito positivo.

ImqBoolean setAlternateSecurityId(const unsigned char * ***ID*** **= 0);**
Imposta l'ID di sicurezza alternativo.

ImqBoolean close ();
Imposta lo stato di apertura su FALSE. Restituisce TRUE in caso di esito positivo.

MQLONG closeOptions () const;
Restituisce le opzioni di chiusura.

void setCloseOpzioni (const MQLONG *opzioni*);
Imposta le opzioni di chiusura.

ImqQueueImqQueue * connectionReference () const;
Restituisce il riferimento di connessione.

void setConnectionRiferimento (ImqQueueGestore & *gestore*);
Imposta il riferimento della connessione.

void setConnectionReference (ImqQueueGestore * *gestore* = 0);
Imposta il riferimento della connessione.

descrizione ImqBoolean virtuale (ImqString & *description*) = 0;
Fornisce una copia della descrizione. Restituisce TRUE in caso di esito positivo.

ImqString description ();
Restituisce una copia della descrizione senza alcuna indicazione di possibili errori.

nome ImqBoolean virtuale (ImqString & *name*);
Fornisce una copia del nome. Restituisce TRUE in caso di esito positivo.

ImqString name ();
Restituisce una copia del nome senza alcuna indicazione di possibili errori.

ImqBoolean setName (const char * *nome* = 0);
Imposta il nome. Il nome può essere impostato solo quando lo stato di apertura è FALSE e, per un gestore ImqQueue, mentre lo stato di connessione è FALSE. Restituisce TRUE in caso di esito positivo.

ImqObject * nextManagedObject () const;
Restituisce il successivo oggetto gestito.

ImqBoolean open ();
Modifica lo stato di apertura in TRUE aprendo l'oggetto come necessario, utilizzando tra gli altri attributi le opzioni di apertura e il nome. Questo metodo utilizza le informazioni di riferimento della connessione e il metodo di connessione del gestore ImqQueue, se necessario, per garantire che lo stato di connessione del gestore ImqQueuesia TRUE. Restituisce lo stato aperto.

ImqBoolean openFor (const MQLONG *required - options* = 0);
Tenta di assicurarsi che l'oggetto sia aperto con le opzioni di apertura o con le opzioni di apertura che garantiscono il funzionamento implicito dal valore del parametro *required - options* .

Se *required - options* è zero, l'input è obbligatorio e qualsiasi opzione di input è sufficiente. Quindi, se le opzioni di apertura contengono già una delle seguenti:

- MQOO_INPUT_AS_Q_DEF
- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE

le opzioni di apertura sono già soddisfacenti e non sono modificate; se le opzioni di apertura non contengono ancora alcuna di queste opzioni, MQOO_INPUT_AS_Q_DEF è impostato nelle opzioni di apertura.

Se *required - options* è diverso da zero, le opzioni richieste vengono aggiunte alle opzioni aperte; se *required - options* è una di queste opzioni, le altre vengono reimpostate.

Se una delle opzioni di apertura viene modificata e l'oggetto è già aperto, l'oggetto viene chiuso temporaneamente e riaperto per modificare le opzioni di apertura.

Restituisce TRUE in caso di esito positivo. L'esito positivo indica che l'oggetto è aperto con le opzioni appropriate.

MQLONG openOptions () const;
Restituisce le opzioni di apertura.

ImqBoolean setOpenOpzioni (const MQLONG *opzioni*);
Imposta le opzioni di apertura. Le opzioni di apertura possono essere impostate solo quando lo stato di apertura è FALSE. Restituisce TRUE in caso di esito positivo.

ImqBoolean openStatus () const;
Restituisce lo stato aperto.

ImqObject * previousManagedObject () const;
Restituisce l'oggetto gestito precedente.

ImqBoolean queueManagerIdentifier (ImqString & *id*);
Fornisce una copia dell'identificativo del gestore code. Restituisce TRUE in caso di esito positivo.

ImqString queueManagerIdentifier ();
Restituisce l'identificativo del gestore code senza alcuna indicazione di possibili errori.

Metodi oggetto (protetti)

ImqBoolean closeTemporarily ();
Chiude un oggetto in modo sicuro prima della riapertura. Restituisce TRUE in caso di esito positivo. Questo metodo presuppone che lo stato aperto sia TRUE.

MQHCONN connectionHandle () const;
Restituisce MQHCONN associato al riferimento della connessione. Questo valore è zero se non vi è alcun riferimento di connessione o se il gestore non è connesso.

ImqBoolean inquire (const MQLONG *int - attr*, MQLONG & *valore*);
Restituisce un valore intero, il cui indice è un valore MQIA_ *. In caso di errore, il valore è impostato su MQIAV_UNDEFINED.

ImqBoolean inquire (const MQLONG *attributo - carattere*, char * & *buffer*, const size_t *lunghezza*);
Restituisce una stringa di caratteri, il cui indice è un valore MQCA_ *.

Nota: Entrambi questi metodi restituiscono solo un singolo valore di attributo. Se è richiesta una *istantanea* di più di un valore, in cui i valori sono congruenti tra loro per un istante, IBM MQ C++ non fornisce questa funzione ed è necessario utilizzare la chiamata MQINQ con i parametri appropriati.

virtual void openInformationDisperse ();
Disperde le informazioni dalla sezione della variabile della struttura dati MQOD immediatamente dopo una chiamata MQOPEN.

ImqBoolean openInformationPrepare ();
Prepara le informazioni per la sezione della variabile della struttura dati MQOD immediatamente prima di una chiamata MQOPEN e restituisce TRUE in caso di esito positivo.

ImqBoolean set (const MQLONG *int - attr*, const MQLONG *valore*);
Imposta un attributo numero intero IBM MQ .

ImqBoolean set (const MQLONG *char - attr*, const char * *buffer*, const size_t *required - length*);
Imposta un attributo carattere IBM MQ .

void setNextManagedObject (const ImqObject * *object* = 0);
Imposta l'oggetto gestito successivo.

Attenzione: utilizzare questa funzione solo se si è certi che non interromperà l'elenco di oggetti gestiti.

void setPreviousManagedObject (const ImqObject * *object* = 0);
Imposta l'oggetto gestito precedente.

Attenzione: utilizzare questa funzione solo se si è certi che non interromperà l'elenco di oggetti gestiti.

Dati oggetto (protetti)

MQHOBJ *ohobj*
L'handle dell'oggetto IBM MQ (valido solo quando lo stato aperto è TRUE).

MQOD *omqod*
La struttura dati MQOD integrata. La quantità di memoria assegnata per questa struttura dati è quella richiesta per un MQOD Versione 2. Controllare il numero di versione *(omqod.Version)* e accedere agli altri campi nel modo seguente:

MQOD_VERSION_1
È possibile accedere a tutti gli altri campi in *omqod* .

MQOD_VERSION_2
È possibile accedere a tutti gli altri campi in *omqod* .

MQOD_VERSION_3
omqod.pmqod è un puntatore a un MQOD più grande, assegnato dinamicamente. Non è possibile accedere ad altri campi in *omqod* . È possibile accedere a tutti i campi indirizzati da *omqod.pmqod* .

Nota: *omqod.pmqod.Version* può essere inferiore a *omqod.Version*, indicando che IBM MQ MQI client ha più funzionalità del server IBM MQ .

Codici di origine

- MQRC_ATTRIBUTE_LOCKED
- MQRC_INCONSISTENT_OBJECT_STATE
- RIFERIMENTO MQRC_NO_CONNECTION_DI
- MQRC_STORAGE_NON_DISPONIBILE
- MQRC_REOPEN_SAVED_CONTEXT_ERR
- (codici di errore da MQCLOSE)
- (codici motivo da MQCONN)
- (codici motivo da MQINQ)
- (codici motivo da MQOPEN)
- (codici motivo da MQSET)

classe C++ ImqProcess

Questa classe incapsula un processo dell'applicazione (un oggetto IBM MQ di tipo MQOT_PROCESS) che può essere attivato da un controllo trigger.

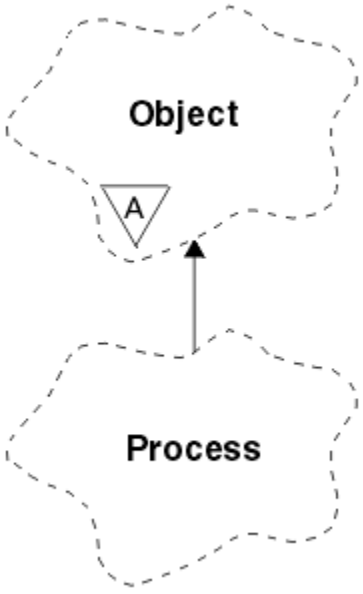


Figura 30. classe ImqProcess

Attributi oggetto

ID applicazione
L'identità del processo della domanda. Questo attributo è di sola lettura.

tipo di applicazione
Il tipo di processo della domanda. Questo attributo è di sola lettura.

Dati ambiente
Le informazioni sull'ambiente per il processo. Questo attributo è di sola lettura.

dati utente
Dati utente per il processo. Questo attributo è di sola lettura.

Costruttori

ImqProcess( );
Il costruttore predefinito.

ImqProcess(const ImqProcess & ***processo*** **);**
Il costruttore di copia. Lo **stato di apertura** di ImqObject è FALSE.

ImqProcess(const char * ***nome*** **);**
Imposta il nome ImqObject .

Metodi oggetto (pubblico)

void operator = (const ImqProcess & ***processo*** **);**
Esegue una chiusura, se necessario, quindi copia i dati dell'istanza dal *processo*. Lo **stato di apertura** di ImqObject sarà FALSE.

ImqBoolean applicationId (ImqString & ***ID*** **);**
Fornisce una copia dell' **ID applicazione**. Restituisce TRUE in caso di esito positivo.

ImqString applicationId ( );
Restituisce l' **ID applicazione** senza alcuna indicazione di possibili errori.

ImqBoolean applicationType (MQLONG & ***tipo*** **);**
Fornisce una copia del **tipo applicazione**. Restituisce TRUE in caso di esito positivo.

MQLONG applicationType ();
Restituisce il **tipo di applicazione** senza alcuna indicazione di possibili errori.

ImqBoolean environmentData (ImqString & ***dati*** **);**
Fornisce una copia dei **dati di ambiente**. Restituisce TRUE in caso di esito positivo.

ImqString environmentData ( );
Restituisce i **dati di ambiente** senza alcuna indicazione di possibili errori.

ImqBoolean userData (ImqString & ***data*** **);**
Fornisce una copia dei **dati utente**. Restituisce TRUE in caso di esito positivo.

ImqString userData ( );
Restituisce i **dati utente** senza alcuna indicazione di possibili errori.

Classe C++ ImqPutMessageOptions

Questa classe incapsula la struttura dati MQPMO.

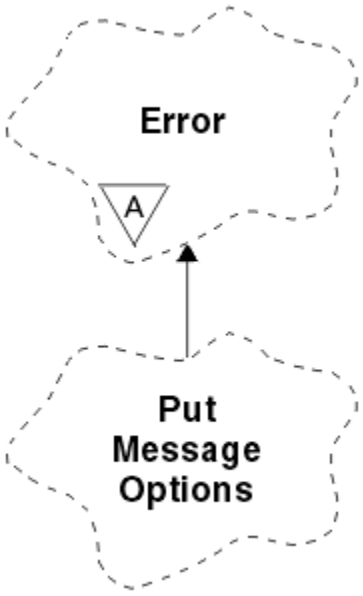


Figura 31. Classe ImqPutMessageOptions

Attributi oggetto

riferimento contesto
Una ImqQueue che fornisce un contesto per i messaggi. Inizialmente non esiste alcun riferimento.

opzioni
Le opzioni di inserimento del messaggio. Il valore iniziale è MQPMO_NONE. Sono possibili i seguenti valori aggiuntivi:

- MQPMO_SYNCPOINT
- MQPMO_NO_SYNCPOINT
- ID_MQPMO_NEW_MSG_
- ID_CORREL_NEW_MQPMO_
- ORDER MQPMO_LOGICAL_
- MQPMO_NO_CONTEXT
- MQPMO_DEFAULT_CONTEXT
- MQPMO_PASS_IDENTITY_CONTEXT
- MQPMO_PASS_ALL_CONTEXT
- MQPMO_SET_IDENTITY_CONTEXT
- MQPMO_SET_ALL_CONTEXT
- MQPMO_ALTERNATE_USER_AUTHORITY
- MQPMO_FAIL_IF_QUIESCING

campi record
Gli indicatori che controllano l'inclusione dei record dei messaggi di inserimento quando viene inserito un messaggio. Il valore iniziale è MQPMRF_NONE. Sono possibili i seguenti valori aggiuntivi:

- ID_MSG_MQPMRF
- ID CORREL_MQPMRF
- ID_GROUP_MQPMRF
- MQPMRF_FEEDBACK
- MQPMRF_ACCOUNTING_TOKEN

ImqMessageGli attributi del programma di traccia vengono presi dall'oggetto per qualsiasi campo specificato. Gli attributi del programma di traccia ImqMessagevengono presi dall'oggetto ImqMessage per qualsiasi campo non specificato.

Nome del gestore code risolto
Nome di un gestore code di destinazione determinato durante un'operazione di inserimento. Il valore iniziale è null. Questo attributo è di sola lettura.

Nome coda risolto
Nome di una coda di destinazione determinata durante un'operazione di inserimento. Il valore iniziale è null. Questo attributo è di sola lettura.

partecipazione punto di sincronizzazione
TRUE quando i messaggi vengono posti sotto il controllo del punto di sincronizzazione.

Costruttori

ImqPutMessageOptions( );
Il costruttore predefinito.

ImqPutMessageOptions(const ImqPutMessageOptions & ***pmo*** **);**
Il costruttore di copia.

Metodi oggetto (pubblico)

operatore void = (const ImqPutMessageOptions & ***pmo*** **);**
Copia i dati di istanza da *pmo*, sostituendo i dati di istanza esistenti.

ImqQueue * contextReference () const;
Restituisce il riferimento di contesto.

void setContextReference (const ImqQueue & ***queue*** **);**
Imposta il riferimento di contesto.

void setContextReference (const ImqQueue * ***queue*** **= 0);**
Imposta il riferimento di contesto.

Opzioni MQLONG () const;
Restituisce le opzioni.

void setOptions (const MQLONG ***opzioni*** **);**
Imposta le opzioni, incluso il valore di partecipazione del punto di sincronizzazione.

MQLONG recordFields () const;
Restituisce i campi del record.

void setRecordFields (const MQLONG ***campi*** **);**
Imposta i campi record.

ImqString resolvedQueueManagerName () const;
Restituisce una copia del nome gestore code risolto.

ImqString resolvedQueueNome () const;
Restituisce una copia del nome della coda risolta.

ImqBoolean syncPointPartecipazione () const;
Restituisce il valore di partecipazione del punto di sincronizzazione, TRUE se le opzioni includono MQPMO_SYNCPOINT.

void setSyncPointParticipation (const ImqBoolean *sync*);
Imposta il valore di partecipazione del punto di sincronizzazione. Se *sync* è TRUE, le opzioni vengono modificate per includere MQPMO_SYNCPOINT e per escludere MQPMO_NO_SYNCPOINT. Se *sync* è FALSE, le opzioni vengono modificate per includere MQPMO_NO_SYNCPOINT ed escludere MQPMO_SYNCPOINT.

Dati oggetto (protetti)

MQPMO *omqpmo*
La struttura dati MQPMO.

Codici di origine

- MQRC_STORAGE_NON_DISPONIBILE

Classe ImqQueue C++

Questa classe incapsula una coda messaggi (un oggetto IBM MQ di tipo MQOT_Q).

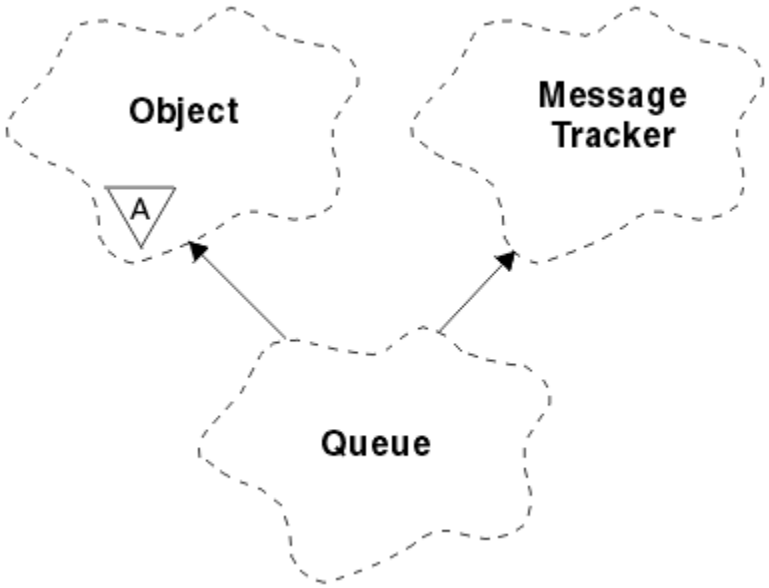


Figura 32. Classe ImqQueue

Questa classe è relativa alle chiamate MQI elencate in Tabella 862 a pagina 1843.

- "Attributi oggetto" a pagina 1904
- "Costruttori" a pagina 1907
- "Metodi oggetto (pubblico)" a pagina 1908
- "Metodi oggetto (protetti)" a pagina 1914
- "Codici di origine" a pagina 1914

Attributi oggetto

Nome di riaccodamento di backout
Nome riaccodamento di backout eccessivo. Questo attributo è di sola lettura.

soglia di backout
La soglia di ripristino. Questo attributo è di sola lettura.

nome coda di base
Nome della coda in cui si risolve l'alias. Questo attributo è di sola lettura.

nome del cluster
Nome cluster. Questo attributo è di sola lettura.

Nome elenco nomi cluster
Nome elenco nomi cluster. Questo attributo è di sola lettura.

Classif. carico lavoro cluster
La classificazione del carico di lavoro del cluster. Questo attributo è di sola lettura.

Priorità carico lavoro cluster
La priorità del carico di lavoro del cluster. Questo attributo è di sola lettura.

Coda utilizzo carico di lavoro cluster
Il valore della coda di utilizzo del carico di lavoro del cluster. Questo attributo è di sola lettura.

data di creazione
Dati di creazione della coda. Questo attributo è di sola lettura.

ora di creazione
Ora di creazione della coda. Questo attributo è di sola lettura.

Profondità corrente
Numero di messaggi sulla coda. Questo attributo è di sola lettura.

bind predefinito
Bind predefinito. Questo attributo è di sola lettura.

Opzione predefinita di apertura in immissione
Opzione open - for - input predefinita. Questo attributo è di sola lettura.

Persistenza predefinita
Persistenza del messaggio predefinita. Questo attributo è di sola lettura.

Priorità predefinita
La priorità messaggi predefinita. Questo attributo è di sola lettura.

tipo di definizione
Il tipo di definizione della coda. Questo attributo è di sola lettura.

evento profondità elevata
Attributo di controllo per eventi di grandezza della coda elevata. Questo attributo è di sola lettura.

limite massimo profondità
Limite massimo per la profondità della coda. Questo attributo è di sola lettura.

evento profondità bassa
Attributo di controllo per gli eventi di profondità della coda bassa. Questo attributo è di sola lettura.

limite minimo profondità
Limite basso per la profondità della coda. Questo attributo è di sola lettura.

evento profondità massima
Attributo di controllo per il numero massimo di eventi di profondità coda. Questo attributo è di sola lettura.

riferimento elenco di distribuzione
Riferimento facoltativo a un elenco ImqDistributionche può essere utilizzato per distribuire i messaggi a più di una coda, inclusa questa. Il valore iniziale è null.

Nota: Quando viene aperto un oggetto ImqQueue , qualsiasi oggetto Elenco ImqDistributionaperto a cui fa riferimento viene chiuso automaticamente.

elenchi di distribuzione
La capacità di una coda di trasmissione di supportare gli elenchi di distribuzione. Questo attributo è di sola lettura.

Nome coda dinamica
Nome coda dinamica. Il valore iniziale è AMQ.* per tutte le piattaforme AIX, Linux, and Windows .

Ripristino forzato get
Indica se rafforzare il conteggio di backout. Questo attributo è di sola lettura.

Tipo di indice
Tipo di indice. Questo attributo è di sola lettura.

inibisci get
Indica se le operazioni get sono consentite. Il valore iniziale dipende dalla definizione della coda. Questo attributo è valido solo per un alias o una coda locale.

inibisci inserimento
Se le operazioni di inserimento sono consentite. Il valore iniziale dipende dalla definizione della coda.

Nome coda di attivazione
Nome della coda di iniziazione. Questo attributo è di sola lettura.

Profondità massima
Numero massimo di messaggi consentiti nella coda. Questo attributo è di sola lettura.

Lunghezza massima dei messaggi
La lunghezza massima per qualsiasi messaggio su questa coda, che può essere inferiore al valore massimo per qualsiasi coda gestita dal gestore code associato. Questo attributo è di sola lettura.

Sequenza di consegna messaggi
Se la priorità del messaggio è rilevante. Questo attributo è di sola lettura.

coda distribuita successiva
L'oggetto successivo di questa classe, in nessun ordine particolare, ha lo stesso **riferimento elenco di distribuzione** di questo oggetto. Il valore iniziale è zero.

Se un oggetto in una catena viene eliminato, l'oggetto precedente e l'oggetto successivo vengono aggiornati in modo che i relativi collegamenti della coda distribuita non puntino più all'oggetto eliminato.

classe di messaggi non persistenti
Livello di affidabilità per i messaggi non persistenti inseriti in questa coda. Questo attributo è di sola lettura.

Conteggio input aperti
Numero di oggetti ImqQueue aperti per l'input. Questo attributo è di sola lettura.

Conteggio output aperti
Numero di oggetti ImqQueue aperti per l'output. Questo attributo è di sola lettura.

coda distribuita precedente
Oggetto precedente di questa classe, in nessun ordine particolare, con lo stesso **riferimento elenco di distribuzione** di questo oggetto. Il valore iniziale è zero.

Se un oggetto in una catena viene eliminato, l'oggetto precedente e l'oggetto successivo vengono aggiornati in modo che i relativi collegamenti della coda distribuita non puntino più all'oggetto eliminato.

nome del processo
Nome della definizione del processo. Questo attributo è di sola lettura.

Conto coda
Livello delle informazioni di account per le code. Questo attributo è di sola lettura.

nome gestore code
Il nome del gestore code (possibilmente remoto) in cui risiede la coda. Non confondere il gestore code qui denominato con il **riferimento di connessione** ImqObject , che fa riferimento al gestore code (locale) che fornisce una connessione. Il valore iniziale è null.

Controllo coda
Livello di raccolta dei dati di controllo per la coda. Questo attributo è di sola lettura.

Informazioni coda
Livello di dati statistici per la coda. Questo attributo è di sola lettura.

tipo di coda
Il tipo di coda. Questo attributo è di sola lettura.

Nome gestore code remoto
Il nome del gestore code remoto. Questo attributo è di sola lettura.

Nome coda remota
Nome della coda remota come noto sul gestore code remoto. Questo attributo è di sola lettura.

Nome del gestore code risolto
Nome gestore code risolto. Questo attributo è di sola lettura.

Nome coda risolto
Nome coda risolto. Questo attributo è di sola lettura.

intervallo di comservazione
Intervallo di conservazione della coda. Questo attributo è di sola lettura.

ambito
Ambito della definizione della coda. Questo attributo è di sola lettura.

intervallo di servizio
Intervallo di servizio. Questo attributo è di sola lettura.

evento intervallo di servizio
Attributo di controllo per eventi di intervallo di servizio. Questo attributo è di sola lettura.

Condivisione
Specifica se la coda può essere condivisa. Questo attributo è di sola lettura.

classe di memorizzazione
Classe di memoria. Questo attributo è di sola lettura.

Nome coda di trasmissione
Il nome della coda di trasmissione. Questo attributo è di sola lettura.

Controllo trigger
Controllo trigger. Il valore iniziale dipende dalla definizione della coda. Questo attributo è valido solo per una coda locale.

dati del trigger
I dati del trigger. Il valore iniziale dipende dalla definizione della coda. Questo attributo è valido solo per una coda locale.

Capacità di Trigger
La lunghezza del trigger. Il valore iniziale dipende dalla definizione della coda. Questo attributo è valido solo per una coda locale.

Priorità messaggio trigger
La priorità dei messaggi per i trigger. Il valore iniziale dipende dalla definizione della coda. Questo attributo è valido solo per una coda locale.

tipo trigger
Il tipo di trigger. Il valore iniziale dipende dalla definizione della coda. Questo attributo è valido solo per una coda locale.

utilizzo
Utilizzo. Questo attributo è di sola lettura.

Costruttori

ImqQueue( );
Il costruttore predefinito.

ImqQueue(coda ImqQueue & *coda*);
Il costruttore di copia. Lo **stato di apertura** di ImqObject sarà FALSE.

ImqQueue(carattere const * *nome*);
Imposta il nome ImqObject .

Metodi oggetto (pubblico)

void operator = (const ImqQueue & *queue*);
Esegue una chiusura, se necessario, e quindi copia i dati dell'istanza dalla *coda*. Lo **stato di apertura** di ImqObject sarà FALSE.

ImqBoolean backoutRequeueNome (ImqString & *name*);
Fornisce una copia del **nome della coda di backout**. Restituisce TRUE in caso di esito positivo.

ImqString backoutRequeueNome ();
Restituisce il **nome della riaccodamento di backout** senza alcuna indicazione di possibili errori.

ImqBoolean backoutThreshold (MQLONG & *soglia*);
Fornisce una copia della **soglia di backout**. Restituisce TRUE in caso di esito positivo.

MQLONG backoutThreshold ();
Restituisce il valore della **soglia di backout** senza alcuna indicazione di possibili errori.

ImqBoolean baseQueueName (ImqString & *nome*);
Fornisce una copia del **nome coda base**. Restituisce TRUE in caso di esito positivo.

ImqString baseQueueNome ();
Restituisce il **nome della coda di base** senza alcuna indicazione di possibili errori.

ImqBoolean clusterName(ImqString & *nome*);
Fornisce una copia del **nome cluster**. Restituisce TRUE in caso di esito positivo.

ImqString clusterName( );
Restituisce il **nome cluster** senza alcuna indicazione di possibili errori.

ImqBoolean clusterNamelistNome (ImqString & *name*);
Fornisce una copia del **nome elenco nomi cluster**. Restituisce TRUE in caso di esito positivo.

ImqString clusterNamelistNome ();
Restituisce il **nome dell'elenco nomi cluster** senza alcuna indicazione di errori.

ImqBoolean clusterWorkLoadPriority (MQLONG e priorità);
Fornisce una copia del valore di priorità del carico di lavoro del cluster. Restituisce TRUE in caso di esito positivo.

MQLONG clusterWorkLoadPriority ();
Restituisce il valore di priorità del carico di lavoro del cluster senza alcuna indicazione di possibili errori.

ImqBoolean clusterWorkLoadRank (MQLONG & rank);
Fornisce una copia del valore di classificazione del workload del cluster. Restituisce TRUE in caso di esito positivo.

MQLONG clusterWorkLoadRank ();
Restituisce il valore di classificazione del carico di lavoro del cluster senza alcuna indicazione di possibili errori.

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);
Fornisce una copia del valore coda di utilizzo del carico di lavoro cluster. Restituisce TRUE in caso di esito positivo.

MQLONG clusterWorkLoadUseQ ();
Restituisce il valore della coda di utilizzo del carico di lavoro del cluster senza alcuna indicazione di possibili errori.

ImqBoolean creationDate (ImqString & *data*);
Fornisce una copia della **data di creazione**. Restituisce TRUE in caso di esito positivo.

ImqString creationDate ( );
Restituisce la **data di creazione** senza alcuna indicazione di possibili errori.

ImqBoolean creationTime (ImqString & *ora*);
Fornisce una copia dell' **ora di creazione**. Restituisce TRUE in caso di esito positivo.

ImqString creationTime ( );
Restituisce l' **ora di creazione** senza alcuna indicazione di possibili errori.

ImqBoolean currentDepth (MQLONG & *profondità*);
Fornisce una copia della **profondità corrente**. Restituisce TRUE in caso di esito positivo.

MQLONG currentDepth ();
Restituisce la **profondità corrente** senza alcuna indicazione di possibili errori.

ImqBoolean defaultInputOpenOption (opzione MQLONG &);
Fornisce una copia dell' **opzione di apertura di input predefinita**. Restituisce TRUE in caso di esito positivo.

MQLONG defaultInputOpenOption ();
Restituisce l' **opzione di apertura di input predefinita** senza alcuna indicazione di possibili errori.

ImqBoolean defaultPersistence (MQLONG & *persistenza*);
Fornisce una copia della **persistenza predefinita**. Restituisce TRUE in caso di esito positivo.

MQLONG defaultPersistence ();
Restituisce la **persistenza predefinita** senza alcuna indicazione di possibili errori.

ImqBoolean defaultPriority (MQLONG & *priorità*);
Fornisce una copia della **priorità predefinita**. Restituisce TRUE in caso di esito positivo.

MQLONG defaultPriority ();
Restituisce la **priorità predefinita** senza alcuna indicazione di possibili errori.

ImqBoolean defaultBind (MQLONG & *bind*);
Fornisce una copia del **bind predefinito**. Restituisce TRUE in caso di esito positivo.

MQLONG defaultBind ();
Restituisce il **bind predefinito** senza alcuna indicazione di possibili errori.

ImqBoolean definitionType (MQLONG & *tipo*);
Fornisce una copia del **tipo di definizione**. Restituisce TRUE in caso di esito positivo.

MQLONG definitionType ();
Restituisce il **tipo di definizione** senza alcuna indicazione di possibili errori.

ImqBoolean depthHighEvent (MQLONG & *evento*);
Fornisce una copia dello stato di abilitazione dell' **evento profondità elevata**. Restituisce TRUE in caso di esito positivo.

MQLONG depthHighEvent ();
Restituisce lo stato di abilitazione dell' **evento di profondità elevata** senza alcuna indicazione di possibili errori.

ImqBoolean depthHighLimit (MQLONG & *limite*);
Fornisce una copia del **limite massimo di profondità**. Restituisce TRUE in caso di esito positivo.

MQLONG depthHighLimite ();
Restituisce il valore **limite massimo profondità** senza alcuna indicazione di possibili errori.

ImqBoolean depthLowEvent (MQLONG & evento);
Fornisce una copia dello stato di abilitazione dell' **evento profondità bassa**. Restituisce TRUE in caso di esito positivo.

MQLONG depthLowEvent ();
Restituisce lo stato di abilitazione dell' **evento profondità bassa** senza alcuna indicazione di possibili errori.

ImqBoolean depthLowLimit (MQLONG & *limite*);
Fornisce una copia del **limite minimo di profondità**. Restituisce TRUE in caso di esito positivo.

MQLONG depthLowLimite ();
Restituisce il valore di **limite minimo di profondità** senza alcuna indicazione di possibili errori.

ImqBoolean depthMaximumEvent (MQLONG e *evento*);
Fornisce una copia dello stato di abilitazione dell' **evento profondità massima**. Restituisce TRUE in caso di esito positivo.

MQLONG depthMaximumEvento ();
Restituisce lo stato di abilitazione dell' **evento di profondità massima** senza alcuna indicazione di possibili errori.

ImqDistributionElenco * distributionListRiferimento () const ;
Restituisce il **riferimento dell'elenco di distribuzione**.

void setDistributionListReference (ImqDistributionList & *list*);
Imposta il **riferimento dell'elenco di distribuzione**.

void setDistributionListReference (ImqDistributionList * *list* = 0);
Imposta il **riferimento dell'elenco di distribuzione**.

ImqBoolean distributionLists (MQLONG & supporto);
Fornisce una copia del valore degli **elenchi di distribuzione** . Restituisce TRUE in caso di esito positivo.

MQLONG distributionLists ();
Restituisce il valore degli **elenchi di distribuzione** senza alcuna indicazione di possibili errori.

ImqBoolean setDistributionLists (const MQLONG *support*);
Imposta il valore **elenchi di distribuzione** . Restituisce TRUE in caso di esito positivo.

ImqString dynamicQueueNome () const ;
Restituisce una copia di **nome coda dinamica**.

ImqBoolean setDynamicQueueName (const char * *name*);
Imposta il **nome coda dinamica**. Il **nome coda dinamica** può essere impostato solo se lo **stato di apertura** di ImqObject è FALSE. Restituisce TRUE in caso di esito positivo.

ImqBoolean get (ImqMessage & *msg*, ImqGetMessageOptions & *options*);
Richiama un messaggio dalla coda, utilizzando le *opzioni*specificate. Richiama il metodo ImqObject **openFor** se necessario per garantire che le **opzioni di apertura** di ImqObject includano uno dei valori MQOO_INPUT_ * o il valore MQOO_BROWSE, a seconda delle *opzioni*. Se l'oggetto *msg* ha un **buffer automatico** ImqCache , il buffer cresce per contenere qualsiasi messaggio richiamato. Il metodo **clearMessage** viene richiamato rispetto all'oggetto *msg* prima del recupero.

Questo metodo restituisce TRUE se ha esito positivo.

Nota: Il risultato del richiamo del metodo è FALSE se il **codice di errore** ImqObject è MQRC_TRUNCATED_MSG_FAILED, anche se questo **codice di errore** è classificato come avvertenza. Se viene accettato un messaggio troncato, la **lunghezza del messaggio** di ImqCache riflette la lunghezza troncata. In entrambi i casi, la **lunghezza totale del messaggio** di ImqMessage indica il numero di byte disponibili.

ImqBoolean get (ImqMessage & *msg*);
Come per il metodo precedente, tranne che vengono utilizzate le opzioni di richiamo del messaggio predefinite.

ImqBoolean get (ImqMessage & *msg*, ImqGetMessageOptions & *options*, const size_t *dimensione - buffer*);
Come per i due metodi precedenti, ad eccezione del fatto che viene indicata una *dimensione - buffer* di sostituzione. Se l'oggetto *msg* utilizza un **buffer automatico** ImqCache , il metodo **resizeBuffer** viene richiamato sull'oggetto *msg* prima del richiamo del messaggio e il buffer non cresce ulteriormente per accogliere i messaggi più grandi.

ImqBoolean ottieni (ImqMessage & *msg*, size_t const *dimensione buffer*);
Come per il metodo precedente, tranne che vengono utilizzate le opzioni di richiamo del messaggio predefinite.

ImqBoolean hardenGetBackout (MQLONG & *harden*);
Fornisce una copia del valore **harden get backout** . Restituisce TRUE in caso di esito positivo.

MQLONG hardenGetBackout ();
Restituisce il valore **harden get backout** senza alcuna indicazione di possibili errori.

ImqBoolean indexType(MQLONG & *tipo*);
Fornisce una copia del **tipo indice**. Restituisce TRUE in caso di esito positivo.

MQLONG indexType();
Restituisce il **tipo di indice** senza alcuna indicazione di possibili errori.

ImqBoolean inhibitGet (MQLONG & ***inibizione*** **);**
Fornisce una copia del valore **inhibit get** . Restituisce TRUE in caso di esito positivo.

MQLONG inhibitGet ();
Restituisce il valore **inhibit get** senza alcuna indicazione di possibili errori.

ImqBoolean setInhibitGet (const MQLONG ***inhibit*** **);**
Imposta il valore **inhibit get** . Restituisce TRUE in caso di esito positivo.

ImqBoolean inhibitPut (MQLONG & ***inibizione*** **);**
Fornisce una copia del valore **inhibit put** . Restituisce TRUE in caso di esito positivo.

MQLONG inhibitPut ();
Restituisce il valore **inhibit put** senza alcuna indicazione di possibili errori.

ImqBoolean setInhibitPut (const MQLONG ***inhibit*** **);**
Imposta il valore **inhibit put** . Restituisce TRUE in caso di esito positivo.

ImqBoolean initiationQueueNome (ImqString & ***nome*** **);**
Fornisce una copia del **nome della coda di avvio**. Restituisce TRUE in caso di esito positivo.

ImqString initiationQueueNome ();
Restituisce il **nome della coda di inizializzazione** senza alcuna indicazione di possibili errori.

ImqBoolean maximumDepth (MQLONG & ***profondità*** **);**
Fornisce una copia della **profondità massima**. Restituisce TRUE in caso di esito positivo.

MQLONG maximumDepth ();
Restituisce la **profondità massima** senza alcuna indicazione di possibili errori.

ImqBoolean maximumMessageLunghezza (MQLONG & ***lunghezza*** **);**
Fornisce una copia della **lunghezza massima del messaggio**. Restituisce TRUE in caso di esito positivo.

MQLONG maximumMessageLunghezza ();
Restituisce la **lunghezza massima del messaggio** senza alcuna indicazione di possibili errori.

ImqBoolean messageDeliverySequenza (MQLONG & ***sequenza*** **);**
Fornisce una copia della **sequenza di consegna dei messaggi**. Restituisce TRUE in caso di esito positivo.

MQLONG messageDeliverySequenza ();
Restituisce il valore **sequenza di consegna del messaggio** senza alcuna indicazione di possibili errori.

ImqQueue * nextDistributednextDistributed () const ;
Restituisce la **successiva coda distribuita**.

ImqBoolean nonPersistentMessageClass (MQLONG & monq);
Fornisce una copia del valore della classe di messaggi non persistenti. Restituisce TRUE in caso di esito positivo.

MQLONG nonPersistentMessageClass ();
Restituisce il valore della classe di messaggi non persistenti senza alcuna indicazione di possibili errori.

ImqBoolean openInputConteggio (MQLONG & ***conteggio*** **);**
Fornisce una copia del **conteggio input aperti**. Restituisce TRUE in caso di esito positivo.

MQLONG openInputConteggio ();
Restituisce il **conteggio di input aperto** senza alcuna indicazione di possibili errori.

ImqBoolean openOutputConteggio (MQLONG & ***conteggio*** **);**
Fornisce una copia del **conteggio di output aperti**. Restituisce TRUE in caso di esito positivo.

MQLONG openOutputConteggio ();
Restituisce il **conteggio di output aperto** senza alcuna indicazione di possibili errori.

ImqQueue * previousDistributedQueue () const ;
Restituisce la **coda distribuita precedente**.

ImqBoolean processName (ImqString & ***nome*** **);**
Fornisce una copia del **nome processo**. Restituisce TRUE in caso di esito positivo.

ImqString processName ( );
Restituisce il **nome processo** senza alcuna indicazione di possibili errori.

ImqBoolean put (ImqMessage & ***msg*** **);**
Inserisce un messaggio nella coda, utilizzando le opzioni di inserimento del messaggio predefinite. Utilizza il metodo ImqObject **openFor** se necessario per garantire che le opzioni di apertura di ImqObject includano MQOO_OUTPUT.

Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean put (ImqMessage & ***msg*****, ImqPutMessageOptions &** ***pmo*** **);**
Inserisce un messaggio nella coda, utilizzando il *pmo*specificato. Utilizza il metodo ImqObject **openFor** come necessario per garantire che le opzioni di apertura di ImqObject includano MQOO_OUTPUT e (se le opzioni *pmo* includono uno qualsiasi dei valori MQPMO_PASS_IDENTITY_CONTEXT, MQPMO_PASS_ALL_CONTEXT, MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT) MQOO* CONTEXT corrispondenti.

Questo metodo restituisce TRUE se ha esito positivo.

Nota: Se il *pmo* include un **riferimento di contesto**, l'oggetto di riferimento viene aperto, se necessario, per fornire un contesto.

ImqBoolean queueAccounting (MQLONG & acctq);
Fornisce una copia del valore di account coda. Restituisce TRUE in caso di esito positivo.

MQLONG queueAccounting ();
Restituisce il valore di account della coda senza alcuna indicazione di possibili errori.

ImqString queueManagerNome () const ;
Restituisce il **nome gestore code**.

ImqBoolean setQueueManagerName (const char * ***nome*** **);**
Imposta il **Nome gestore code**. Il **nome gestore code** può essere impostato solo se lo **stato di apertura** di ImqObject è FALSE. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean queueMonitoring (MQLONG & monq);
Fornisce una copia del valore di controllo della coda. Restituisce TRUE in caso di esito positivo.

MQLONG queueMonitoring ();
Restituisce il valore di monitoraggio della coda senza alcuna indicazione di possibili errori.

ImqBoolean queueStatistics (MQLONG & statq);
Fornisce una copia del valore delle statistiche della coda. Restituisce TRUE in caso di esito positivo.

MQLONG queueStatistics ();
Restituisce il valore delle statistiche della coda senza alcuna indicazione di possibili errori.

ImqBoolean queueType (MQLONG & ***tipo*** **);**
Fornisce una copia del valore **tipo coda** . Restituisce TRUE in caso di esito positivo.

MQLONG queueType ();
Restituisce il **tipo di coda** senza alcuna indicazione di possibili errori.

ImqBoolean remoteQueueManagerName (ImqString & ***name*** **);**
Fornisce una copia del **nome gestore code remoto**. Restituisce TRUE in caso di esito positivo.

ImqString remoteQueueManagerName ( );
Restituisce il **nome gestore code remoto** senza alcuna indicazione di possibili errori.

ImqBoolean remoteQueueName (ImqString & ***name*** **);**
Fornisce una copia del **nome coda remota**. Restituisce TRUE in caso di esito positivo.

ImqString remoteQueueNome ();
Restituisce il **nome della coda remota** senza alcuna indicazione di possibili errori.

ImqBoolean resolvedQueueManagerName(ImqString & ***nome*** **);**
Fornisce una copia del **nome gestore code risolto**. Restituisce TRUE in caso di esito positivo.

Nota: Questo metodo non riesce a meno che MQOO_RESOLVE_NAMES non sia tra le ImqObject **opzioni di apertura**.

ImqString resolvedQueueManagerName( ) ;
Restituisce il **nome gestore code risolto**, senza alcuna indicazione di possibili errori.

ImqBoolean resolvedQueueNome (ImqString & ***name*** **);**
Fornisce una copia del **nome coda risolto**. Restituisce TRUE in caso di esito positivo.

Nota: Questo metodo non riesce a meno che MQOO_RESOLVE_NAMES non sia tra le ImqObject **opzioni di apertura**.

ImqString resolvedQueueNome ();
Restituisce il **nome della coda risolta**, senza alcuna indicazione di possibili errori.

ImqBoolean retentionInterval (MQLONG & ***intervallo*** **);**
Fornisce una copia dell' **intervallo di conservazione**. Restituisce TRUE in caso di esito positivo.

MQLONG retentionInterval ();
Restituisce l' **intervallo di conservazione** senza alcuna indicazione di possibili errori.

ImqBoolean ambito (MQLONG & ***ambito*** **);**
Fornisce una copia dell' **ambito**. Restituisce TRUE in caso di esito positivo.

MQLONG ambito ();
Restituisce l' **ambito** senza alcuna indicazione di possibili errori.

ImqBoolean serviceInterval (MQLONG & ***intervallo*** **);**
Fornisce una copia dell' **intervallo di servizio**. Restituisce TRUE in caso di esito positivo.

MQLONG serviceInterval ();
Restituisce l' **intervallo di servizio** senza alcuna indicazione di possibili errori.

ImqBoolean serviceIntervalEvent (evento MQLONG &);
Fornisce una copia dello stato di abilitazione dell' **evento intervallo di servizi**. Restituisce TRUE in caso di esito positivo.

MQLONG serviceIntervalEvent ();
Restituisce lo stato di abilitazione dell' **evento intervallo di servizio** senza alcuna indicazione di possibili errori.

ImqBoolean condivisionabilità (MQLONG & ***condivisionabilità*** **);**
Fornisce una copia del valore **shareability** . Restituisce TRUE in caso di esito positivo.

MQLONG condivisibile ();
Restituisce il valore di **condivisibilità** senza alcuna indicazione di possibili errori.

ImqBoolean storageClass(ImqString & ***classe*** **);**
Fornisce una copia della **classe di memoria**. Restituisce TRUE in caso di esito positivo.

ImqString storageClass( );
Restituisce la **classe di memorizzazione** senza alcuna indicazione di possibili errori.

ImqBoolean transmissionQueueNome (ImqString & ***nome*** **);**
Fornisce una copia del **nome coda di trasmissione**. Restituisce TRUE in caso di esito positivo.

ImqString transmissionQueueNome ();
Restituisce il **nome coda di trasmissione** senza alcuna indicazione di possibili errori.

ImqBoolean triggerControl (MQLONG & ***controllo*** **);**
Fornisce una copia del valore **controllo trigger** . Restituisce TRUE in caso di esito positivo.

MQLONG triggerControl ();
Restituisce il valore di **controllo trigger** senza alcuna indicazione di possibili errori.

ImqBoolean setTriggerControl (const MQLONG ***controllo*** **);**
Imposta il valore **controllo trigger** . Restituisce TRUE in caso di esito positivo.

ImqBoolean triggerData (ImqString & ***dati*** **);**
Fornisce una copia dei **dati trigger**. Restituisce TRUE in caso di esito positivo.

ImqString triggerData ( );
Restituisce una copia dei **dati trigger** senza alcuna indicazione di possibili errori.

ImqBoolean setTriggerData (const char * ***data*** **);**
Imposta i **dati trigger**. Restituisce TRUE in caso di esito positivo.

ImqBoolean triggerDepth (MQLONG & ***profondità*** **);**
Fornisce una copia della **profondità trigger**. Restituisce TRUE in caso di esito positivo.

MQLONG triggerDepth ();
Restituisce la **profondità trigger** senza alcuna indicazione di possibili errori.

ImqBoolean setTriggerProfondità (const MQLONG ***profondità*** **);**
Imposta la **profondità trigger**. Restituisce TRUE in caso di esito positivo.

ImqBoolean triggerMessagePriority (MQLONG & ***priorità*** **);**
Fornisce una copia della **priorità del messaggio trigger**. Restituisce TRUE in caso di esito positivo.

MQLONG triggerMessagePriorità ();
Restituisce la **priorità del messaggio trigger** senza alcuna indicazione di possibili errori.

ImqBoolean setTriggerMessagePriority (const MQLONG ***priorità*** **);**
Imposta la **priorità del messaggio trigger**. Restituisce TRUE in caso di esito positivo.

ImqBoolean triggerType (MQLONG & ***tipo*** **);**
Fornisce una copia del **tipo di trigger**. Restituisce TRUE in caso di esito positivo.

MQLONG triggerType ();
Restituisce il **tipo di trigger** senza alcuna indicazione di possibili errori.

ImqBoolean setTriggerTipo (const MQLONG ***tipo*** **);**
Imposta il **tipo di trigger**. Restituisce TRUE in caso di esito positivo.

ImqBoolean utilizzo (MQLONG & ***utilizzo*** **);**
Fornisce una copia del valore **usage** . Restituisce TRUE in caso di esito positivo.

MQLONG utilizzo ();
Restituisce il valore **usage** senza alcuna indicazione di possibili errori.

Metodi oggetto (protetti)

void setNextDistributedQueue (ImqQueue * ***queue*** **= 0);**
Imposta la **successiva coda distribuita**.

Attenzione: utilizzare questa funzione solo se si è certi che non interromperà l'elenco di code distribuite.

void setPreviousDistributedQueue (ImqQueue * ***coda*** **= 0);**
Imposta la **coda distribuita precedente**.

Attenzione: utilizzare questa funzione solo se si è certi che non interromperà l'elenco di code distribuite.

Codici di origine

- MQRC_ATTRIBUTE_LOCKED
- MQRC_CONTEXT_OBJECT_NOT_VALID
- ERRORE APERTURA CONTENUTO MQRC_
- MQRC_CURSOR_NO_VALID
- MQRC_NO_BUFFER
- ERRORE_INPUT_REOPEN_MQRC_REOPEN_EXCL_
- MQRC_REOPEN_INQUIRE_ERROR
- ERRORE CODA MQRC_REOPEN_TEMPORARY_Q
- (codici motivo da MQGET)
- (codici motivo da MQPUT)

Classe C++ ImqQueueManager

Questa classe incapsula un gestore code (un oggetto IBM MQ di tipo MQOT_Q_MGR).

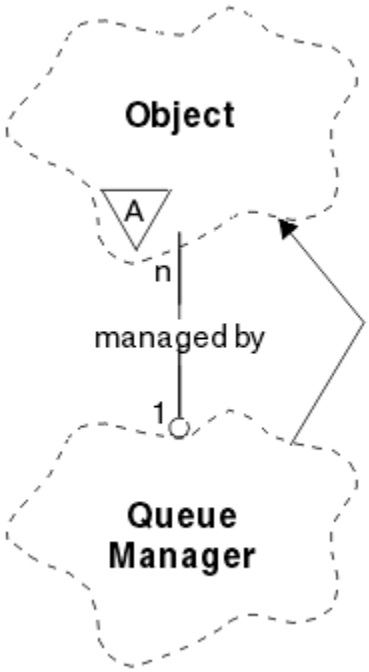


Figura 33. Classe gestore ImqQueue

Questa classe è relativa alle chiamate MQI elencate in "Riferimento incrociato ImqQueueManager" a pagina 1845. Non tutti i metodi elencati sono applicabili a tutte le piattaforme; per ulteriori dettagli, consultare ALTER QMGR .

Attributi classe

comportamento
Controlla il comportamento della connessione implicita e della disconnessione.

IMQ_EXPL_DISC_BACKOUT (0L)
Una chiamata esplicita al metodo disconnect implica il backout. Questo attributo si esclude reciprocamente con IMQ_EXPL_DISC_COMMIT.

IMQ_EXPL_DISC_COMMIT (1L)
Una chiamata esplicita al metodo di disconnessione implica il commit (impostazione predefinita). Questo attributo si esclude reciprocamente con IMQ_EXPL_DISC_BACKOUT.

IMQ_IMPL_CONN (2L)
La connessione implicita è consentita (impostazione predefinita).

IMQ_IMPL_DISC_BACKOUT (0L)
Una chiamata implicita al metodo di disconnessione, che può verificarsi durante la distruzione dell'oggetto, implica il backout. Questo attributo si esclude a vicenda con IMQ_IMPL_DISC_COMMIT.

IMQ_IMPL_DISC_COMMIT (4L)
Una chiamata implicita al metodo di disconnessione, che può verificarsi durante la distruzione dell'oggetto, implica il commit (impostazione predefinita). Questo attributo si esclude a vicenda con IMQ_IMPL_DISC_BACKOUT.

Su IBM MQ V7.0 e versioni successive, le applicazioni C++ che utilizzano una connessione implicita, devono specificare IMQ_IMPL_CONN insieme a qualsiasi altra opzione fornita nel metodo `setBehavior()` su un oggetto di classe `ImqQueueManager`. Se l'applicazione non utilizza il metodo `setBehavior()` per impostare esplicitamente le opzioni di comportamento, ad esempio,

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

questa modifica non influisce sull'utente poiché MQ_IMPL_CONN è abilitato per impostazione predefinita.

Se l'applicazione imposta esplicitamente le opzioni di comportamento, ad esempio,

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

è necessario includere IMQ_IMPL_CONN nel metodo `setBehavior()` nel modo seguente, per consentire all'applicazione di completare una connessione implicita:

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_CONN | IMQ_IMPL_DISC_COMMIT)
```

Attributi oggetto

sovrascrittura connessioni di account
Consente alle applicazioni di sovrascrivere l'impostazione dei valori values.This è di sola lettura.

Intervallo account
Il tempo prima che vengano scritti i record di account intermedi (in secondi). Questo attributo è di sola lettura.

registrazione attività
Controlla la generazione dei prospetti delle attività. Questo attributo è di sola lettura.

Utilizza nuova verifica MCA
Gli elementi controllati per determinare se un MCA deve essere adottato quando viene rilevato un nuovo canale in entrata che ha lo stesso nome di un MCA già attivo. Questo attributo è di sola lettura.

Adotta nuovo tipo MCA
Indica se un'istanza orfana di un MCA di un particolare tipo di canale deve essere riavviata automaticamente quando viene rilevata una nuova richiesta di canale in entrata che corrisponde ai parametri di adozione del nuovo controllo mca. Questo attributo è di sola lettura.

tipo autenticazione
Indica il tipo di autenticazione che viene eseguita.

evento autorizzazione
Controlla gli eventi di autorizzazione. Questo attributo è di sola lettura.

Opzioni di inizio
Le opzioni che si applicano al metodo begin. Il valore iniziale è MQBO_NONE.

evento bridge
Se vengono generati eventi bridge IMS . Questo attributo è di sola lettura.

Definizione automatica canale
Valore di definizione automatica del canale. Questo attributo è di sola lettura.

evento di definizione automatica del canale
Valore dell'evento di definizione automatica del canale. Questo attributo è di sola lettura.

Uscita definizione automatica canale
Nome uscita di definizione automatica del canale. Questo attributo è di sola lettura.

evento del canale
Indica se vengono generati eventi del canale. Questo attributo è di sola lettura.

Adattatori dell'iniziatore di canali
Il numero di attività secondarie dell'adattatore da utilizzare per elaborare le chiamate IBM MQ . Questo attributo è di sola lettura.

Controllo programma di avvio canale
Indica se l'iniziatore di canali deve essere avviato automaticamente all'avvio del gestore code. Questo attributo è di sola lettura.

Dispatcher dell'iniziatore di canali
Il numero di dispatcher da utilizzare per l'iniziatore di canali. Questo attributo è di sola lettura.

avvio automatico traccia iniziatore di canale
Indica se la traccia dell'iniziatore di canali deve essere avviata automaticamente o meno. Questo attributo è di sola lettura.

Dimensione tabella di traccia dell'iniziatore di canali
La dimensione dello spazio dati di traccia dell'iniziatore di canali (in MB). Questo attributo è di sola lettura.

Controllo canale
Controlla la raccolta dei dati di controllo online per i canali. Questo attributo è di sola lettura.

riferimento canale
Un riferimento a una definizione di canale da utilizzare durante la connessione client. Durante la connessione, questo attributo può essere impostato su null, ma non può essere modificato in un altro valore. Il valore iniziale è null.

Statistiche canale
Controlla la raccolta dei dati statistici per i canali. Questo attributo è di sola lettura.

character set
CCSID (Coded Character Set Identifier). Questo attributo è di sola lettura.

Controllo mittente cluster
Controlla la raccolta dei dati di controllo online per i canali mittenti del cluster definiti automaticamente. Questo attributo è di sola lettura.

Informazioni mittente cluster
Controlla la raccolta dei dati statistici per i canali mittenti del cluster definiti automaticamente. Questo attributo è di sola lettura.

Dati carico di lavoro cluster
Dati di uscita carico di lavoro cluster. Questo attributo è di sola lettura.

Uscita carico di lavoro cluster
Nome uscita Cluster Workload. Questo attributo è di sola lettura.

Lunghezza carico di lavoro cluster
Lunghezza carico di lavoro cluster. Questo attributo è di sola lettura.

mru carico di lavoro cluster
Valore dei canali utilizzati più di recente del carico di lavoro del cluster. Questo attributo è di sola lettura.

Coda utilizzo carico di lavoro cluster
Il valore della coda di utilizzo del carico di lavoro del cluster. Questo attributo è di sola lettura.

evento di comando
Se vengono generati eventi di comando. Questo attributo è di sola lettura.

Nome coda di input comandi
Il nome della coda di immissione del comando di sistema. Questo attributo è di sola lettura.

livello comando
Livello di comando supportato dal gestore code. Questo attributo è di sola lettura.

Controllo server di comandi
Indica se il server dei comandi deve essere avviato automaticamente all'avvio del gestore code. Questo attributo è di sola lettura.

Opzioni di connessione
Opzioni che si applicano al metodo di connessione. Il valore iniziale è MQCNO_NONE. I seguenti valori aggiuntivi possono essere possibili, a seconda della piattaforma:

- MQCNO_STANDARD_BINDING
- MQCNO_FASTPATH_BINDING
- MQCNO_HANDLE_SHARE_NONE
- MQCNO_HANDLE_SHARE_BLOCK
- MQCNO_HANDLE_SHARE_NO_BLOCK
- MQCNO_SERIALIZE_CONN_TAG_Q_MGR
- MQCNO_SERIALIZE_CONN_TAG_QSG
- MQCNO_RESTRICT_CONN_TAG_Q_MGR
- MQCNO_RESTRICT_CONN_TAG_QSG

id connessione
Un identificativo univoco che consente a MQ di identificare in modo affidabile un'applicazione.

stato della connessione
TRUE quando si è connessi al gestore code. Questo attributo è di sola lettura.

tag di connessione
Un tag da associare a una connessione. Questo attributo può essere impostato solo quando non è connesso. Il valore iniziale è null.

hardware di crittografia
Dettagli di configurazione per hardware crittografico. Per connessioni client MQ MQI.

nome coda di messaggi non instradabili
Nome della coda di messaggi non recapitabili. Questo attributo è di sola lettura.

nome coda di trasmissione predefinito
Il nome della coda di trasmissione predefinita. Questo attributo è di sola lettura.

elenchi di distribuzione
Capacità del gestore code di supportare gli elenchi di distribuzione.

gruppo dns
Il nome del gruppo a cui il listener TCP che gestisce le trasmissioni in entrata per il gruppo di condivisione code deve unirsi quando si utilizza il supporto DDNS (Dynamic Domain Name Services) di Workload Manager. Questo attributo è di sola lettura.

wlm dns
Se il listener TCP che gestisce le trasmissioni in entrata per il gruppo di condivisione code deve registrarsi con Workload Manager per DDNS (Dynamic Domain Name Services). Questo attributo è di sola lettura.

primo record di autenticazione
Il primo di uno o più oggetti della classe ImqAuthenticationRecord, in nessun ordine particolare, in cui la connessione record ImqAuthenticationfa riferimento a questo oggetto. Per connessioni client MQ MQI.

primo oggetto gestito
Il primo di uno o più oggetti della classe ImqObject, in nessun ordine particolare, in cui la connessione ImqObject fa riferimento a questo oggetto. Il valore iniziale è zero.

evento di inibizione
Controlla gli eventi di inibizione. Questo attributo è di sola lettura.

Versione indirizzo IP
Quale protocollo IP (IPv4 o IPv6) utilizzare per una connessione di canale. Questo attributo è di sola lettura.

repository delle chiavi
Ubicazione del file database di chiavi in cui sono memorizzate le chiavi e i certificati. Per connessioni IBM MQ MQI client .

conteggio reimpostazioni chiave
Il numero di byte non codificati inviati e ricevuti all'interno di una conversazione TLS prima che la chiave segreta venga rinegoziata. Questo attributo si applica solo a connessioni client che utilizzano MQCONNX. Vedere anche conteggio reimpostazioni chiave ssl.

Timer listener
L'intervallo di tempo (in secondi) tra i tentativi da parte di IBM MQ di riavviare il listener se si è verificato un errore APPC o TCP/IP. Questo attributo è di sola lettura.

evento locale
Controlla gli eventi locali. Questo attributo è di sola lettura.

evento logger
Controlla se vengono generati eventi del log di ripristino. Questo attributo è di sola lettura.

Nome gruppo LU
Il nome LU generico che il listener LU 6.2 che gestisce le trasmissioni in entrata per il gruppo di condivisione code deve utilizzare. Questo attributo è di sola lettura.

Nome LU
Il nome della LU da utilizzare per le trasmissioni LU in uscita 6.2 . Questo attributo è di sola lettura.

Suffisso braccio lu62
Il suffisso di SYS1.PARMLIB membro APPCPMxx, che designa LUADD per questo iniziatore di canali. Questo attributo è di sola lettura.

lu62 canali
Il numero massimo di canali che possono essere correnti o client che possono essere connessi, che utilizzano il protocollo di trasmissione LU 6.2 . Questo attributo è di sola lettura.

numero massimo canali attivi
Indica il numero massimo di canali che possono essere attivi contemporaneamente. Questo attributo è di sola lettura.

Numero massimo di canali
Il numero massimo di canali che possono essere correnti (compresi i canali di connessione server con i client connessi). Questo attributo è di sola lettura.

numero massimo di handle
Numero massimo di puntatori. Questo attributo è di sola lettura.

Lunghezza massima dei messaggi
Lunghezza massima possibile per qualsiasi messaggio su qualsiasi coda gestita da questo gestore code. Questo attributo è di sola lettura.

Priorità massima
Priorità massima del messaggio. Questo attributo è di sola lettura.

Num. mass. mess. non sincronizzati
Numero massimo di messaggi di cui non è stato eseguito il commit all'interno di un'unità o di un lavoro. Questo attributo è di sola lettura.

Account MQI
Controlla la raccolta delle informazioni di account per i dati MQI. Questo attributo è di sola lettura.

Statistiche MQI
Controlla la raccolta delle informazioni di controllo per il gestore code. Questo attributo è di sola lettura.

numero massimo di porte in uscita
L'estremità superiore dell'intervallo di numeri di porta da utilizzare quando si collegano i canali in uscita. Questo attributo è di sola lettura.

minimo porta in uscita
L'estremità inferiore dell'intervallo di numeri di porta da utilizzare durante il bind dei canali in uscita. Questo attributo è di sola lettura.

password
password associata all'ID utente

evento delle prestazioni
Controlla gli eventi delle prestazioni. Questo attributo è di sola lettura.

enterprise
Piattaforma su cui risiede il gestore code. Questo attributo è di sola lettura.

Conto coda
Controlla la raccolta delle informazioni di account per le code. Questo attributo è di sola lettura.

Controllo coda
Controlla la raccolta dei dati di controllo online per le code. Questo attributo è di sola lettura.

Informazioni coda
Controlla la raccolta dei dati statistici per le code. Questo attributo è di sola lettura.

timeout di ricezione
Circa il tempo di attesa di un canale di messaggi TCP/IP per ricevere i dati, inclusi gli heartbeat, dal relativo partner, prima di tornare allo stato inattivo. Questo attributo è di sola lettura.

minimo timeout di ricezione
Il tempo minimo di attesa di un canale TCP/IP per ricevere i dati, inclusi gli heartbeat, dal partner, prima di ritornare allo stato inattivo. Questo attributo è di sola lettura.

Tipo di timeout di ricezione
Un qualificatore applicato per il timeout di ricezione. Questo attributo è di sola lettura.

evento remoto
Controlla gli eventi remoti. Questo attributo è di sola lettura.

nome repository
Nome del repository. Questo attributo è di sola lettura.

Elenco nomi repository
Nome elenco nomi repository. Questo attributo è di sola lettura.

nome gestore code condiviso
Indica se le MQOPEN di una coda condivisa in cui il nome ObjectQMgrè un altro gestore code nel gruppo di condivisione code devono risolversi in un'apertura della coda condivisa sul gestore code locale. Questo attributo è di sola lettura.

evento ssl
Se vengono generati eventi SSL. Questo attributo è di sola lettura.

SSL FIPS richiesto
Se devono essere utilizzati solo algoritmi certificati FIPS se la crittografia viene eseguita nel software IBM MQ . Questo attributo è di sola lettura.

Conteggio reimpostazioni chiave SSL
Il numero di byte non crittografati inviati e ricevuti in una conversazione SSL prima che la chiave segreta venga rinegoziata. Questo attributo è di sola lettura.

evento start - stop
Controlla gli eventi di avvio. Questo attributo è di sola lettura.

intervallo statistico
La frequenza con cui i dati di monitoraggio delle statistiche vengono scritti nella coda di monitoraggio. Questo attributo è di sola lettura.

Disponibilità Syncpoint
Disponibilità della partecipazione al punto di sincronizzazione. Questo attributo è di sola lettura.

Nota: Le unità di lavoro globali coordinate dal gestore code non sono supportate sulla piattaforma IBM i . 

È possibile programmare un'unità di lavoro, coordinata esternamente da IBM i, utilizzando le chiamate di sistema native _Rcommit e _Rback. Avviare questo tipo di unità di lavoro avviando l'applicazione IBM MQ sotto il controllo di commit a livello di lavoro utilizzando il comando STRCMTCTL. Per ulteriori dettagli, consultare Interfacce al gestore del punto di sincronizzazione

esterno IBM i . Il backout e il commit sono supportati sulla piattaforma IBM i per le unità di lavoro locali coordinate da un gestore code.

canali tcp
Il numero massimo di canali che possono essere correnti o di client che possono essere connessi, che utilizzano il protocollo di trasmissione TCP/IP. Questo attributo è di sola lettura.

Keepalive TCP
Se la funzione TCP KEEPALIVE deve essere utilizzata per verificare che l'altra estremità della connessione sia ancora disponibile. Questo attributo è di sola lettura.

Nome TCP
Il nome del sistema TCP/IP unico o predefinito da utilizzare, in base al valore del tipo di stack tcp. Questo attributo è di sola lettura.

Tipo di stack TCP
Se all'iniziatore di canali è consentito utilizzare solo lo spazio di indirizzo TCP/IP specificato nel nome tcp oppure è possibile eseguire il bind a qualsiasi indirizzo TCP/IP selezionato. Questo attributo è di sola lettura.

Registrazione instradamento traccia
Controlla la registrazione delle informazioni di traccia del percorso. Questo attributo è di sola lettura.

Intervallo trigger
Intervallo trigger. Questo attributo è di sola lettura.

identificativo utente
Su piattaforme AIX and Linux , l'ID utente reale dell'applicazione. Su piattaforme Windows , l'ID utente dell'applicazione.

Costruttori

ImqQueue();
Il costruttore predefinito.

ImqQueue(gestore ImqQueue&);
Il costruttore di copia. Lo stato della connessione sarà FALSE.

ImqQueueManager (const char * *nome*);
Imposta il nome ImqObject su *name*.

Distruttori

Quando un oggetto gestore ImqQueueviene eliminato, viene automaticamente disconnesso.

Metodi di classe (pubblico)

comportamento statico MQLONG ();
Restituisce il comportamento.

void setBehavior(const MQLONG *comportamento* = 0);
Imposta il comportamento.

Metodi oggetto (pubblico)

operatore void = (const ImqQueueManager & *mgr*);
Se necessario, si disconnette e copia i dati di istanza da *mgr*. Lo stato della connessione è FALSE.

ImqBoolean accountingConnOverride (MQLONG & statint);
Fornisce una copia del valore di sovrascrittura delle connessioni di account. Restituisce TRUE in caso di esito positivo.

MQLONG accountingConnOverride ();
Restituisce il valore di sovrascrittura delle connessioni di account senza alcuna indicazione di possibili errori.

ImqBoolean accountingInterval (MQLONG & statint);
Fornisce una copia del valore dell'intervallo di account. Restituisce TRUE in caso di esito positivo.

MQLONG accountingInterval ();
Restituisce il valore dell'intervallo di account senza alcuna indicazione di possibili errori.

ImqBoolean activityRecording (MQLONG & rec);
Fornisce una copia del valore di registrazione dell'attività. Restituisce TRUE in caso di esito positivo.

MQLONG activityRecording ();
Restituisce il valore di registrazione dell'attività senza alcuna indicazione di possibili errori.

ImqBoolean adoptNewMCACheck (MQLONG & check);
Fornisce una copia del valore di adozione del nuovo controllo MCA. Restituisce TRUE in caso di esito positivo.

MQLONG adoptNewMCACheck ();
Restituisce il valore di adozione del nuovo controllo MCA senza alcuna indicazione di possibili errori.

ImqBoolean adoptNewMCAType (MQLONG & tipo);
Fornisce una copia del nuovo tipo di MCA adottato. Restituisce TRUE in caso di esito positivo.

MQLONG adoptNewMCAType ();
Restituisce l'adozione di un nuovo tipo MCA senza alcuna indicazione di possibili errori.

QLONG authenticationType () const;
Restituisce il tipo di autenticazione.

void setAuthenticationType (const MQLONG type = MQCSP_AUTH_NONE);
Imposta il tipo di autenticazione.

ImqBoolean authorityEvent(MQLONG & ***evento*** **);**
Fornisce una copia dello stato di abilitazione dell'evento di autorizzazione. Restituisce TRUE in caso di esito positivo.

MQLONG authorityEvent();
Restituisce lo stato di abilitazione dell'evento di autorizzazione senza alcuna indicazione di possibili errori.

backout ImqBoolean ();
Esegue il backout delle modifiche non sottoposte a commit. Restituisce TRUE in caso di esito positivo.

ImqBoolean begin ();
Inizia un'unità di lavoro. Le opzioni di inizio influiscono sul comportamento di questo metodo. Restituisce TRUE se l'operazione ha esito positivo, ma restituisce anche TRUE anche se la chiamata MQBEGIN sottostante restituisce MQRC_NO_EXTERNAL_PARTICIPANTS o MQRC_PARTICIPANT_NOT_AVAILABLE (entrambi associati a MQCC_WARNING).

MQLONG beginOptions() const;
Restituisce le opzioni iniziali.

void setBeginOptions (const MQLONG ***options*** **= MQBO_NONE);**
Imposta le opzioni di inizio.

ImqBoolean bridgeEvent (MQLONG & evento);
Fornisce una copia del valore dell'evento bridge. Restituisce TRUE in caso di esito positivo.

MQLONG bridgeEvent ();
Restituisce il valore dell'evento bridge senza alcuna indicazione di possibili errori.

ImqBoolean channelAutoDefinition (MQLONG & ***valore*** **);**
Fornisce una copia del valore di definizione automatica del canale. Restituisce TRUE in caso di esito positivo.

Definizione MQLONG channelAuto();
Restituisce il valore di definizione automatica del canale senza alcuna indicazione di possibili errori.

ImqBoolean channelAutoDefinitionEvent(MQLONG & ***valore*** **);**
Fornisce una copia del valore dell'evento di definizione automatica del canale. Restituisce TRUE in caso di esito positivo.

MQLONG channelAutoDefinitionEvent();
Restituisce il valore dell'evento di definizione automatica del canale senza alcuna indicazione di possibili errori.

ImqBoolean channelAutoDefinitionExit(ImqString &*nome*);
Fornisce una copia del nome dell'uscita di definizione automatica del canale. Restituisce TRUE in caso di esito positivo.

ImqString channelAutoDefinitionExit( );
Restituisce il nome dell'uscita di definizione automatica del canale senza alcuna indicazione di possibili errori.

ImqBoolean channelEvent (MQLONG & evento);
Fornisce una copia del valore evento del canale. Restituisce TRUE in caso di esito positivo.

MQLONG channelEvent();
Restituisce il valore dell'evento del canale senza alcuna indicazione di possibili errori.

MQLONG channelInitiatorAdapters ();
Restituisce il valore degli adattatori dell'iniziatore del canale senza alcuna indicazione di possibili errori.

Adattatori ImqBoolean channelInitiator(MQLONG & adapters);
Fornisce una copia del valore degli adattatori iniziatore di canale. Restituisce TRUE in caso di esito positivo.

MQLONG channelInitiatorControl ();
Restituisce il valore di avvio dell'iniziatore di canali senza alcuna indicazione di possibili errori.

ImqBoolean Controllo channelInitiator(MQLONG & init);
Fornisce una copia del valore di avvio del controllo dell'iniziatore di canali. Restituisce TRUE in caso di esito positivo.

MQLONG channelInitiatorDispatcher ();
Restituisce il valore dei dispatcher dell'iniziatore di canali senza alcuna indicazione di possibili errori.

ImqBoolean channelInitiatorDispatcher (MQLONG & dispatcher);
Fornisce una copia del valore dei dispatcher dell'iniziatore di canali. Restituisce TRUE in caso di esito positivo.

MQLONG channelInitiatorTraceAutoStart ();
Restituisce il valore di avvio automatico della traccia dell'iniziatore del canale senza alcuna indicazione di possibili errori.

ImqBoolean channelInitiatorTraceAutoStart (MQLONG & auto);
Fornisce una copia del valore di avvio automatico della traccia dell'iniziatore di canali. Restituisce TRUE in caso di esito positivo.

MQLONG channelInitiatorTraceTableDimensione ();
Restituisce il valore della dimensione della tabella di traccia dell'iniziatore del canale senza alcuna indicazione di possibili errori.

ImqBoolean channelInitiatorTraceTableDimensione (MQLONG & size);
Fornisce una copia del valore della dimensione della tabella di traccia dell'iniziatore del canale. Restituisce TRUE in caso di esito positivo.

ImqBoolean channelMonitoring (MQLONG & monchl);
Fornisce una copia del valore di monitoraggio del canale. Restituisce TRUE in caso di esito positivo.

MQLONG channelMonitoring ();
Restituisce il valore di monitoraggio del canale senza alcuna indicazione di possibili errori.

ImqBoolean channelReference(ImqChannel * &*pchannel*);
Fornisce una copia del riferimento del canale. Se il riferimento del canale non Š valido, imposta *pchannel* su null. Questo metodo restituisce TRUE se ha esito positivo.

ImqChannel * channelReference( );
Restituisce il riferimento del canale senza alcuna indicazione di possibili errori.

ImqBoolean setChannelRiferimento (ImqChannel & *channel*);
Imposta il riferimento del canale. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setChannelRiferimento (ImqChannel * *canale* = 0);
Imposta o reimposta il riferimento del canale. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean channelStatistics (MQLONG e statchl);
Fornisce una copia del valore delle statistiche del canale. Restituisce TRUE in caso di esito positivo.

MQLONG channelStatistics ();
Restituisce il valore delle statistiche del canale senza alcuna indicazione di possibili errori.

ImqBoolean characterSet(MQLONG & *ccsid*);
Fornisce una copia della serie di caratteri. Restituisce TRUE in caso di esito positivo.

MQLONG characterSet();
Restituisce una copia della serie di caratteri, senza alcuna indicazione di possibili errori.

MQLONG clientSslKeyResetConteggio () const;
Restituisce il valore del conteggio di ripristino della chiave SSL utilizzato sulle connessioni client.

void setClientSslKeyResetCount(const MQLONG count);
Imposta il conteggio di ripristino della chiave SSL utilizzato sulle connessioni client.

ImqBoolean clusterSenderMonitoring (MQLONG & monacls);
Fornisce una copia del valore predefinito di monitoraggio del mittente del cluster. Restituisce TRUE in caso di esito positivo.

Monitoraggio MQLONG clusterSender();
Restituisce il valore predefinito di monitoraggio del mittente del cluster senza alcuna indicazione di possibili errori.

ImqBoolean clusterSenderStatistics (MQLONG & statacls);
Fornisce una copia del valore delle statistiche del mittente del cluster. Restituisce TRUE in caso di esito positivo.

Statistiche MQLONG clusterSender();
Restituisce il valore delle statistiche del mittente del cluster senza alcuna indicazione di possibili errori.

ImqBoolean clusterWorkloadData (ImqString & *data*);
Fornisce una copia dei dati di uscita del carico di lavoro del cluster. Restituisce TRUE in caso di esito positivo.

ImqString clusterWorkloadDati ();
Restituisce i dati di uscita del carico di lavoro del cluster senza alcuna indicazione di possibili errori.

ImqBoolean clusterWorkloadExit (ImqString & *name*);
Fornisce una copia del nome dell'uscita del carico di lavoro del cluster. Restituisce TRUE in caso di esito positivo.

ImqString clusterWorkloadExit ();
Restituisce il nome dell'uscita del carico di lavoro del cluster senza alcuna indicazione di possibili errori.

ImqBoolean clusterWorkloadLunghezza (MQLONG & *lunghezza*);
Fornisce una copia della lunghezza del carico di lavoro del cluster. Restituisce TRUE in caso di esito positivo.

MQLONG clusterWorkloadLunghezza ();
Restituisce la lunghezza del carico di lavoro del cluster senza alcuna indicazione di possibili errori.

ImqBoolean clusterWorkLoadMRU (MQLONG & mru);
Fornisce una copia del valore dei canali utilizzati più di recente del carico di lavoro del cluster. Restituisce TRUE in caso di esito positivo.

MQLONG clusterWorkLoadMRU ();
Restituisce il valore dei canali utilizzati più di recente del carico di lavoro del cluster senza alcuna indicazione di possibili errori.

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);
Fornisce una copia del valore coda di utilizzo del carico di lavoro cluster. Restituisce TRUE in caso di esito positivo.

MQLONG clusterWorkLoadUseQ ();
Restituisce il valore della coda di utilizzo del carico di lavoro del cluster senza alcuna indicazione di possibili errori.

ImqBoolean commandEvent (MQLONG & evento);
Fornisce una copia del valore evento del comando. Restituisce TRUE in caso di esito positivo.

MQLONG commandEvent ();
Restituisce il valore dell'evento comando senza alcuna indicazione di possibili errori.

ImqBoolean commandInputQueueName(ImqString &*nome*);
Fornisce una copia del nome della coda di immissione comandi. Restituisce TRUE in caso di esito positivo.

ImqString commandInputQueueName( );
Restituisce il nome della coda di input del comando senza alcuna indicazione di possibili errori.

ImqBoolean commandLevel(MQLONG &*livello*);
Fornisce una copia del livello di comando. Restituisce TRUE in caso di esito positivo.

MQLONG commandLevel();
Restituisce il livello di comando senza alcuna indicazione di possibili errori.

MQLONG commandServerControl ();
Restituisce il valore di avvio del server dei comandi senza alcuna indicazione di possibili errori.

Controllo ImqBoolean commandServer(MQLONG & server);
Fornisce una copia del valore di avvio del controllo del server dei comandi. Restituisce TRUE in caso di esito positivo.

ImqBoolean commit ();
Esegue il commit delle modifiche non sottoposte a commit. Restituisce TRUE in caso di esito positivo.

ImqBoolean connect ();
Si connette al gestore code con il nome ImqObject fornito, il valore predefinito è il gestore code locale. Se si desidera connettersi a uno specifico gestore code, utilizzare il metodo ImqObject setName prima della connessione. Se è presente un riferimento del canale, viene utilizzato per passare le informazioni sulla definizione del canale a MQCONNX in un MQCD. ChannelType in MQCD è impostato su MQCHT_CLNTCONN. le informazioni di riferimento del canale, che sono significative solo per le connessioni client, vengono ignorate per le connessioni server. Le opzioni di connessione influenzano il comportamento di questo metodo. Questo metodo imposta lo stato della connessione su TRUE se l'operazione ha esito positivo. Restituisce lo stato della nuova connessione.

Se è presente un primo record di autenticazione, la catena di record di autenticazione viene utilizzata per autenticare i certificati digitali per i canali client sicuri.

È possibile connettere più di un oggetto ImqQueueallo stesso gestore code. Tutti utilizzano lo stesso handle di connessione MQHCONN e condividono la funzione UOW per la connessione associata al thread. Il primo gestore ImqQueuea connettersi ottiene l'handle MQHCONN. L'ultimo gestore ImqQueueda disconnettere esegue MQDISC.

Per un programma a più sottoprocessi, si consiglia di utilizzare un oggetto ImqQueueManager separato per ciascun thread.

ImqBinary connectionId () const;
Restituisce l'ID connessione.

ImqBinary connectionTag () const;
Restituisce la tag di connessione.

ImqBoolean setConnectionTag (const MQBYTE128 *tag* = 0);
Imposta la tag di connessione. Se *tag* è zero, elimina la tag di connessione. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setConnectionTag (const ImqBinary e *tag*);
Imposta la tag di connessione. La lunghezza dati di *tag* deve essere zero (per cancellare la tag di connessione) o MQ_CONN_TAG_LENGTH. Questo metodo restituisce TRUE se ha esito positivo.

MQLONG connectOptions() const;
Restituisce le opzioni di connessione.

void setConnectOpzioni (const MQLONG *opzioni* = MQCNO_NONE);
Imposta le opzioni di collegamento.

ImqBoolean connectionStatus() const;
Restituisce lo stato della connessione.

ImqString cryptographicHardware ( );
Restituisce l'hardware crittografico.

ImqBoolean setCryptographicHardware (const char * *hardware* = 0);
Imposta l'hardware crittografico. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean deadLetterQueueName(ImqString & *nome*);
Fornisce una copia del nome della coda di messaggi non recapitabili. Restituisce TRUE in caso di esito positivo.

ImqString deadLetterQueueName( );
Restituisce una copia del nome della coda di messaggi non recapitabili, senza alcuna indicazione di possibili errori.

ImqBoolean defaultTransmissionQueueName(ImqString & *nome*);
Fornisce una copia del nome della coda di trasmissione predefinita. Restituisce TRUE in caso di esito positivo.

ImqString defaultTransmissionQueueName( );
Restituisce il nome della coda di trasmissione predefinito senza alcuna indicazione di possibili errori.

ImqBoolean disconnect ();
Si disconnette dal gestore code e imposta lo stato della connessione su FALSE. Chiude tutti gli oggetti ImqProcess e ImqQueue associati a questo oggetto e interrompe il riferimento della connessione prima della disconnessione. Se più di un oggetto ImqQueueManager è connesso allo stesso gestore code, solo l'ultimo a disconnettersi esegue una disconnessione fisica; altri eseguono una disconnessione logica. Le modifiche non sottoposte a commit vengono sottoposte a commit solo in seguito alla disconnessione fisica.

Questo metodo restituisce TRUE se ha esito positivo. Se viene richiamato quando non esiste alcuna connessione, anche il codice di ritorno è true.

ImqBoolean distributionLists(supporto MQLONG &);
Fornisce una copia del valore degli elenchi di distribuzione. Restituisce TRUE in caso di esito positivo.

MQLONG distributionLists();
Restituisce il valore degli elenchi di distribuzione senza alcuna indicazione di possibili errori.

ImqBoolean dnsGroup (ImqString & gruppo);
Fornisce una copia del nome del gruppo DNS. Restituisce TRUE in caso di esito positivo.

ImqString dnsGroup ( );
Restituisce il nome del gruppo DNS senza alcuna indicazione di possibili errori.

ImqBoolean dnsWlm (MQLONG & wlm);
Fornisce una copia del valore WLM DNS. Restituisce TRUE in caso di esito positivo.

MQLONG dnsWlm ();
Restituisce il valore WLM DNS senza alcuna indicazione di possibili errori.

ImqAuthenticationRecord * firstAuthenticationRecord () const;
Restituisce il primo record di autenticazione.

void setFirstAuthenticationRecord (const ImqAuthenticationRecord * *air* = 0);
Imposta il primo record di autenticazione.

ImqObject * firstManagedObject () const;
Restituisce il primo oggetto gestito.

ImqBoolean inhibitEvent(evento MQLONG &);
Fornisce una copia dello stato di abilitazione dell'evento inibito. Restituisce TRUE in caso di esito positivo.

MQLONG inhibitEvent();
Restituisce lo stato di abilitazione dell'evento di inibizione senza alcuna indicazione di possibili errori.

ImqBoolean ipAddressVersione (MQLONG & versione);
Fornisce una copia del valore della versione dell'indirizzo IP. Restituisce TRUE in caso di esito positivo.

MQLONG ipAddressVersione ();
Restituisce il valore della versione dell'indirizzo IP senza alcuna indicazione di possibili errori.

ImqBoolean keepAlive (MQLONG & keepalive);
Fornisce una copia del valore keep alive. Restituisce TRUE in caso di esito positivo.

MQLONG keepAlive ();
Restituisce il valore keep alive senza alcuna indicazione di possibili errori.

ImqString keyRepository ( );
Restituisce il repository delle chiavi.

ImqBoolean setKeyRepository (const char * *repository* = 0);
Imposta il repository chiavi. Restituisce TRUE in caso di esito positivo.

ImqBoolean listenerTimer (MQLONG & timer);
Fornisce una copia del valore del timer del listener. Restituisce TRUE in caso di esito positivo.

MQLONG listenerTimer ();
Restituisce il valore del timer del listener senza alcuna indicazione di possibili errori.

ImqBoolean localEvent(evento MQLONG &);
Fornisce una copia dello stato di abilitazione dell'evento locale. Restituisce TRUE in caso di esito positivo.

MQLONG localEvent();
Restituisce lo stato di abilitazione dell'evento locale senza alcuna indicazione di possibili errori.

ImqBoolean loggerEvent (MQLONG & conteggio);
Fornisce una copia del valore dell'evento del programma di registrazione. Restituisce TRUE in caso di esito positivo.

MQLONG loggerEvent ();
Restituisce il valore dell'evento del logger senza alcuna indicazione di possibili errori.

ImqBoolean luGroupNome (ImqString & name);.
Fornisce una copia del nome del gruppo LU. Restituisce TRUE se l'operazione ha esito positivo

ImqString luGroupNome ();
Restituisce il nome del gruppo LU senza alcuna indicazione di possibili errori.

ImqBoolean lu62ARMSuffix (ImqString & suffix);
Fornisce una copia del suffisso ARM LU62 . Restituisce TRUE in caso di esito positivo.

ImqString lu62ARMSuffix ( );
Restituisce il suffisso ARM LU62 senza alcuna indicazione di possibili errori

ImqBoolean luName (ImqString & nome);
Fornisce una copia del nome LU. Restituisce TRUE in caso di esito positivo.

ImqString luName ( );
Restituisce il nome LU senza alcuna indicazione di possibili errori.

ImqBoolean maximumActiveCanali (MQLONG & canali);
Fornisce una copia del numero massimo di canali attivi. Restituisce TRUE in caso di esito positivo.

Canali MQLONG maximumActive();
Restituisce il valore massimo dei canali attivi senza alcuna indicazione di possibili errori.

ImqBoolean maximumCurrentcanali (MQLONG & canali);
Fornisce una copia del valore massimo dei canali correnti. Restituisce TRUE in caso di esito positivo.

MQLONG maximumCurrentcanali ();
Restituisce il valore massimo dei canali correnti senza alcuna indicazione di possibili errori.

ImqBoolean maximumHandles(MQLONG & ***numero*** **);**
Fornisce una copia del numero massimo di handle. Restituisce TRUE in caso di esito positivo.

MQLONG maximumHandles();
Restituisce il numero massimo di handle senza alcuna indicazione di possibili errori.

ImqBoolean maximumLu62Channels (MQLONG e canali);
Fornisce una copia del valore massimo dei canali LU62 . Restituisce TRUE in caso di esito positivo.

MQLONG maximumLu62Channels ();.
Restituisce il valore massimo dei canali LU62 senza alcuna indicazione di possibili errori

ImqBoolean maximumMessageLunghezza (MQLONG & ***lunghezza*** **);**
Fornisce una copia della lunghezza massima del messaggio. Restituisce TRUE in caso di esito positivo.

MQLONG maximumMessageLunghezza ();
Restituisce la lunghezza massima del messaggio senza alcuna indicazione di possibili errori.

ImqBoolean maximumPriority(MQLONG & ***priorità*** **);**
Fornisce una copia della priorità massima. Restituisce TRUE in caso di esito positivo.

MQLONG maximumPriority();
Restituisce una copia della priorità massima, senza alcuna indicazione di possibili errori.

ImqBoolean maximumTcpCanali (MQLONG & canali);
Fornisce una copia del valore massimo dei canali TCP. Restituisce TRUE in caso di esito positivo.

Canali MQLONG maximumTcp();
Restituisce il valore massimo dei canali TCP senza alcuna indicazione di possibili errori.

ImqBoolean maximumUncommittedMessaggi (MQLONG & ***numero*** **);**
Fornisce una copia del numero massimo di messaggi di cui non è stato eseguito il commit. Restituisce TRUE in caso di esito positivo.

Messaggi MQLONG maximumUncommitted();
Restituisce il numero massimo di messaggi senza commit senza alcuna indicazione di possibili errori.

ImqBoolean mqiAccounting (MQLONG & statint);
Fornisce una copia del valore di account MQI. Restituisce TRUE in caso di esito positivo.

MQLONG mqiAccounting ();
Restituisce il valore di account MQI senza alcuna indicazione di possibili errori.

ImqBoolean mqiStatistics (MQLONG & statmqi);
Fornisce una copia del valore delle statistiche MQI. Restituisce TRUE in caso di esito positivo.

MQLONG mqiStatistics ();
Restituisce il valore delle statistiche MQI senza alcuna indicazione di possibili errori.

ImqBoolean outboundPortMax (MQLONG & max);
Fornisce una copia del valore massimo della porta in uscita. Restituisce TRUE in caso di esito positivo.

MQLONG outboundPortMax ();
Restituisce il valore massimo della porta in uscita senza alcuna indicazione di possibili errori.

ImqBoolean outboundPortMin (MQLONG & min);
Fornisce una copia del valore minimo della porta in uscita. Restituisce TRUE in caso di esito positivo.

MQLONG outboundPortMin ();
Restituisce il valore minimo della porta in uscita senza alcuna indicazione di possibili errori.

Password ImqBinary () const;
Restituisce la password utilizzata sulle connessioni client.

ImqBoolean setPassword (const ImqString & password);
Imposta la password utilizzata sulle connessioni client.

ImqBoolean setPassword (const char * = 0 password);
Imposta la password utilizzata sulle connessioni client.

ImqBoolean setPassword (const ImqBinary & password);
Imposta la password utilizzata sulle connessioni client.

ImqBoolean performanceEvent(MQLONG &*evento*);
Fornisce una copia dello stato di abilitazione dell'evento prestazioni. Restituisce TRUE in caso di esito positivo.

MQLONG performanceEvent();
Restituisce lo stato di abilitazione dell'evento prestazioni senza alcuna indicazione di possibili errori.

Piattaforma ImqBoolean (MQLONG &*platform*);
Fornisce una copia della piattaforma. Restituisce TRUE in caso di esito positivo.

Piattaforma MQLONG ();
Restituisce la piattaforma senza alcuna indicazione di possibili errori.

ImqBoolean queueAccounting (MQLONG & acctq);
Fornisce una copia del valore di account coda. Restituisce TRUE in caso di esito positivo.

MQLONG queueAccounting ();
Restituisce il valore di account della coda senza alcuna indicazione di possibili errori.

ImqBoolean queueMonitoring (MQLONG & monq);
Fornisce una copia del valore di controllo della coda. Restituisce TRUE in caso di esito positivo.

MQLONG queueMonitoring ();
Restituisce il valore di monitoraggio della coda senza alcuna indicazione di possibili errori.

ImqBoolean queueStatistics (MQLONG & statq);
Fornisce una copia del valore delle statistiche della coda. Restituisce TRUE in caso di esito positivo.

MQLONG queueStatistics ();
Restituisce il valore delle statistiche della coda senza alcuna indicazione di possibili errori.

ImqBoolean receiveTimeout (MQLONG & timeout);
Fornisce una copia del valore di timeout di ricezione. Restituisce TRUE in caso di esito positivo.

MQLONG receiveTimeout ();
Restituisce il valore di timeout di ricezione senza alcuna indicazione di possibili errori.

ImqBoolean receiveTimeoutMin (MQLONG & min);
Fornisce una copia del valore di timeout di ricezione minimo. Restituisce TRUE in caso di esito positivo.

MQLONG receiveTimeoutMin ();
Restituisce il valore minimo di timeout di ricezione senza alcuna indicazione di possibili errori.

ImqBoolean receiveTimeoutTipo (MQLONG & type);
Fornisce una copia del tipo di timeout di ricezione. Restituisce TRUE in caso di esito positivo.

MQLONG receiveTimeoutTipo ();
Restituisce il tipo di timeout di ricezione senza alcuna indicazione di possibili errori.

ImqBoolean remoteEvent(evento MQLONG &);
Fornisce una copia dello stato di abilitazione dell'evento remoto. Restituisce TRUE in caso di esito positivo.

MQLONG remoteEvent();
Restituisce lo stato di abilitazione dell'evento remoto senza alcuna indicazione di possibili errori.

ImqBoolean repositoryName(ImqString &*nome*);
Fornisce una copia del nome repository. Restituisce TRUE in caso di esito positivo.

ImqString repositoryName( );
Restituisce il nome del repository senza alcuna indicazione di possibili errori.

ImqBoolean repositoryNamelistNome (ImqString &*name*);
Fornisce una copia del nome dell'elenco nomi del repository. Restituisce TRUE in caso di esito positivo.

ImqString repositoryNamelistNome ();
Restituisce una copia del nome dell'elenco nomi del repository senza alcuna indicazione di possibili errori.

ImqBoolean sharedQueueQueueManagerNome (MQLONG & name);
Fornisce una copia del valore del nome del gestore code condiviso. Restituisce TRUE in caso di esito positivo.

MQLONG sharedQueueQueueManagerName ();
Restituisce il valore del nome del gestore code condiviso senza alcuna indicazione di possibili errori.

ImqBoolean sslEvent (MQLONG & evento);
Fornisce una copia del valore evento SSL. Restituisce TRUE in caso di esito positivo.

MQLONG sslEvent ();
Restituisce il valore dell'evento SSL senza alcuna indicazione di possibili errori.

ImqBoolean sslFips (MQLONG & sslfips);
Fornisce una copia del valore FIPS SSL. Restituisce TRUE in caso di esito positivo.

MQLONG sslFips ();
Restituisce il valore FIPS SSL senza alcuna indicazione di possibili errori.

ImqBoolean sslKeyResetCount (MQLONG & count);
Fornisce una copia del valore del conteggio di reimpostazione della chiave SSL. Restituisce TRUE in caso di esito positivo.

MQLONG sslKeyResetCount ();
Restituisce il valore del conteggio di reimpostazione della chiave SSL senza alcuna indicazione di possibili errori.

ImqBoolean startStopEvent (evento MQLONG &);
Fornisce una copia dello stato di abilitazione dell'evento di avvio arresto. Restituisce TRUE in caso di esito positivo.

Evento MQLONG startStop();
Restituisce lo stato di abilitazione dell'evento di avvio senza alcuna indicazione di possibili errori.

ImqBoolean statisticsInterval (MQLONG & statint);
Fornisce una copia del valore di intervallo delle statistiche. Restituisce TRUE in caso di esito positivo.

MQLONG statisticsInterval ();
Restituisce il valore dell'intervallo delle statistiche senza alcuna indicazione di possibili errori.

ImqBoolean syncPointDisponibilità (MQLONG & *sync*);
Fornisce una copia del valore di disponibilità del punto di sincronizzazione. Restituisce TRUE in caso di esito positivo.

Disponibilità MQLONG syncPoint();
Restituisce una copia del valore di disponibilità del punto di sincronizzazione, senza alcuna indicazione di possibili errori.

ImqBoolean tcpName (ImqString & nome);
Fornisce una copia del nome del sistema TCP. Restituisce TRUE in caso di esito positivo.

ImqString tcpName ( );
Restituisce il nome del sistema TCP senza alcuna indicazione di possibili errori.

ImqBoolean tcpStackTipo (MQLONG & type);
Fornisce una copia del tipo di stack TCP. Restituisce TRUE in caso di esito positivo.

MQLONG tcpStackTipo ();
Restituisce il tipo di stack TCP senza alcuna indicazione di possibili errori.

ImqBoolean traceRouteRecording (MQLONG & routerec);
Fornisce una copia del valore di registrazione dell'instradamento traccia. Restituisce TRUE in caso di esito positivo.

MQLONG traceRouteRecording ();
Restituisce il valore di registrazione dell'instradamento di traccia senza alcuna indicazione di possibili errori.

ImqBoolean triggerInterval(MQLONG & *intervallo*);
Fornisce una copia dell'intervallo del trigger. Restituisce TRUE in caso di esito positivo.

MQLONG triggerInterval();
Restituisce l'intervallo di trigger senza alcuna indicazione di possibili errori.

ImqBinary userId () const;
Restituisce l'ID utente utilizzato sulle connessioni client.

ImqBoolean setUserId (const ImqString & id);
Imposta l'ID utente utilizzato sulle connessioni client.

ImqBoolean setUserId (const char * = 0 id);
Imposta l'ID utente utilizzato sulle connessioni client.

ImqBoolean setUserId (const ImqBinary & id);
Imposta l'ID utente utilizzato sulle connessioni client.

Metodi oggetto (protetti)

void setFirstManagedObject (const ImqObject * *object* = 0);
Imposta il primo oggetto gestito.

Dati oggetto (protetti)

MQHCONN *ohconn*
L'handle di connessione IBM MQ (significativo solo quando lo stato della connessione è TRUE).

Codici di origine

- MQRC_ATTRIBUTE_LOCKED
- ERRORE MQRC_ENVIRONMENT_ERROR
- MQRC_FUNZIONE_NON_SUPPORTATA
- ERRORE MQRC_REFERENCE_ERROR
- (codici di origine errore per MQBACK)
- (codici motivo per MQBEGIN)
- (codici motivo per MQCMIT)
- (codici motivo per MQCONNX)
- (codici motivo per MQDISC)
- (codici motivo per MQCONN)

Classe C++ intestazione ImqReference

Questa classe incapsula le funzioni della struttura dati MQRMH.

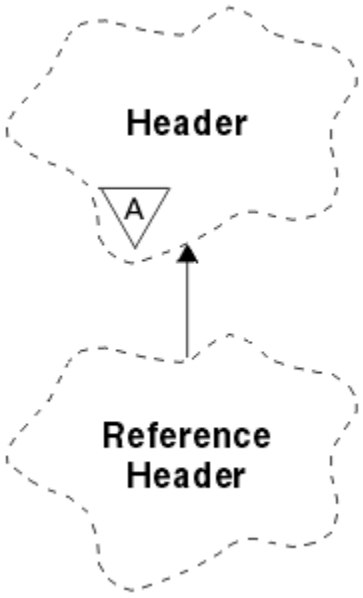


Figura 34. Classe intestazione ImqReference

Questa classe è relativa alle chiamate MQI elencate in “Riferimento incrociato intestazione ImqReference” a pagina 1850.

Attributi oggetto

ambiente di destinazione
Ambiente per la destinazione. Il valore iniziale è una stringa null.

nome destinazione
Nome della destinazione dati. Il valore iniziale è una stringa null.

ID istanza
Identificativo istanza. Un valore binario (MQBYTE24) di lunghezza MQ_OBJECT_INSTANCE_ID_LENGTH. Il valore iniziale è MQOII_NONE.

lunghezza logica
Lunghezza logica o prevista dei dati del messaggio che seguono questa intestazione. Il valore iniziale è zero.

offset logico
Offset logico per i dati del messaggio che seguono, da interpretare nel contesto dei dati nel loro insieme, alla destinazione finale. Il valore iniziale è zero.

offset logico 2
Estensione di ordine superiore all'offset logico. Il valore iniziale è zero.

tipo di riferimento
Tipo di riferimento. Il valore iniziale è una stringa null.

ambiente di origine
Ambiente per l'origine. Il valore iniziale è una stringa null.

nome dell'origine
Nome dell'origine dati. Il valore iniziale è una stringa null.

Costruttori

Intestazione ImqReference();
Il costruttore predefinito.

ImqReferenceImqReference (const ImqReferenceHeader & ***intestazione*** **);**
Il costruttore di copia.

Metodi ImqItem sovraccaricati

ImqBoolean copyOut virtuale (ImqMessage & ***msg*** **);**
Inserisce una struttura di dati MQRMH nel buffer di messaggi all'inizio, spostando ulteriormente i dati del messaggio esistenti e imposta il formato *msg* su MQFMT_REF_MSG_HEADER.

Consultare la descrizione del metodo della classe ImqHeader su "Classe ImqHeader C++" a pagina 1877 per ulteriori dettagli.

ImqBoolean pasteIn virtuale (ImqMessage & ***msg*** **);**
Legge una struttura dati MQRMH dal buffer del messaggio.

Per avere esito positivo, il formato ImqMessage deve essere MQFMT_REF_MSG_HEADER.

Consultare la descrizione del metodo della classe ImqHeader su "Classe ImqHeader C++" a pagina 1877 per ulteriori dettagli.

Metodi oggetto (pubblico)

void operator = (const ImqReferenceintestazione & ***intestazione*** **);**
Copia i dati di istanza da *intestazione*, sostituendo quelli esistenti.

ImqString destinationEnvironment () const;
Restituisce una copia dell'ambiente di destinazione.

void setDestinationEnvironment (const char * ***environment*** **= 0);**
Imposta l'ambiente di destinazione.

ImqString destinationName () const;
Restituisce una copia del nome della destinazione.

void setDestinationName (const char * ***nome*** **= 0);**
Imposta il nome della destinazione.

ImqBinary instanceId () const;
Restituisce una copia dell'ID istanza.

ImqBoolean setInstanceId (const ImqBinary & ***id*** **);**
Imposta l'ID istanza. La lunghezza dei dati del *token* deve essere 0 o MQ_OBJECT_INSTANCE_ID_LENGTH. Questo metodo restituisce TRUE se ha esito positivo.

void setInstanceId (const MQBYTE24 ***ID*** **= 0);**
Imposta l'ID istanza. *id* può essere zero, che equivale a specificare MQOII_NONE. Se *id* è diverso da zero, deve indirizzare i byte MQ_OBJECT_INSTANCE_ID_LENGTH dei dati binari. Quando si utilizzano valori predefiniti come MQOII_NONE, potrebbe essere necessario eseguire un cast per garantire una corrispondenza di firma, ad esempio (MQBYTE *) MQOII_NONE.

MQLONG logicalLength () const;
Restituisce la lunghezza logica.

void setLogicalLunghezza (const MQLONG ***lunghezza*** **);**
Imposta la lunghezza logica.

MQLONG logicalOffset () const;
Restituisce l'offset logico.

void setLogicalOffset (const MQLONG *offset*);
Imposta l'offset logico.

MQLONG logicalOffset2 () const;
Restituisce l'offset logico 2.

void setLogicalOffset2 (const MQLONG *offset*);
Imposta l'offset logico 2.

ImqString referenceType () const;
Restituisce una copia del tipo di riferimento.

void setReferenceType (const char * *name* = 0);
Imposta il tipo di riferimento.

ImqString sourceEnvironment () const;
Restituisce una copia dell'ambiente di origine.

void setSourceEnvironment (const char * *environment* = 0);
Imposta l'ambiente di origine.

ImqString sourceName () const;
Restituisce una copia del nome origine.

void setSourceNome (const char * *nome* = 0);
Imposta il nome origine.

Dati oggetto (protetti)

MQRMH *omqrmh*
La struttura dati MQRMH.

Codici di origine

- MQRC_BINARY_DATA_LENGTH_ERROR
- ERRORE MQRC_STRUC_LENGTH
- ERRORE MQRC_STRUC_ID
- DATI MQRC_INSUFFICIENT_
- MQRC_INCONSISTENT_FORMAT
- ERRORE MQRC_ENCODING_

Classe ImqString C++

Questa classe fornisce la memorizzazione della stringa di caratteri e la manipolazione per le stringhe con terminazione null.

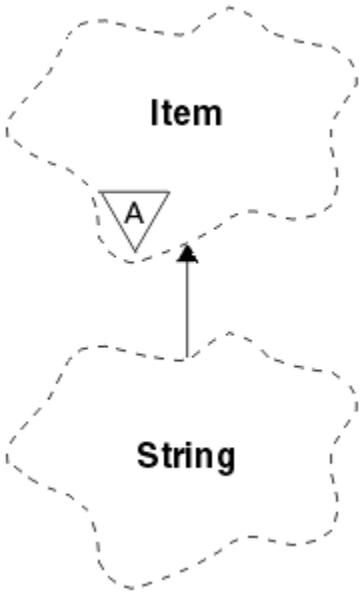


Figura 35. Classe ImqString

Utilizzare ImqString al posto di **char *** nella maggior parte delle situazioni in cui un parametro richiama un **char ***.

- "Attributi oggetto" a pagina 1935
- "Costruttori" a pagina 1935
- "Metodi di classe (pubblico)" a pagina 1936
- "Metodi ImqItem sovraccaricati" a pagina 1936
- "Metodi oggetto (pubblico)" a pagina 1936
- "Metodi oggetto (protetti)" a pagina 1939
- "Codici di origine" a pagina 1939

Attributi oggetto

caratteri
Caratteri nella **memoria** che precedono un valore null finale.

lunghezza
Numero di byte nei **caratteri**. Se non è presente alcuna **memoria**, la **lunghezza** è zero. Il valore iniziale è zero.

defined
Un array volatile di byte di dimensione arbitraria. Un valore null finale deve essere sempre presente nella **memoria** dopo **caratteri**, in modo che sia possibile rilevare la fine dei **caratteri** . I metodi assicurano che questa situazione venga mantenuta, ma assicurano, quando si impostano i byte direttamente nell'array, che esista un valore null finale dopo la modifica. Inizialmente, non esiste alcun attributo **storage** .

Costruttori

ImqString( );
Il costruttore predefinito.

ImqString(const ImqString & ***stringa*** **);**
Il costruttore di copia.

ImqString(carattere const ***c*** **);**
I **caratteri** comprendono *c*.

ImqString(const char * *testo*);
I **caratteri** vengono copiati da *testo*.

ImqString(const void * *buffer*, const size_t *lunghezza*);
Copia *lunghezza* byte a partire da *buffer* e li assegna ai **caratteri**. La sostituzione viene effettuata per tutti i caratteri null copiati. Il carattere di sostituzione è un punto (.). Non viene data particolare considerazione a nessun altro carattere non stampabile o non visualizzabile copiato.

Metodi di classe (pubblico)

copia statica ImqBoolean (char * *destination - buffer*, const size_t *length*, const char * *source - buffer*, const char *pad* = 0);
Copia fino a *lunghezza* byte da *source - buffer* a *destination - buffer*. Se il numero di caratteri in *source - buffer* non è sufficiente, riempie lo spazio rimanente in *destination - buffer* con i caratteri *pad* . *source - buffer* può essere zero. *destination - buffer* può essere zero se anche *lunghezza* è zero. Tutti i codici di errore vengono persi. Questo metodo restituisce TRUE se ha esito positivo.

statico ImqBoolean copy (char * *destination - buffer*, const size_t *length*, const char * *source - buffer*, ImqError & *error - object*, const char *pad* = 0);
Copia fino a *lunghezza* byte da *source - buffer* a *destination - buffer*. Se il numero di caratteri in *source - buffer* non è sufficiente, riempie lo spazio rimanente in *destination - buffer* con i caratteri *pad* . *source - buffer* può essere zero. *destination - buffer* può essere zero se anche *lunghezza* è zero. Tutti i codici di errore sono impostati in *error - object*. Questo metodo restituisce TRUE se ha esito positivo.

Metodi ImqItem sovraccaricati

ImqBoolean copyOut (ImqMessage & *msg*);
Copia i **caratteri** nel buffer di messaggi, sostituendo il contenuto esistente. Imposta il formato *msg* su MQFMT_STRING.

Consultare la descrizione del metodo della classe parent per ulteriori dettagli.

ImqBoolean pasteIn (ImqMessage & *msg*);
Imposta i **caratteri** trasferendo i restanti dati dal buffer di messaggi, sostituendo i **caratteri**esistenti.

Per avere successo, la **codifica** dell'oggetto *msg* deve essere MQENC_NATIVE. Richiamare i messaggi con MQGMO_CONVERT in MQENC_NATIVE.

Per avere esito positivo, il formato ImqMessage deve essere MQFMT_STRING.

Consultare la descrizione del metodo della classe parent per ulteriori dettagli.

Metodi oggetto (pubblico)

char & operator [] (const size_t *offset*) const;
Fa riferimento al carattere all'offset *offset* nella **memoria**. Verificare che il byte pertinente esista e che sia indirizzabile.

Operatore ImqString () (const size_t *offset*, const size_t *lunghezza* = 1) const;
Restituisce una sottostringa copiando i byte dai **caratteri** a partire da *offset*. Se *lunghezza* è zero, restituisce il resto dei **caratteri**. Se la combinazione di *offset* e *lunghezza* non produce un riferimento all'interno dei **caratteri**, restituisce un ImqStringvuoto.

operatore void = (const ImqString & *stringa*);
Copia i dati di istanza da *stringa*, sostituendo i dati di istanza esistenti.

operatore ImqString + (const char *c*) const;
Restituisce il risultato dell'accodamento di *c* ai **caratteri**.

Operatore ImqString + (const char * *text*) const;
Restituisce il risultato dell'aggiunta di *testo* ai **caratteri**. Questo può anche essere invertito. Ad esempio:

```
strOne + "string two" ;
"string one" + strTwo ;
```

Nota: Anche se la maggior parte dei compilatori accetta **strOne + "string two";** Microsoft Visual C++ richiede **strOne + (char *) "string two";**

operatore ImqString + (const ImqString & *string1*) const;
Restituisce il risultato dell'accodamento di *string1* ai **caratteri**.

ImqString operatore + (const double *number*) const;
Restituisce il risultato dell'accodamento di *numero* ai **caratteri** dopo la conversione in testo.

ImqString operatore + (const long *numero*) const;
Restituisce il risultato dell'accodamento di *numero* ai **caratteri** dopo la conversione in testo.

operatore void + = (const char *c*);
Aggiunge *c* ai **caratteri**.

operatore void + = (const char * *testo*);
Aggiunge *testo* ai **caratteri**.

operatore vuoto + = (const ImqString & *string*);
Accoda *stringa* ai **caratteri**.

operatore void + = (const double *numero*);
Aggiunge *numero* ai **caratteri** dopo la conversione in testo.

operatore void + = (const long *numero*);
Aggiunge *numero* ai **caratteri** dopo la conversione in testo.

operatore char * () const;
Restituisce l'indirizzo del primo byte nella **memoria**. Questo valore può essere zero ed è volatile. Utilizzare questo metodo solo per scopi di sola lettura.

Operatore ImqBoolean < (const ImqString & *string*) const;
Confronta i **caratteri** con quelli della *stringa* utilizzando il metodo **compare** . Il risultato è TRUE se minore di e FALSE se maggiore o uguale a.

operatore ImqBoolean > (const ImqString & *string*) const;
Confronta i **caratteri** con quelli della *stringa* utilizzando il metodo **compare** . Il risultato è TRUE se maggiore di e FALSE se minore o uguale a.

ImqBoolean operatore < = (const ImqString & *stringa*) const;
Confronta i **caratteri** con quelli della *stringa* utilizzando il metodo **compare** . Il risultato è TRUE se minore o uguale a e FALSE se maggiore di.

ImqBoolean operatore> = (const ImqString & *stringa*) const;
Confronta i **caratteri** con quelli della *stringa* utilizzando il metodo **compare** . Il risultato è TRUE se maggiore o uguale a e FALSE se minore di.

operatore ImqBoolean == (const ImqString & *stringa*) const;
Confronta i **caratteri** con quelli della *stringa* utilizzando il metodo **compare** . Restituisce TRUE o FALSE.

ImqBoolean operator! = (const ImqString & *stringa*) const;
Confronta i **caratteri** con quelli della *stringa* utilizzando il metodo **compare** . Restituisce TRUE o FALSE.

confronto breve (const ImqString & *string*) const;
Confronta i **caratteri** con quelli di *stringa*. Il risultato è zero se i **caratteri** sono uguali, negativo se minore di e positivo se maggiore di. Il confronto è sensibile al maiuscolo / minuscolo. Una ImqString null viene considerata minore di una ImqStringnon null.

ImqBoolean copyOut(char * *buffer*, const size_t *lunghezza*, const char *prisma* = 0);
Copia fino a *lunghezza* byte dai **caratteri** al *buffer*. Se il numero di **caratteri** non è sufficiente, riempie lo spazio rimanente nel *buffer* con *caratteri di riempimento* . *buffer* può essere zero se anche *lunghezza* è zero. Restituisce TRUE in caso di esito positivo.

size_t copyOut(long & *numero*) const;
Imposta *numero* dai **caratteri** dopo la conversione dal testo e restituisce il numero di caratteri coinvolti nella conversione. Se è zero, non è stata eseguita alcuna conversione e *numero* non viene impostato. Una sequenza di caratteri convertibili deve iniziare con i valori seguenti:

```
<blank(s)>
<+|->
digit(s)
```

size_t copyOut(ImqString & *token*, const char *c* = '') const;
Se i **caratteri** contengono uno o più caratteri diversi da *c*, identifica un token come la prima sequenza contigua di tali caratteri. In questo caso *token* è impostato su tale sequenza e il valore restituito è la somma del numero di caratteri iniziali *c* ed il numero di byte nella sequenza. Altrimenti, restituisce zero e non imposta *token*.

size_t cutOut(long & *numero*);
Imposta *numero* come per il metodo **copy** , ma rimuove da **caratteri** il numero di byte indicato dal valore di ritorno. Ad esempio, la stringa mostrata nel seguente esempio può essere tagliata in tre numeri utilizzando **cutOut** (*numero*) tre volte:

```
strNumbers = "-1 0   +55 "

while ( strNumbers.cutOut( number ) );
number becomes -1, then 0, then 55
leaving strNumbers == " "
```

size_t cutOut(ImqString & *token*, const char *c* = ' '
Imposta *token* come per il metodo **copyOut** e rimuove dai **caratteri** i caratteri *strToken* e tutti i caratteri *c* che precedono i caratteri *token* . Se *c* non è uno spazio vuoto, rimuove i caratteri *c* che succedono direttamente i caratteri *token* . Restituisce il numero di caratteri rimossi. Ad esempio, la stringa mostrata nel seguente esempio può essere tagliata in tre token utilizzando **cutOut** (*token*) tre volte:

```
strText = " Program Version 1.1 "

while ( strText.cutOut( token ) );

// token becomes "Program", then "Version",
// then "1.1" leaving strText == " "
```

Il seguente esempio mostra come analizzare un nome percorso DOS:

```
strPath = "C:\OS2\BITMAP\OS2LOGO.BMP"

strPath.cutOut( strDrive, ':' );
strPath.stripLeading( ':' );
while ( strPath.cutOut( strFile, '\' ) );

// strDrive becomes "C".
// strFile becomes "OS2", then "BITMAP",
// then "OS2LOGO.BMP" leaving strPath empty.
```

ImqBoolean find (const ImqString & *stringa*);
Ricerca una corrispondenza esatta per *stringa* in qualsiasi punto all'interno dei **caratteri**. Se non viene trovata alcuna corrispondenza, restituisce FALSE. Altrimenti, restituisce TRUE. Se *stringa* è null, restituisce TRUE.

ImqBoolean find (const ImqString & *stringa*, size_t e *offset*);
Ricerca una corrispondenza esatta per *stringa* da qualche parte all'interno dei **caratteri** dall'offset *offset* in poi. Se *stringa* è null, restituisce TRUE senza aggiornare *offset*. Se non viene trovata alcuna corrispondenza, restituisce FALSE (il valore di *offset* potrebbe essere stato aumentato). Se viene trovata una corrispondenza, restituisce TRUE e aggiorna *offset* all'offset di *stringa* all'interno dei **caratteri**.

size_t lunghezza () const;
Restituisce la **lunghezza**.

ImqBoolean pasteIn(const doppio *numero*, const char * *formato* = "%f");
Aggiunge *numero* ai **caratteri** dopo la conversione in testo. Restituisce TRUE in caso di esito positivo.

La specifica *formato* viene utilizzata per formattare la conversione a virgola mobile. Se specificato, deve essere adatto per l'utilizzo con **printf** e numeri a virgola mobile, ad esempio **%.3f**.

ImqBoolean pasteIn(const long *numero*);
Aggiunge *numero* ai **caratteri** dopo la conversione in testo. Restituisce TRUE in caso di esito positivo.

ImqBoolean pasteIn(const void * *buffer*, const size_t *lunghezza*);
Accoda *lunghezza* byte da *buffer* ai **caratteri**e aggiunge un valore finale null finale. Sostituisce tutti i caratteri null copiati. Il carattere di sostituzione è un punto (.). Non viene data alcuna considerazione speciale ad altri caratteri non stampabili o non visualizzabili copiati. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean set (const char * *buffer*, const size_t *lunghezza*);
Imposta i **caratteri** da un campo di caratteri a lunghezza fissa, che potrebbe contenere un valore null. Aggiunge un valore null ai caratteri del campo a lunghezza fissa, se necessario. Questo metodo restituisce TRUE se ha esito positivo.

ImqBoolean setStorage(const size_t *lunghezza*);
Assegna (o riassegna) la **memoria**. Conserva tutti i **caratteri**originali, incluso qualsiasi valore null finale, se c'è ancora spazio per essi, ma non inizializza alcuna memoria aggiuntiva.

Questo metodo restituisce TRUE se ha esito positivo.

dimensione_t storage () const;
Restituisce il numero di byte nella **memoria**.

size_t stripLeading(const char *c* = '');
Rimuove i caratteri iniziali *c* dai **caratteri** e restituisce il numero rimosso.

size_t stripTrailing(const char *c* = '');
Elimina i caratteri finali *c* dai **caratteri** e restituisce il numero rimosso.

ImqString upperCase() const;
Restituisce una copia in maiuscolo dei **caratteri**.

Metodi oggetto (protetti)

ImqBoolean assign (const ImqString & *stringa*);
Equivalente al metodo **operator =** equivalente, ma non virtuale. Restituisce TRUE in caso di esito positivo.

Codici di origine

- DATA_TRUNCATED MQRC_
- NULL_POINTER MQRC
- MQRC_STORAGE_NON_DISPONIBILE
- ERRORE MQRC_BUFFER_
- MQRC_INCONSISTENT_FORMAT

classe C++ ImqTrigger

Questa classe incapsula la struttura dati MQTM (trigger message).

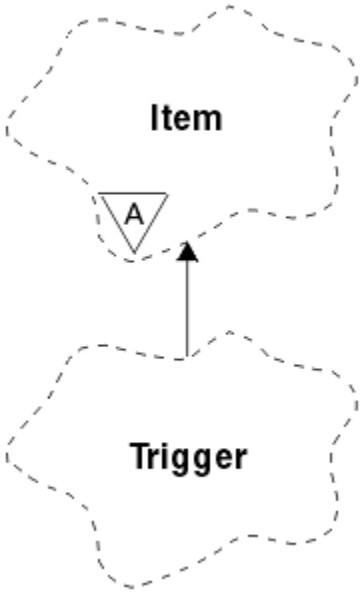


Figura 36. classe ImqTrigger

Gli oggetti di questa classe vengono generalmente utilizzati da un programma di controllo trigger. L'attività di un programma di controllo trigger consiste nell'attendere questi particolari messaggi e agire su di essi per garantire che altre applicazioni IBM MQ vengano avviate quando i messaggi sono in attesa di tali messaggi.

Consultare il programma di esempio IMQSTRG per un esempio di utilizzo.

- "Attributi oggetto" a pagina 1940
- "Costruttori" a pagina 1941
- "Metodi ImqItem sovraccaricati" a pagina 1941
- "Metodi oggetto (pubblico)" a pagina 1941
- "Dati oggetto (protetti)" a pagina 1942
- "Codici di origine" a pagina 1942

Attributi oggetto

ID applicazione
Identità dell'applicazione che ha inviato il messaggio. Il valore iniziale è una stringa null.

tipo di applicazione
Tipo di applicazione che ha inviato il messaggio. Il valore iniziale è zero. Sono possibili i seguenti valori aggiuntivi:

- AIX MQAT
- MQAT_CICS
- DOS MQAT
- IMS MQAT
- MVS MQAT
- MQAT_NOTES_AGENT
- MQAT_OS2
- MQAT_OS390

- MQAT_OS400
- UNIX MQAT
- WINDOWS MQAT
- MQAT_WINDOWS_NT
- MQAT_USER_FIRST
- MQAT_USER_LAST

Dati ambiente
Dati di ambiente per il processo. Il valore iniziale è una stringa null.

nome del processo
Il nome del processo. Il valore iniziale è una stringa null.

nome coda
Nome della coda da avviare. Il valore iniziale è una stringa null.

dati del trigger
Dati trigger per il processo. Il valore iniziale è una stringa null.

dati utente
Dati utente per il processo. Il valore iniziale è una stringa null.

Costruttori

ImqTrigger( );
Il costruttore predefinito.

ImqTrigger(const ImqTrigger & *trigger*);
Il costruttore di copia.

Metodi ImqItem sovraccaricati

ImqBoolean copyOut virtuale (ImqMessage & *msg*);
Scrive una struttura di dati MQTM nel buffer di messaggi, sostituendo qualsiasi contenuto esistente. Imposta il formato *msg* su MQFMT_TRIGGER.

Consultare la descrizione del metodo della classe ImqItem all'indirizzo “Classe ImqItem C++” a pagina 1882 per ulteriori dettagli.

ImqBoolean pasteIn virtuale (ImqMessage & *msg*);
Legge una struttura dati MQTM dal buffer di messaggi.

Per avere esito positivo, il formato ImqMessage deve essere MQFMT_TRIGGER.

Consultare la descrizione del metodo della classe ImqItem all'indirizzo “Classe ImqItem C++” a pagina 1882 per ulteriori dettagli.

Metodi oggetto (pubblico)

void operator = (const ImqTrigger & *trigger*);
Copia i dati istanza da *trigger*, sostituendo i dati istanza esistenti.

ImqString applicationId () const;
Restituisce una copia dell'ID applicazione.

void setApplicationId (const char * *id*);
Imposta l'ID applicazione.

MQLONG applicationType () const;
Restituisce il tipo di applicazione.

void setApplicationType (const MQLONG *type*);
Imposta il tipo di applicazione.

ImqBoolean copyOut (MQTMC2 * *ptmc2*);
Incapsula la struttura dati MQTM, che è quella ricevuta sulle code di iniziazione. Compila una struttura dati MQTMC2 equivalente fornita dal chiamante e imposta il campo QMgrName (che non è presente nella struttura dati MQTM) su tutti i valori vuoti. La struttura dati MQTMC2 viene utilizzata tradizionalmente come parametro per le applicazioni avviate da un controllo trigger. Questo metodo restituisce TRUE se ha esito positivo.

ImqString environmentData () const;
Restituisce una copia dei dati di ambiente.

void setEnvironmentData (const char * *data*);
Imposta i dati di ambiente.

ImqString processName () const;
Restituisce una copia del nome processo.

void setProcessNome (const char * *nome*);
Imposta il nome del processo, riempito con spazi vuoti a 48 caratteri.

ImqString queueName () const;
Restituisce una copia del nome della coda.

void setQueueNome (const char * *nome*);
Imposta il nome della coda, riempiendo con spazi vuoti fino a 48 caratteri.

ImqString triggerData () const;
Restituisce una copia dei dati trigger.

void setTriggerData (const char * *dati*);
Imposta i dati trigger.

ImqString userData () const;
Restituisce una copia dei dati utente.

void setUserData (const char * *data*);
Imposta i dati utente.

Dati oggetto (protetti)

MQTM *omqtm*
La struttura dei dati MQTM.

Codici di origine

- NULL_POINTER MQRC
- MQRC_INCONSISTENT_FORMAT
- ERRORE MQRC_ENCODING_
- ERRORE MQRC_STRUC_ID

Classe C++ ImqWorkHeader

Questa classe incapsula funzioni specifiche della struttura dati MQWIH.

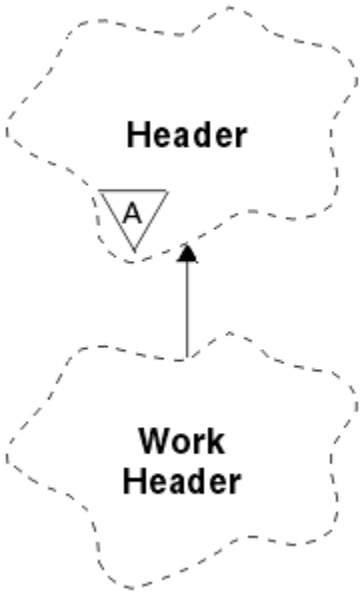


Figura 37. Classe di intestazione ImqWork

Gli oggetti di questa classe vengono utilizzati da applicazioni che inseriscono messaggi nella coda gestita da z/OS Workload Manager.

- "Attributi oggetto" a pagina 1943
- "Costruttori" a pagina 1943
- "Metodi ImqItem sovraccaricati" a pagina 1943
- "Metodi oggetto (pubblico)" a pagina 1944
- "Dati oggetto (protetti)" a pagina 1944
- "Codici di origine" a pagina 1944

Attributi oggetto

token del messaggio
Token messaggio per z/OS Workload Manager, di lunghezza MQ_MSG_TOKEN_LENGTH. Il valore iniziale è MQMTOK_NONE.

nome servizio
Il nome di 32 caratteri di un processo. Il nome è inizialmente vuoto.

fase di servizio
Il nome di 8 caratteri di un passo all'interno del processo. Il nome è inizialmente vuoto.

Costruttori

Intestazione ImqWork();
Il costruttore predefinito.

Intestazione ImqWork(const ImqWorkHeader & *header*);
Il costruttore di copia.

Metodi ImqItem sovraccaricati

ImqBoolean copyOutvirtuale (ImqMessage & *msg*);
Inserisce una struttura dati MQWIH all'inizio del buffer del messaggio, spostando ulteriormente i dati del messaggio esistenti e imposta il formato *msg* su MQFMT_WORK_INFO_HEADER.

Consultare la descrizione del metodo della classe parent per ulteriori dettagli.

ImqBoolean pasteInvirtuale (ImqMessage & *msg*);
Legge una struttura dati MQWIH dal buffer di messaggio.

Per avere esito positivo, la codifica dell'oggetto *msg* deve essere MQENC_NATIVE. Richiamare i messaggi con MQGMO_CONVERT in MQENC_NATIVE.

Il formato di ImqMessage deve essere MQFMT_WORK_INFO_HEADER.

Consultare la descrizione del metodo della classe parent per ulteriori dettagli.

Metodi oggetto (pubblico)

void operator = (const ImqWorkHeader & *header*);
Copia i dati di istanza da *intestazione*, sostituendo quelli esistenti.

ImqBinary messageToken () const;
Restituisce il **token del messaggio**.

ImqBoolean setMessageToken (const ImqBinary & *token*);
Imposta il **token messaggio**. La lunghezza dei dati di *token* deve essere zero o MQ_MSG_TOKEN_LENGTH. Restituisce TRUE in caso di esito positivo.

void setMessageToken (const MQBYTE16 *token* = 0);
Imposta il **token messaggio**. *token* può essere zero, che equivale a specificare MQMTOK_NONE. Se *token* è diverso da zero, deve indirizzare i byte MQ_MSG_TOKEN_LENGTH dei dati binari.

Quando si utilizzano valori predefiniti come MQMTOK_NONE, potrebbe essere necessario eseguire un cast per garantire una corrispondenza della firma; ad esempio, (MQBYTE *) MQMTOK_NONE.

ImqString serviceName () const;
Restituisce il **nome servizio**, inclusi gli spazi finali.

void setServiceNome (const char * *nome*);
Imposta il **nome servizio**.

ImqString serviceStep () const;
Restituisce il **passo del servizio**, inclusi i vuoti finali.

void setServicePasso (const char * *passo*);
Imposta il **passo del servizio**.

Dati oggetto (protetti)

MQWIH omqwih
La struttura dati MQWIH.

Codici di origine

- MQRC_BINARY_DATA_LENGTH_ERROR

Proprietà degli oggetti IBM MQ classes for JMS

Tutti gli oggetti in IBM MQ classes for JMS hanno proprietà. Proprietà differenti si applicano a tipi di oggetto differenti. Proprietà differenti hanno valori consentiti differenti e i valori delle proprietà simbolici differiscono tra lo strumento di gestione e il codice programma.

IBM MQ classes for JMS fornisce funzioni per impostare e interrogare le proprietà degli oggetti utilizzando lo strumento di amministrazione IBM MQ JMS , IBM MQ Explorer o in un'applicazione. Molte delle proprietà sono rilevanti solo per uno specifico sottoinsieme dei tipi di oggetto.

Per informazioni su come utilizzare lo Strumento di gestione di IBM MQ JMS , consultare Configurazione degli oggetti JMS mediante lo strumento di gestione.

Tabella 868 a pagina 1945 fornisce una breve descrizione di ciascuna proprietà e mostra per ogni proprietà a quali tipi di oggetto si applica. I tipi di oggetto vengono identificati utilizzando parole

chiave; consultare Configurazione degli oggetti JMS mediante lo strumento di amministrazione per una spiegazione di tali oggetti.

I numeri si riferiscono alle note alla fine della tabella. Vedi anche “Dipendenze tra proprietà di oggetti IBM MQ classes for JMS” a pagina 1948.

Una proprietà è costituita da una coppia nome - valore nel formato:

```
PROPERTY_NAME(property_value)
```

Gli argomenti in questa sezione elencano, per ogni proprietà, il nome della proprietà e una breve descrizione e mostrano i valori di proprietà validi utilizzati nello strumento di amministrazione. e il metodo set utilizzato per impostare il valore della proprietà in un'applicazione. Gli argomenti mostrano anche i valori di proprietà validi per ciascuna proprietà e la corrispondenza tra i valori di proprietà simbolici utilizzati nello strumento e i relativi equivalenti programmabili.

I nomi proprietà non sono sensibili al maiuscolo / minuscolo e sono limitati alla serie di nomi riconosciuti mostrati in questi argomenti.

Tabella 868. Nomi proprietà e tipi di oggetto applicabili

Proprietà	**Forma breve**	**Tipo oggetto**							
		CF	**QCF**	**TCF**	**Q**	**T**	**XACF**	**XAQCF**	**XATCF**
“APPLICATIONNAME” a pagina 1950	APPNAME	Y	Y	Y			Y	Y	Y
“ECCEZIONE asincrona” a pagina 1950	AEX	Y	Y	Y			Y	Y	Y
“BALOPTIONS” a pagina 1951	OPZIONI	Y	Y	Y			Y	Y	Y
“TIPOALB” a pagina 1952	TIPO	Y	Y	Y			Y	Y	Y
“SUPEROTEMPO” a pagina 1952	TIMEOUT	Y	Y	Y			Y	Y	Y
“BROKERCCDURSUBQ” a pagina 1953 [1]	CCDSUB					Y			
“BROKERCCSUBQ” a pagina 1954 [1]	CCSUB	Y		Y			Y		Y
“BROKERCONQ” a pagina 1954 [1]	BCON	Y		Y			Y		Y
“BROKERDURSUBQ” a pagina 1954 [1]	BDSUB					Y			
“BROKERPUBQ” a pagina 1955 [1]	BPUB	Y		Y		Y	Y		Y
“BROKERPUBQMGR” a pagina 1955 [1]	BPQM					Y			
“BROKERQMGR” a pagina 1956 [1]	BQM	Y		Y			Y		Y
“BROKERSUBQ” a pagina 1956 [1]	BSUB	Y		Y			Y		Y
“BROKERVER” a pagina 1957 [1]	BVER	S [2]		S [2]		Y	Y		Y
“CCDTURL” a pagina 1957 [3]	CCDT	Y	Y	Y			Y	Y	Y
“CCSID” a pagina 1958	CCS	Y	Y	Y	Y	Y	Y	Y	Y
“CHANNEL” a pagina 1959 [3]	CHAN	Y	Y	Y			Y	Y	Y
“CLEANUP” a pagina 1959 [1]	CL	Y		Y			Y		Y
“CLEANUPINT” a pagina 1960 [1]	CLINT	Y		Y			Y		Y
“ConnectionNameList” a pagina 1960	ELENCO	Y	Y	Y					

Tabella 868. Nomi proprietà e tipi di oggetto applicabili (Continua)

Proprietà	Forma breve	Tipo oggetto							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
"CLIENTRECONNECTOPTIONS" a pagina 1961	CROPT	Y	Y	Y					
"CLIENTRECONNECTTIMEOUT" a pagina 1962	CRT	Y	Y	Y					
"CLIENTID" a pagina 1962	CID	S [2]	Y	S [2]			Y	Y	Y
"CLONESUPP" a pagina 1962	CLS	Y		Y			Y		Y
"COMPHDR" a pagina 1963	HC	Y		Y			Y		Y
"COMPMSG" a pagina 1963	MC	Y	Y	Y			Y	Y	Y
"CONNOPT" a pagina 1964	CNOPT	Y	Y	Y			Y	Y	Y
"CONNTAG" a pagina 1965	CNTAG	Y	Y	Y			Y	Y	Y
"DESCRIZIONE" a pagina 1965	DESC	S [2]	Y	S [2]	Y	Y	Y	Y	Y
"DIRECTAUTH" a pagina 1966	DAUTH	S [2]		S [2]					
"Codifica" a pagina 1966	ENC				Y	Y			
"EXPIRY" a pagina 1967	EXP				Y	Y			
"FAILIFQUIESCE" a pagina 1968	FIQ	Y	Y	Y	Y	Y	Y	Y	Y
"HOSTNAME" a pagina 1968	HOST	S [2]	Y	S [2]			Y	Y	Y
"LOCALADDRESS" a pagina 1969	LA	S [2]	Y	S [2]			Y	Y	Y
"NOMEMAPPA" a pagina 1970	MNST	Y	Y	Y			Y	Y	Y
"MAXBUFFSIZE" a pagina 1970	MBSZ	S [2]		S [2]					
"MDREAD" a pagina 1971	MDR				Y	Y			
"MDWRITE" a pagina 1971	MDW				Y	Y			
"MDMSGCTX" a pagina 1972	MDCTX				Y	Y			
"MSGBATCHSZ" a pagina 1972 [1]	MBS	Y	Y	Y			Y	Y	Y
"MSGBODY" a pagina 1973	MBODY				Y	Y			
"MSGRETENTION" a pagina 1973	MRET	Y	Y				Y	Y	
"MSGSELECTION" a pagina 1974 [1]	MSEL	Y		Y			Y		Y
"MULTICAST" a pagina 1974	MCAST	S [2]		S [2]		Y			
"OPTIMISTICPUBLICATION" a pagina 1975 [1]	OPTPUB	Y		Y					
"OUTCOMENOTIFICATION" a pagina 1976 [1]	NOTIFY	Y		Y					
"PERSISTENCE" a pagina 1976	PER				Y	Y			
"POLLINGINT" a pagina 1977 [1]	PINT	Y	Y	Y			Y	Y	Y
"PORTA" a pagina 1977	PORTA	S [2]	Y	S [2]			Y	Y	Y
"PRIORITY" a pagina 1978	PRI				Y	Y			

Tabella 868. Nomi proprietà e tipi di oggetto applicabili (Continua)

Proprietà	Forma breve	Tipo oggetto							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
"PROCESSDURATION" a pagina 1978 [1]	PROCDUR	Y		Y					
"PROVIDERVERSION" a pagina 1979	PVER	Y	Y	Y			Y	Y	Y
"PROXYHOSTNAME" a pagina 1981	PHOST	S [2]		S [2]					
"PROXYPORT" a pagina 1982	PPORT	S [2]		S [2]					
"PUBACKINT" a pagina 1982 [1]	PAI	Y		Y			Y		Y
"PUTASYNCALLOWED" a pagina 1982	PAALD				Y	Y			
"QMANAGER" a pagina 1983	QMGR	Y	Y	Y	Y		Y	Y	Y
"CODA" a pagina 1984	QU				Y				
"READAHEADALLOWED" a pagina 1984	RAALD				Y	Y			
"READAHEADCLOSEPOLICY" a pagina 1985	RACP				Y	Y			
"RECEIVECCSID" a pagina 1985	RCCS				Y	Y			
"RECEIVECONVERSION" a pagina 1986	RCNV				Y	Y			
"RECEIVEISOLATION" a pagina 1986 [1]	RCVISOL	Y		Y					
"RECEXIT" a pagina 1987	RCX	Y	Y	Y			Y	Y	Y
"RECEXITINIT" a pagina 1987	RCXI	Y	Y	Y			Y	Y	Y
"REPLYTOSTYLE" a pagina 1988	RTOST				Y	Y			
"RESCANINT" a pagina 1988 [1]	RINT	Y	Y				Y	Y	
"SECEXIT" a pagina 1989	SCX	Y	Y	Y			Y	Y	Y
"SECEXITINIT" a pagina 1989	SCXI	Y	Y	Y			Y	Y	Y
"SENDCHECKCOUNT" a pagina 1990	SCC	Y	Y	Y			Y	Y	Y
"SENDEXIT" a pagina 1990	SDX	Y	Y	Y			Y	Y	Y
"SENDEXITINIT" a pagina 1991	SDXI	Y	Y	Y			Y	Y	Y
"SHARECONVALLOWED" a pagina 1991	SCALD	Y	Y	Y			Y	Y	Y
"SPARSESUBS" a pagina 1992 [1]	SSUBS	Y		Y					
"SSLCIPHERSUITE" a pagina 1992	SCPHS	Y	Y	Y			Y	Y	Y
"SSLCRL" a pagina 1993	SCRL	Y	Y	Y			Y	Y	Y
"SSLFIPSREQUIRED" a pagina 1993	SFIPS	Y	Y	Y			Y	Y	Y
"SSLPEERNAME" a pagina 1994	SPEER	Y	Y	Y			Y	Y	Y
"SSLRESETCOUNT" a pagina 1994	SRC	Y	Y	Y			Y	Y	Y

Tabella 868. Nomi proprietà e tipi di oggetto applicabili (Continua)

Proprietà	Forma breve	Tipo oggetto							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
"STATREFRESHINT" a pagina 1995 [1]	SRI	Y		Y			Y		Y
"SUBSTORE" a pagina 1995 [1]	SS	Y		Y			Y		Y
"SYNCPOINTALLGETS" a pagina 1996	SPAG	Y	Y	Y			Y	Y	Y
"TARGCLIENT" a pagina 1996	TC				Y	Y			
"TARGCLIENTMATCHING" a pagina 1997	TCM	Y	Y				Y	Y	
"TEMPMODEL" a pagina 1997	TM	Y	Y				Y	Y	
"TEMPQPREFIX" a pagina 1998	TQP	Y	Y				Y	Y	
"TEMPTOPICPREFIX" a pagina 1998	TTP	Y		Y			Y		Y
"TOPIC" a pagina 1999	TOP					Y			
"TRANSPORT" a pagina 1999	TRAN	S [2]	Y	S [2]			Y	Y	Y
"WILDCARDFORMAT" a pagina 2000	WCFMT	Y		Y			Y		Y

Nota:

1. Questa proprietà può essere utilizzata con la versione 70 di IBM MQ classes for JMS ma non ha alcun effetto per un'applicazione connessa a un gestore code IBM WebSphere MQ 7.0 a meno che la proprietà PROVIDERVERSION della factory di connessione non sia impostata su un numero di versione inferiore a 7.
2. Solo le proprietà BROKERVER, CLIENTID, DESCRIPTION, DIRECTAUTH, HOSTNAME, LOCALADDRESS, MAXBUFFSIZE, MULTICAST, PORT, PROXYHOSTNAME, PROXYPORT e TRANSPORT sono supportate per un oggetto factory ConnectionFactory o TopicConnectionquando si utilizza una connessione in tempo reale ad un broker.
3. Le proprietà CCDTURL e CHANNEL di un oggetto non devono essere impostate contemporaneamente.

Dipendenze tra proprietà di oggetti IBM MQ classes for JMS

La validità di alcune proprietà dipende dai valori particolari di altre proprietà.

Questa dipendenza può verificarsi nei seguenti gruppi di proprietà:

- Proprietà client
- Proprietà per una connessione in tempo reale a un broker
- Esci dalle stringhe di inizializzazione

Proprietà client

Per una connessione a un gestore code, le seguenti proprietà sono rilevanti solo se TRANSPORT ha il valore CLIENT:

- HOSTNAME
- PORTA
- CHANNEL
- LOCALADDRESS
- CCDTURL
- CCSID

- COMPHDR
- COMPMSG
- RECEXIT
- RECEXITINIT
- SECEXIT
- SECEXITINIT
- SENDEXIT
- SENDEXITINIT
- SHARECONVALLOWED
- SSLCIPHERSUITE
- SSLCRL
- SSLFIPSREQUIRED
- SSLPEERNAME
- SSLRESETCOUNT
- APPLICATIONNAME

Non è possibile impostare i valori per queste proprietà utilizzando lo strumento di amministrazione se TRANSPORT ha il valore BIND.

Se TRANSPORT ha il valore CLIENT, il valore predefinito della proprietà BROKERVER è V1 e il valore predefinito della proprietà PORT è 1414. Se si imposta esplicitamente il valore di BROKERVER o PORT, una modifica successiva al valore di TRANSPORT non sovrascrive le scelte.

Proprietà per una connessione in tempo reale a un broker

Solo le seguenti proprietà sono rilevanti se TRANSPORT ha il valore DIRECT o DIRECTHTTP:

- BROKERVER
- CLIENTID
- DESCRIZIONE
- DIRECTAUTH
- HOSTNAME
- LOCALADDRESS
- MAXBUFFSIZE
- MULTICAST (supportato solo per DIRECT)
- PORTA
- PROXYHOSTNAME (supportato solo per DIRECT)
- PROXYPORT (supportato solo per DIRECT)

Se TRANSPORT ha il valore DIRECT o DIRECTHTTP, il valore predefinito della proprietà BROKERVER è V2e il valore predefinito della proprietà PORT è 1506. Se si imposta esplicitamente il valore di BROKERVER o PORT, una modifica successiva al valore di TRANSPORT non sovrascrive le scelte.

Esci dalle stringhe di inizializzazione

Non impostare alcuna delle stringhe di inizializzazione dell'uscita senza fornire il nome dell'uscita corrispondente. Le proprietà di inizializzazione dell'uscita sono:

- RECEXITINIT
- SECEXITINIT
- SENDEXITINIT

Ad esempio, la specifica di `RECEXITINIT(myString)` senza specificare `RECEXIT(some.exit.classname)` causa un errore.

Riferimenti correlati

"TRANSPORT" a pagina 1999

La natura di una connessione a un gestore code o broker.

APPLICATIONNAME

Un'applicazione può impostare un nome che identifica la connessione al gestore code. Questo nome applicazione viene visualizzato dal comando **DISPLAY CONN MQSC/PCF** (dove il campo è denominato **APPLTAG**) o nella visualizzazione IBM MQ Explorer **Connessioni alle applicazioni** (dove il campo è denominato **App name**).

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : APPLICATIONNAME

Nome breve dello strumento di amministrazione JMS : APPNAME

Accesso programmatico

Setter / getter

- MQConnectionFactory.setAppNome ()
- MQConnectionFactory.getAppNome ()

Valori

Qualsiasi stringa valida che non superi i 28 caratteri. I nomi più lunghi vengono adattati rimuovendo i nomi dei pacchetti iniziali, se necessario. Ad esempio, se la classe di richiamo è `com.example.MainApp`, viene utilizzato il nome completo, ma se la classe di richiamo è `com.example.dictionaryAndThesaurus.multilingual.mainApp`, viene utilizzato il nome `multilingual.mainApp` , poiché è la combinazione più lunga di nome classe e nome pacchetto più a destra che si adatta alla lunghezza disponibile.

Se il nome della classe è più lungo di 28 caratteri, viene troncato per adattarlo. Ad esempio, `com.example.mainApplicationForSecondTestCase` diventa `mainApplicationForSecondTest`.

z/OS Su z/OS, APPNAME in:

- La modalità di bind viene ignorata se impostata e, se impostata, può essere impostata solo su spazi vuoti.
- La modalità client può essere impostata e utilizzata.

ECCEZIONE asincrona

Questa proprietà determina se IBM MQ classes for JMS informa un ExceptionListener solo quando una connessione viene interrotta o quando si verifica una qualsiasi eccezione in modo asincrono a una chiamata API JMS . Ciò si applica a tutte le connessioni create da questo ConnectionFactory che hanno un ExceptionListener registrato.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : ASYNCEXCEPTION

Nome breve dello strumento di amministrazione JMS : AEX

Accesso programmatico

Setter / Getter

- MQConnectionFactory.setAsyncEccezioni ()
- MQConnectionFactory.getAsyncEccezioni ()

Valori

ASYNC_EXCEPTIONS_ALL

Qualsiasi eccezione rilevata in modo asincrono, al di fuori dell'ambito di una chiamata API sincrona e tutte le eccezioni di connessione interrotta vengono inviate a ExceptionListener.

Tabella 869. Tutte le eccezioni asincrone: ambienti e nomi di costanti correlati

Ambiente	Valore
JMS strumento di amministrazione	TUTTO
Programmatico	WMQCONSTANTS.ASYNC_EXCEPTIONS_ALL = -1
IBM MQ Explorer	Tutto

ASYNC_EXCEPTIONS_CONNECTIONBROKEN

Solo le eccezioni che indicano una connessione interrotta vengono inviate a ExceptionListener. Tutte le altre eccezioni che si verificano durante l'elaborazione asincrona non vengono notificate a ExceptionListenere quindi l'applicazione non viene informata di tali eccezioni. Questo è il valore predefinito da IBM MQ 8.0.0 Fix Pack 2. Consultare JMS: Exception listener changes in IBM MQ 8.0.

Tabella 870. Eccezioni che indicano una connessione interrotta: ambienti e nomi costanti correlati

Ambiente	Valore
JMS strumento di amministrazione	COLLEGAMENTOINTERROTTO
Programmatico	WMQCONSTANTS.ASYNC_EXCEPTIONS_CONNECTIONBROKEN = 1
IBM MQ Explorer	Connessione interrotta

Viene definita la seguente costante aggiuntiva:

- Da IBM MQ 8.0.0 Fix Pack 2: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_CONNECTIONBROKEN
- Prima di IBM MQ 8.0.0 Fix Pack 2: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_ALL

Concetti correlati

Eccezioni in IBM MQ classes for JMS

V 9.4.0 BALOPTIONS

Controlla il modo in cui le applicazioni IBM MQ classes for JMS che utilizzano il trasporto client vengono ribilanciate in cluster uniformi.

Oggetti applicabili

`ConnectionFactory`, `QueueConnectionFactory`, `TopicConnectionFactory`, `XAConnectionFactory`, `XAQueueConnectionFactory`, `XATopicConnectionFactory`.

JMS nome lungo strumento di amministrazione: **BALOPTIONS**

JMS nome breve strumento di amministrazione: **OPTIONS**

Accesso programmatico

Setter / getter

- `MQConnectionFactory.setBalancingOptions()`
- `MQConnectionFactory.getBalancingOptions()`

Valori

IGNNONE
Si applica la normale gestione delle transazioni e non è richiesto lo spostamento delle applicazioni durante una transazione.

Questo valore è associato a IBM MQ *BalancingOption* `MQBNO_OPTIONS_NONE`.

IGNTRANS
È possibile che venga richiesto lo spostamento delle applicazioni durante una transazione.

Questo valore viene associato a IBM MQ *BalancingOption* `MQBNO_OPTIONS_IGNORE_TRANS`.

V 9.4.0 TIPOALB

Controlla il modo in cui le applicazioni IBM MQ classes for JMS che utilizzano il trasporto client possono essere ribilanciate in un cluster uniforme.

Oggetti applicabili

`ConnectionFactory`, `QueueConnectionFactory`, `TopicConnectionFactory`, `XAConnectionFactory`, `XAQueueConnectionFactory`, `XATopicConnectionFactory`.

JMS nome lungo strumento di amministrazione: **BALTYPE**

JMS nome breve strumento di amministrazione: **TYPE**

Accesso programmatico

Setter / getter

- `MQConnectionFactory.setBalancingApplicationType()`
- `MQConnectionFactory.getBalancingApplicationType()`

Valori

Semplice
Si applica la gestione predefinita delle applicazioni in un cluster uniforme.

Questo valore è associato a IBM MQ *BalancingOption* `MQBNO_BALTYPE_SIMPLE`.

RICHIESTA_RISPOSTA
All'applicazione non verrà richiesto di riconnettersi se un **MQPUT** non è stato bilanciato da un **MQGET**, a meno che non sia trascorso il periodo di timeout.

Questo valore è associato a IBM MQ *BalancingOption* `MQBNO_BALTYPE_REQREP`.

V 9.4.0 SUPEROTEMPO

Controlla il modo in cui le applicazioni IBM MQ classes for JMS che utilizzano il trasporto client vengono ribilanciate in un cluster uniforme.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory.

JMS nome lungo strumento di amministrazione: **BALTIMEOUT**

JMS nome breve strumento di amministrazione: **TIMEOUT**

Accesso programmatico

Setter / getter

- MQConnectionFactory.setBalancingTimeout()
- MQConnectionFactory.getBalancingTimeout()

Valori

MAI
L'applicazione non va mai in timeout ai fini del ribilanciamento in un cluster uniforme.

Questo valore si associa a IBM MQ *BalancingOption* MQBNO_TIMEOUT_NEVER.

IMMEDIATO
L'applicazione va immediatamente in timeout per il ribilanciamento in un cluster uniforme.

Questo valore viene associato a IBM MQ *BalancingOption* MQBNO_TIMEOUT_IMMEDIATE.

VALORE PREDEFINITO
Il timeout dell'applicazione ai fini del ribilanciamento in un cluster uniforme dopo il periodo predefinito di 10 secondi.

Questo valore si associa a IBM MQ *BalancingOption* MQBNO_TIMEOUT_AS_DEFAULT.

nn
Il timeout dell'applicazione ai fini del ribilanciamento in un cluster uniforme dopo un periodo di *nn* secondi.

nn può essere compreso tra 1 e 9999999999.

BROKERCCDURSUBQ

Il nome della coda da cui vengono richiamati i messaggi di sottoscrizione durevoli per un ConnectionConsumer.

Oggetti applicabili

Argomento

Nome esteso dello strumento di gestione JMS : BROKERCCDURSUBQ

Nome breve dello strumento di amministrazione JMS : CCDSUB

Accesso programmatico

Setter / getter

- MQTopic.setBrokerCCDurSubQueue()
- MQTopic.getBrokerCCDurSubQueue()

Valori

SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE
Questo è il valore predefinito.

Qualsiasi stringa valida

BROKERCCSUBQ

Il nome della coda da cui vengono richiamati i messaggi di sottoscrizione non durevoli per un ConnectionConsumer.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : BROKERCCSUBQ

Nome breve dello strumento di amministrazione JMS : CCSUB

Accesso programmatico

Setter / getter

- MQConnectionFactory.setBrokerCCSubQueue()
- MQConnectionFactory.getBrokerCCSubQueue()

Valori

SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE
Questo è il valore predefinito.

Qualsiasi stringa valida

BROKERCONQ

Il nome della coda di controllo del broker.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : BROKER

Nome breve dello strumento di amministrazione JMS : BCON

Accesso programmatico

Setter / getter

- MQConnectionFactory.setBrokerControlQueue()
- MQConnectionFactory.getBrokerControlQueue()

Valori

SYSTEM.BROKER.CONTROL.QUEUE
Questo è il valore predefinito.

Qualsiasi stringa valida

BROKERDURSUBQ

Quando IBM MQ classes for JMS viene utilizzato in modalità di migrazione del fornitore di messaggistica IBM MQ , questa proprietà specifica il nome della coda da cui vengono richiamati i messaggi di sottoscrizione durevoli.

Oggetti applicabili

Argomento

Nome esteso dello strumento di gestione JMS : BROKERDURSUBQ

Nome breve dello strumento di amministrazione JMS : BDSUB

Accesso programmatico

Setter / getter

- MQTopic.setBrokerDurSubQueue()
- MQTopic.getBrokerDurSubQueue()

Valori

SYSTEM.JMS.D.SUBSCRIBER.QUEUE
Questo è il valore predefinito.

Qualsiasi stringa valida
Avvio con SYSTEM.JMS.D

Attività correlate

Configurazione della proprietà JMS **PROVIDERVERSION**

BROKERPUBQ

Il nome della coda in cui vengono inviati i messaggi pubblicati (la coda di flusso).

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, Topic, XAConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di amministrazione JMS : BROKERPUBQ

Nome breve dello strumento di amministrazione JMS : BPUB

Accesso programmatico

Setter / getter

- MQConnectionFactory.setBrokerPubQueue
- MQConnectionFactory.getBrokerPubQueue

Valori

SYSTEM.BROKER.DEFAULT.STREAM
Questo è il valore predefinito.

Qualsiasi stringa valida

BROKERPUBQMGR

Il nome del gestore code proprietario della coda in cui vengono inviati i messaggi pubblicati sull'argomento.

Oggetti applicabili

Argomento

Nome completo dello Strumento di gestione JMS : BROKERPUBQMGR

Nome breve dello strumento di amministrazione JMS : BPQM

Accesso programmatico

Setter / getter

- MQTopic.setBrokerPubQueueManager()
- MQTopic.getBrokerPubQueueManager()

Valori

vuoto
Questo è il valore predefinito.

Qualsiasi stringa valida

BROKERQMGR

Il nome del gestore code su cui è in esecuzione il broker.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nome completo dello strumento di gestione JMS : BROKERQMGR

Nome breve dello strumento di amministrazione JMS : BQM

Accesso programmatico

Setter / getter

- MQConnectionFactory.setBrokerQueueManager()
- MQConnectionFactory.getBrokerQueueManager()

Valori

vuoto
Questo è il valore predefinito.

Qualsiasi stringa valida

BROKERSUBQ

Quando IBM MQ classes for JMS viene utilizzato in modalità di migrazione del provider dei messaggi IBM MQ , questa proprietà specifica il nome della coda da cui vengono richiamati i messaggi di sottoscrizione non durevoli.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nome esteso dello Strumento di gestione JMS : BROKERSUBQ

Nome breve dello strumento di amministrazione JMS : BSUB

Accesso programmatico

Setter / getter

- MQConnectionFactory.setBrokerSubQueue()
- MQConnectionFactory.getBrokerSubQueue()

Valori

SYSTEM.JMS.ND.SUBSCRIBER.QUEUE
Questo è il valore predefinito.

Qualsiasi stringa valida
Avvio con SYSTEM.JMS.ND

Attività correlate
Configurazione della proprietà JMS **PROVIDERVERSION**

BROKERVER

La versione del broker utilizzato.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, Topic, XAConnectionFactory, XATopicConnectionFactory

Nome esteso dello Strumento di gestione JMS : BROKER

Nome breve dello strumento di amministrazione JMS : BVER

Accesso programmatico

Setter / getter

- MQConnectionFactory.setBrokerVersione ()
- MQConnectionFactory.getBrokerVersione ()

Valori

V1
Per utilizzare un broker di pubblicazione / sottoscrizione IBM MQ o un broker di IBM MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker o WebSphere Business Integration Message Broker in modalità di compatibilità. Questo è il valore predefinito se TRANSPORT è impostato su BIND o CLIENT.

V2
Per utilizzare un broker di IBM MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker o WebSphere Business Integration Message Broker in modalità nativa. Questo è il valore predefinito se TRANSPORT è impostato su DIRECT o DIRECTHTTP.

non specificato
Una volta eseguita la migrazione del broker da V6 a V7, impostare questa proprietà in modo che le intestazioni RFH2 non vengano più utilizzate. Dopo la migrazione, questa proprietà non è più rilevante.

CCDTURL

Un URL (Uniform Resource Locator) che identifica il nome e l'ubicazione del file contenente la tabella di definizione del canale client e specifica come è possibile accedere al file.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : CCDTURL

Nome breve dello strumento di amministrazione JMS : CCDT

Accesso programmatico

Setter / getter

- MQConnectionFactory.setCCDTURL()
- MQConnectionFactory.getCCDTURL()

Valori

vuoto
Questo è il valore predefinito.

URL (Uniform Resource Locator)

CCSID

Per le produzioni connessioni, questa proprietà specifica il CCSID (coded character set ID) da utilizzare per i flussi di dati interni con il gestore code. Per le destinazioni, la proprietà definisce il CCSID da utilizzare per codificare i dati stringa in MapMessages, StreamMessagese TextMessages inseriti in tale destinazione.

Nota: Normalmente non è necessario modificare questa proprietà per le factory di connessione.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : CCSID

Nome breve di JMS Administration Tool: CCS

Accesso programmatico

Setter / getter

- MQConnectionFactory.setCCSID()
- MQConnectionFactory.getCCSID()

Valori

819
Il valore predefinito per una produzione connessioni.

1208
Il valore predefinito per una destinazione.

Qualsiasi numero intero positivo

Concetti correlati
JMS Conversione del messaggio

V 9.4.0 V 9.4.0 CERTVALPO

La politica di convalida del certificato da utilizzare per una connessione TLS.

Oggetti applicabili

ConnectionFactory

Nome completo strumento di amministrazione JMS : CERTVALPO

Nome breve dello strumento di amministrazione JMS : CVAP

Accesso programmatico

Setter / getter

- MQConnectionFactory.setCertificateValPolicy()
- MQConnectionFactory.getCertificateValPolicy()

Valori

0/ANY
Questo è il valore predefinito per una produzione connessioni.

2 / NONE
Questa normativa non include la validazione del certificato. Può essere impostato solo su applicazioni client.

CHANNEL

Il nome del canale di connessione client utilizzato.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome completo strumento di amministrazione JMS : CHANNEL

Nome breve dello strumento di amministrazione JMS : CHAN

Accesso programmatico

Setter / getter

- MQConnectionFactory.setChannel()
- MQConnectionFactory.getChannel()

Valori

SYSTEM.DEF.SVRCONN
Questo è il valore predefinito.

Qualsiasi stringa valida

CLEANUP

Livello di ripulitura per gli archivi di sottoscrizioni BROKER o MIGRATE.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nome completo strumento di gestione JMS : CLEANUP

Nome breve dello strumento di amministrazione JMS : CL

Accesso programmatico

Setter / getter

- MQConnectionFactory.setCleanupLivello ()
- MQConnectionFactory.getCleanupLivello ()

Valori

SICURA
Utilizzare il cleanup sicuro. Questo è il valore predefinito.

PROPAS
Utilizzare sicuro, forte o nessuna ripulitura in base a una proprietà impostata sulla riga comandi Java .

Nessuna
Non utilizzare ripulitura.

forte
Utilizzare il cleanup forte.

CLEANUPINT

L'intervallo, in millisecondi, tra le esecuzioni in background del programma di utilità di ripulitura di pubblicazione / sottoscrizione.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : CLEANUPINT

Nome breve dello strumento di amministrazione JMS : CLINT

Accesso programmatico

Setter / getter

- MQConnectionFactory.setCleanupIntervallo ()
- MQConnectionFactory.getCleanupInterval ()

Valori

3600000
Questo è il valore predefinito.

Qualsiasi numero intero positivo

ConnectionNameList

Elenco dei nomi di connessione TCP/IP. L'elenco viene tentato in ordine, una volta per ogni nuovo tentativo di riconnessione.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Nome esteso dello strumento di gestione JMS : CONNECTIONNAMELIST

Nome breve dello strumento di amministrazione JMS : CNLIST

Accesso programmatico

Setter / getter

- MQConnectionFactory.setconnectionNameElenco ()
- MQConnectionFactory.getconnectionNameList ()

Valori

Elenco separato da virgole di HOSTNAME (PORT). HOSTNAME può essere un nome DNS o un indirizzo IP.

Il valore predefinito di PORT è 1414.

CLIENTRECONNECTOPTIONS

Opzioni che gestiscono la riconnessione.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Nome esteso dello strumento di gestione JMS : CLIENTRECONNECTOPTIONS

Nome breve dello strumento di amministrazione JMS : CROPT

Accesso programmatico

Setter / getter

- MQConnectionFactory.setClientReconnectOptions()
- MQConnectionFactory.getClientReconnectOptions()

Valori

QMGR

L'applicazione può riconnettersi allo stesso gestore code a cui è originariamente connessa.

Viene restituito un errore con codice motivo `MQRC_RECONNECT_QMID_MISMATCH` se il gestore code a cui l'applicazione tenta di connettersi, come specificato nell'elenco di nomi di connessioni, ha un QMID diverso rispetto al gestore code a cui si è originariamente connessa.

Utilizzare questo valore se un'applicazione può essere riconnessa, ma c'è un'affinità tra l'applicazione IBM MQ classes for JMS e il gestore code a cui ha stabilito per la prima volta una connessione.

Scegliere questo valore se si desidera che un'applicazione si riconnetti automaticamente all'istanza in standby di un gestore code ad alta disponibilità.

Per utilizzare questo valore in modo programmatico, utilizzare la costante WMQConstants.WMQ_CLIENT_RECONNECT_Q_MGR.

ANY

L'applicazione può riconnettersi a qualsiasi gestore code specificato nell'elenco dei nomi di connessione.

Utilizzare l'opzione di riconnessione solo se non esiste alcuna affinità tra le classi IBM MQ per l'applicazione JMS e il gestore code con cui ha inizialmente stabilito una connessione.

Per utilizzare questo valore da un programma, utilizzare la costante WMQConstants.WMQ_CLIENT_RECONNECT.

DISABILITATO

L'applicazione non verrà riconnessa.

Per utilizzare questo valore in modo programmatico, utilizzare la costante WMQConstants.WMQ_CLIENT_RECONNECT_DISABLED.

ASDEF

Se l'applicazione si riconnetterà automaticamente dipende dal valore dell'attributo del canale IBM MQ DefReconnect.

Questo è il valore predefinito.

Per utilizzare questo valore da un programma, utilizzare la costante WMQConstants.WMQ_CLIENT_RECONNECT_AS_DEF.

CLIENTRECONNECTTIMEOUT

Tempo prima che cessino i tentativi di riconnessione.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Nome esteso dello strumento di gestione JMS : CLIENTRECONNECTTIMEOUT

Nome breve dello strumento di amministrazione JMS : CRT

Accesso programmatico

Setter / getter

- MQConnectionFactory.setClientReconnectTimeout()
- MQConnectionFactory.setClientReconnectTimeout()

Valori

Intervallo in secondi. Valore predefinito 1800 (30 minuti).

CLIENTID

L'identificatore client viene utilizzato per identificare in modo univoco la connessione dell'applicazione per sottoscrizioni durevoli.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso strumento di amministrazione JMS : CLIENTID

Nome breve dello strumento di amministrazione JMS : CID

Accesso programmatico

Setter / getter

- MQConnectionFactory.setClientId ()
- MQConnectionFactory.getClientId ()

Valori

vuoto
Questo è il valore predefinito.

Qualsiasi stringa valida

CLONESUPP

Se due o più istanze dello stesso sottoscrittore di argomenti durevoli possono essere eseguite contemporaneamente.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nome completo dello strumento di gestione JMS : CLONESUPP

Nome breve dello strumento di amministrazione JMS : CLS

Accesso programmatico

Setter / getter

- MQConnectionFactory.setCloneSupport ()
- MQConnectionFactory.getCloneSupport ()

Valori

DISABILITATO
È possibile eseguire solo un'istanza di un sottoscrittore di argomenti durevoli alla volta. Questo è il valore predefinito.

Abilitato
Due o più istanze dello stesso sottoscrittore di argomenti durevoli possono essere eseguite contemporaneamente, ma ciascuna istanza deve essere eseguita in una JVM (Java virtual machine) separata.

COMPHDR

Un elenco delle tecniche che è possibile utilizzare per comprimere i dati di intestazione su una connessione.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : COMPHDR

Nome breve dello strumento di amministrazione JMS : HC

Accesso programmatico

Setter / getter

- MQConnectionFactory.setHdrCompList()
- MQConnectionFactory.getHdrCompList()

Valori

Nessuna
Questo è il valore predefinito.

SISTEMA
Viene eseguita la compressione dell'intestazione del messaggio RLE.

COMPMSG

Un elenco delle tecniche che è possibile utilizzare per comprimere i dati del messaggio su una connessione.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di amministrazione JMS : COMPMSG

Nome breve dello strumento di amministrazione JMS : MC

Accesso programmatico

Setter / getter

- MQConnectionFactory.setMsgCompList()
- MQConnectionFactory.getMsgCompList()

Valori

Nessuna
Questo è il valore predefinito.

Un elenco di uno o più dei seguenti valori separati da caratteri vuoti:
V 9.4.0 LZ4FAST LZ4HIGH

CONNOPT

Controlla il modo in cui le applicazioni IBM MQ classes for JMS che utilizzano il trasporto bind si collegano al gestore code.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory.

Nome completo dello strumento di amministrazione di JMS : CONNOPT

Nome breve dello strumento di amministrazione JMS : CNOPT

Accesso programmatico

Setter / getter

- MQConnectionFactory.setMQConnectionOpzioni ()
- MQConnectionFactory.getMQConnectionOpzioni ()

Valori

STANDARD
La natura del collegamento tra l'applicazione e il gestore code dipende dal valore dell'attributo *DefaultBindTipo* del gestore code. Il valore STANDARD è associato a IBM MQ *ConnectOption* MQCNO_STANDARD_BINDING.

CONDIVISO
L'applicazione e il gestore code locale vengono eseguiti in unità di esecuzione separate, ma condividono alcune risorse. Questo valore viene associato a IBM MQ *ConnectOption* MQCNO_SHARED_BINDING.

isolato
L'applicazione e il gestore code locale vengono eseguiti in unità di esecuzione separate e non condividono alcuna risorsa. Il valore ISOLATO viene associato a IBM MQ *ConnectOption* MQCNO_ISOLATED_BINDING.

Percorso veloce
L'applicazione e il gestore code locale vengono eseguiti nella stessa unità di esecuzione. Questo valore viene associato a IBM MQ *ConnectOption* MQCNO_FASTPATH_BINDING.

SERIALQM
L'applicazione richiede l'uso esclusivo della tag di connessione nell'ambito del gestore code. Questo valore è associato a IBM MQ *ConnectOption* MQCNO_SERIALIZE_CONN_TAG_Q_MGR.

SERIALQSG
L'applicazione richiede l'uso esclusivo della tag di connessione nell'ambito del gruppo di condivisione code a cui appartiene il gestore code. Il valore SERIALQSG viene associato a IBM MQ *ConnectOption* MQCNO_SERIALIZE_CONN_TAG_QSG.

LIMITATOQM
L'applicazione richiede l'uso condiviso della tag di connessione, ma esistono limitazioni sull'uso condiviso della tag di connessione nell'ambito del gestore code. Questo valore è associato a IBM MQ *ConnectOption* MQCNO_RESTRICT_CONN_TAG_Q_MGR.

RISTRICTQSG
L'applicazione richiede l'uso condiviso della tag di connessione, ma esistono delle restrizioni sull'uso condiviso della tag di connessione nell'ambito del gruppo di condivisione code a cui appartiene il gestore code. Questo valore è associato a IBM MQ *ConnectOption* MQCNO_RESTRICT_CONN_TAG_QSG.

Per ulteriori informazioni relative alle opzioni di connessione IBM MQ , fare riferimento a Connessione a un gestore code mediante la chiamata MQCONNX.

CONNTAG

Una tag che il gestore code associa alle risorse aggiornate dall'applicazione in un'unità di lavoro mentre l'applicazione è connessa al gestore code.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : CONNTAG

Nome breve dello strumento di amministrazione JMS : CNTAG

Accesso programmatico

Setter / getter

- MQConnectionFactory.setConnTag ()
- MQConnectionFactory.getConnTag ()

Valori

Un array di byte di 128 elementi, dove ogni elemento è 0
Questo è il valore predefinito.

Qualsiasi stringa
Il valore viene troncato se supera i 128 byte.

DESCRIZIONE

Una descrizione dell'oggetto memorizzato.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : DESCRIPTION

Nome breve dello strumento di amministrazione JMS : DESC

Accesso programmatico

Setter / getter

- MQConnectionFactory.setDescription()
- MQConnectionFactory.getDescription()

Valori

vuoto
Questo è il valore predefinito.

Qualsiasi stringa valida
`

DIRECTAUTH

Se l'autenticazione TLS viene utilizzata su una connessione in tempo reale ad un broker.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory

Nome esteso strumento di amministrazione JMS : DIRECTAUTH

Nome breve dello strumento di amministrazione JMS : DAUTH

Accesso programmatico

Setter / getter

- MQConnectionFactory.setDirectAuth ()
- MQConnectionFactory.getDirectAuth ()

Valori

BASE
Nessuna autenticazione, autenticazione nome utente o autenticazione password. Questo è il valore predefinito.

CERTIFICATO
Autenticazione certificato chiave pubblica.

Codifica

Modalità di rappresentazione dei dati numerici nel corpo di un messaggio quando il messaggio viene inviato a questa destinazione. La proprietà specifica la rappresentazione di numeri interi binari, interi decimali compressi e numeri a virgola mobile.

Oggetti applicabili

Coda, Argomento

Nome completo dello strumento di amministrazione JMS : ENCODING

Nome breve dello strumento di amministrazione JMS : ENC

Accesso programmatico

Setter / getter

- MQDestination.setEncoding()

- MQDestination.getEncoding()

Valori

proprietà ENCODING

I valori validi che la proprietà ENCODING può assumere sono creati dalle tre proprietà secondarie:

codifica a numero intero
Normale o invertito

codifica a numero decimale
Normale o invertito

codifica a virgola mobile
IEEE normale, IEEE invertito o z/OS

La proprietà ENCODING è espressa come una stringa di tre caratteri con la sintassi seguente:

```
{N|R}{N|R}{N|R|3}
```

In questa stringa:

- N indica normale
- R indica l'inversione
- 3 indica z/OS
- Il primo carattere rappresenta la *codifica numero intero*
- Il secondo carattere rappresenta la *codifica decimale*
- Il terzo carattere rappresenta la *codifica a virgola mobile*

Fornisce una serie di dodici valori possibili per la proprietà ENCODING .

Esiste un ulteriore valore, la stringa NATIVE, che imposta i valori di codifica appropriati per la piattaforma Java .

I seguenti esempi mostrano combinazioni valide per ENCODING:

```
ENCODING(NNR)
ENCODING(NATIVE)
ENCODING(RR3)
```

EXPIRY

L'ora dopo la quale scadono i messaggi in una destinazione.

Oggetti applicabili

Coda, Argomento

Nome esteso dello strumento di gestione JMS : SCADENZA

Nome breve dello strumento di amministrazione JMS : EXP

Accesso programmatico

Setter / getter

- MQDestination.setExpiry()
- MQDestination.getExpiry()

Valori

APP
La scadenza può essere definita dall'applicazione JMS . Questo è il valore predefinito.

UNLIM
Non si verifica alcuna scadenza.

0
Non si verifica alcuna scadenza.

Qualsiasi numero intero positivo che rappresenta la scadenza in millisecondi.

FAILIFQUIESCE

Questa proprietà determina se le chiamate a determinati metodi hanno esito negativo se il gestore code è in uno stato di sospensione o se un'applicazione si sta connettendo a un gestore code utilizzando il trasporto CLIENT e il canale utilizzato dall'applicazione è stato impostato in uno stato di sospensione, ad esempio utilizzando il comando **STOP CHANNEL** o **STOP CHANNEL MODE(QUIESCE)** MQSC .

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : FAILIFQUIESCE

Nome breve dello strumento di amministrazione JMS : FIQ

Accesso programmatico

Setter / getter

- MQConnectionFactory.setFailIfQuiesce()
- MQConnectionFactory.getFailIfQuiesce()

Valori

SÌ
Le chiamate a determinati metodi non riescono se il gestore code è in stato di sospensione o se il canale utilizzato per connettersi a un gestore code è in fase di sospensione. Se un'applicazione rileva una di queste condizioni, può completare l'attività immediata e chiudere la connessione, consentendo l'arresto del gestore code o dell'istanza del canale. Questo è il valore predefinito.

No
Nessuna chiamata al metodo ha esito negativo perché il gestore code o il canale utilizzato per connettersi a un gestore code si trova in uno stato di inattività. Se si specifica questo valore, un'applicazione non è in grado di rilevare che il gestore code o il canale è in fase di sospensione. L'applicazione potrebbe continuare ad eseguire operazioni sul gestore code e quindi impedire l'arresto del gestore code.

HOSTNAME

Per una connessione a un gestore code, il nome host o l'indirizzo IP del sistema su cui è in esecuzione il gestore code o, per una connessione in tempo reale a un broker, il nome host o l'indirizzo IP del sistema su cui è in esecuzione il broker.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : HOSTNAME

Nome breve dello strumento di amministrazione JMS : HOST

Accesso programmatico

Setter / getter

- MQConnectionFactory.setHost()
- MQConnectionFactory.getHostNome ()

Valori

host locale
Questo è il valore predefinito.

Qualsiasi stringa valida

LOCALADDRESS

Per una connessione a un gestore code, questa proprietà specifica l'interfaccia di rete locale da utilizzare o la porta locale o l'intervallo di porte locali da utilizzare.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome completo dello Strumento di gestione JMS : LOCALADDRESS

Nome breve dello strumento di amministrazione JMS : LA

Accesso programmatico

Setter / getter

- MQConnectionFactory.setLocalAddress ()
- MQConnectionFactory.getLocalAddress ()

Valori

"" (stringa vuota)
Questo è il valore predefinito.

Una stringa nel formato [ip-addr] [(low-port [, high-port])]
Di seguito sono riportati alcuni esempi:

`192.0.2.0`

Il canale si collega all'indirizzo 192.0.2.0 localmente.

`192.0.2.0(1000)`

Il canale esegue il bind all'indirizzo 192.0.2.0 localmente e utilizza la porta 1000.

`192.0.2.0(1000,2000)`

Il canale si collega all'indirizzo 192.0.2.0 localmente e utilizza una porta compresa tra 1000 e 2000.

`(1000)`

Il canale si collega alla porta 1000 localmente.

`(1000,2000)`

Il canale si collega a una porta nell'intervallo compreso tra 1000 e 2000 localmente.

È possibile indicare un nome host invece di un indirizzo IP. Per una connessione in tempo reale a un broker, questa proprietà è rilevante solo quando si utilizza il multicast e il valore della proprietà non deve contenere un numero di porta o un intervallo di numeri di porta. Gli unici valori validi della proprietà in questo caso sono null, un indirizzo IP o un nome host.

NOMEMAPPA

Consente lo stile di compatibilità da utilizzare per i nomi elemento MapMessage .

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome completo dello strumento di gestione JMS : MAPNAMESTYLE

Nome breve dello strumento di amministrazione JMS : MNST

Accesso programmatico

Setter / getter

- MQConnectionFactory.setMapNameStyle()
- MQConnectionFactory.getMapNameStyle()

Valori

STANDARD
È necessario utilizzare il formato di denominazione elemento com.ibm.jms.JMSMapMessage standard. Questo è il valore predefinito e consente l'utilizzo di identificatori Java non legali come nome elemento.

Compatibile
Deve essere utilizzato il vecchio formato di denominazione dell'elemento com.ibm.jms.JMSMapMessage . Solo gli identificatori Java legali possono essere utilizzati come nome elemento. Ciò è necessario solo se i messaggi della mappa vengono inviati a un'applicazione che utilizza una versione di IBM MQ classes for JMS precedente a 5.3.

MAXBUFFSIZE

Il numero massimo di messaggi ricevuti che possono essere memorizzati in un buffer di messaggi interno in attesa di essere elaborati dall'applicazione. Questa proprietà si applica solo quando TRANSPORT ha il valore DIRECT o DIRECTHTTP.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory

Nome esteso dello strumento di gestione JMS : MAXBUFFSIZE

Nome breve dello strumento di amministrazione JMS : MBSZ

Accesso programmatico

Setter / getter

- MQConnectionFactory.setMaxBufferSize()
- MQConnectionFactory.getMaxBufferSize()

Valori

1000
Questo è il valore predefinito.

Qualsiasi numero intero positivo

MDREAD

Questa proprietà determina se un'applicazione JMS può estrarre i valori dei campi MQMD.

Oggetti applicabili

Nome completo strumento di amministrazione JMS : MDREAD

Nome breve dello strumento di amministrazione JMS : MDR

Accesso programmatico

Setter / getter

- MQDestination.setMQMDReadEnabled()
- MQDestination.getMQMDReadEnabled()

Valori

No
Quando si inviano messaggi, le proprietà JMS_IBM_MQMD* di un messaggio inviato non vengono aggiornate per riflettere i valori dei campi aggiornati in MQMD. Alla ricezione dei messaggi, nessuna delle proprietà JMS_IBM_MQMD* è disponibile sul messaggio ricevuto, anche se il mittente ne ha impostate alcune o tutte. Questo è il valore predefinito per gli strumenti di gestione.

Per i programmi, utilizzare False.

Sì
Quando si inviano messaggi, tutte le proprietà JMS_IBM_MQMD* di un messaggio inviato vengono aggiornate per riflettere i valori dei campi aggiornati in MQMD, incluse le proprietà che il mittente non ha impostato esplicitamente. Quando si ricevono messaggi, tutte le proprietà JMS_IBM_MQMD* sono disponibili su un messaggio ricevuto, incluse le proprietà che il mittente non ha impostato esplicitamente.

Per i programmi, utilizzare True.

MDWRITE

Questa proprietà determina se un'applicazione JMS può impostare i valori dei campi MQMD.

Oggetti applicabili

Coda, Argomento

Nome esteso dello strumento di gestione JMS : MDWRITE

Nome breve dello strumento di amministrazione JMS : MDR

Accesso programmatico

Setter / getter

- MQDestination.setMQMDWriteEnabled()
- MQDestination.getMQMDWriteEnabled()

Valori

No
Tutte le proprietà JMS_IBM_MQMD* vengono ignorate e i relativi valori non vengono copiati nella struttura MQMD sottostante. Questo è il valore predefinito per gli strumenti di gestione.

Per i programmi, utilizzare False.

SÌ
Le proprietà JMS_IBM_MQMD* vengono elaborate. I loro valori vengono copiati nella struttura MQMD sottostante.

Per i programmi, utilizzare True.

MDMSGCTX

Quale livello di contesto del messaggio deve essere impostato dall'applicazione JMS . L'applicazione deve essere in esecuzione con l'autorizzazione di contesto appropriata perché questa proprietà diventi effettiva.

Oggetti applicabili

Nome esteso dello strumento di gestione JMS : MDMSGCTX

Nome breve dello strumento di amministrazione JMS : MDCTX

Accesso programmatico

Setter / getter

- MQDestination.setMQMDMessageContext()
- MQDestination.getMQMDMessageContext()

Valori

VALORE PREDEFINITO
La chiamata API MQOPEN e la struttura MQPM non specificano opzioni di contesto del messaggio esplicite. Questo è il valore predefinito per gli strumenti di gestione.

Per programmi, utilizzare WMQ_MDCTX_DEFAULT.

SET_IDENTITY_CONTEXT
La chiamata API MQOPEN specifica l'opzione di contesto del messaggio MQOO_SET_IDENTITY_CONTEXT e la struttura di MQPMO specifica MQPMO_SET_IDENTITY_CONTEXT.

Per programmi, utilizzare WMQ_MDCTX_SET_IDENTITY_CONTEXT.

IMPOSTA_TUTTI_CONTESTO
La chiamata API MQOPEN specifica l'opzione di contesto del messaggio MQOO_SET_ALL_CONTEXT e la struttura MQPMO specifica MQPMO_SET_ALL_CONTEXT.

Per i programmi, utilizzare WMQ_MDCTX_SET_ALL_CONTEXT.

MSGBATCHSZ

Il numero massimo di messaggi da prendere da una coda in un pacchetto quando si utilizza la consegna asincrona dei messaggi.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : MAXBUFFSIZE

Nome breve dello strumento di amministrazione JMS : MBSZ

Accesso programmatico

Setter / getter

- MQConnectionFactory.setMsgBatchSize()
- MQConnectionFactory.getMsgBatchSize()

Valori

10
Questo è il valore predefinito.

Qualsiasi numero intero positivo

MSGBODY

Determina se un'applicazione JMS accede a MQRFH2 di un messaggio IBM MQ come parte del payload del messaggio.

Oggetti applicabili

Coda, Argomento

Nome completo dello strumento di gestione di JMS : WMQ_MESSAGE_BODY

Nome breve dello strumento di amministrazione JMS : MBODY

Accesso programmatico

Setter / getter

- MQConnectionFactory.setMessageBodyStyle()
- MQConnectionFactory.getMessageBodyStyle()

Valori

NON SPECIFICATO
Durante l'invio, IBM MQ classes for JMS genera o non genera e include un'intestazione MQRFH2 , in base al valore di WMQ_TARGET_CLIENT. Durante la ricezione, agisce come valore JMS.

JMS
Durante l'invio, IBM MQ classes for JMS genera automaticamente un'intestazione MQRFH2 e la include nel messaggio IBM MQ .

Quando si riceve, IBM MQ classes for JMS imposta le proprietà del messaggio JMS in base ai valori in MQRFH2 (se presente); non presenta MQRFH2 come parte del corpo del messaggio JMS .

MQ
Durante l'invio, IBM MQ classes for JMS non genera un MQRFH2.

Quando si riceve, IBM MQ classes for JMS presenta MQRFH2 come parte del contenuto del messaggio JMS .

MSGRETENTION

Se il consumer della connessione conserva i messaggi non recapitati nella coda di input.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory,

Nome esteso dello strumento di amministrazione JMS : MSGRETENTION

Nome breve dello strumento di amministrazione JMS : MRET

Accesso programmatico

Setter / getter

- MQConnectionFactory.setMessageRetention ()
- MQConnectionFactory.getMessageRetention ()

Valori

Sì
I messaggi non recapitati rimangono nella coda di immissione. Questo è il valore predefinito.

No
I messaggi non recapitati vengono gestiti in base alle opzioni di disposizione.

MSGSELECTION

Determina se la selezione del messaggio viene effettuata da IBM MQ classes for JMS o dal broker. Se TRANSPORT ha il valore DIRECT, la selezione del messaggio viene sempre effettuata dal broker e il valore di MSGSELECTION viene ignorato. La selezione dei messaggi da parte di un broker non è supportata quando BROKER ha il valore V1.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di amministrazione JMS : MSGSELECTION

Nome breve dello strumento di amministrazione JMS : MSEL

Accesso programmatico

Setter / getter

- MQConnectionFactory.setMessageSelezione ()
- MQConnectionFactory.getMessageSelezione ()

Valori

CLIENTE
La selezione del messaggio viene effettuata tramite IBM MQ classes for JMS. Questo è il valore predefinito.

BROKER
La selezione del messaggio viene effettuata dal broker.

MULTICAST

Per abilitare il multicast su una connessione in tempo reale ad un broker e, se abilitato, per specificare il modo preciso in cui il multicast viene utilizzato per consegnare i messaggi dal broker ad un consumatore di messaggi. La proprietà non ha alcun effetto sul modo in cui un produttore di messaggi invia i messaggi a un broker.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, Argomento

Nome esteso dello strumento di gestione JMS : MULTICAST

Nome breve dello strumento di amministrazione JMS : MCAST

Accesso programmatico

Setter / getter

- MQConnectionFactory.setMulticast()
- MQConnectionFactory.getMulticast()

Valori

DISABILITATO
I messaggi non vengono consegnati a un consumatore di messaggi utilizzando il trasporto multicast. Questo è il valore predefinito per gli oggetti ConnectionFactory e TopicConnectionFactory.

ASCF
I messaggi vengono consegnati ad un consumatore di messaggi in base all'impostazione multicast per la produzione connessioni associata al consumatore di messaggi. L'impostazione multicast per il factory di connessione viene annotata al momento della creazione del consumer del messaggio. Questo valore è valido solo per gli oggetti Argomento ed è il valore predefinito per gli oggetti Argomento.

Abilitato
Se l'argomento è configurato per il multicast nel broker, i messaggi vengono consegnati a un consumatore di messaggi utilizzando il trasporto multicast. Viene utilizzata una QoS (quality of service) affidabile se l'argomento è configurato per il multicast affidabile.

Affidabile
Se l'argomento è configurato per il multicast affidabile nel broker, i messaggi vengono consegnati al consumatore di messaggi utilizzando il trasporto multicast con una qualità del servizio affidabile. Se l'argomento non è configurato per il multicast affidabile, non è possibile creare un consumer di messaggi per l'argomento.

NOTR
Se l'argomento è configurato per multicast nel broker, i messaggi vengono consegnati al consumatore di messaggi utilizzando il trasporto multicast. Una QoS (quality of service) affidabile non viene utilizzata anche se l'argomento è configurato per il multicast affidabile.

OPTIMISTICPUBLICATION

Questa proprietà determina se IBM MQ classes for JMS restituisce il controllo immediatamente a un publisher che ha pubblicato un messaggio o se restituisce il controllo solo dopo aver completato tutta l'elaborazione associata alla chiamata e può notificare il risultato al publisher.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory

Nome esteso dello strumento di gestione JMS : OTTIMISTICPUBLICATION

Nome breve dello strumento di amministrazione JMS : OPTPUB

Accesso programmatico

Setter / getter

- MQConnectionFactory.setOptimisticPubblicazione ()
- MQConnectionFactory.getOptimisticPublication ()

Valori

No
Quando un publisher pubblica un messaggio, IBM MQ classes for JMS non restituisce il controllo al publisher fino a quando non ha completato tutta l'elaborazione associata alla chiamata e non può notificare il risultato al publisher. Questo è il valore predefinito.

SÌ
Quando un publisher pubblica un messaggio, IBM MQ classes for JMS restituisce il controllo al publisher immediatamente, prima di aver completato tutta l'elaborazione associata con la chiamata e può notificare il risultato al publisher. IBM MQ classes for JMS riporta il risultato solo quando il publisher esegue il commit del messaggio.

OUTCOMENOTIFICATION

Questa proprietà determina se IBM MQ classes for JMS restituisce il controllo immediatamente a un sottoscrittore che ha appena riconosciuto o eseguito il commit di un messaggio, oppure se restituisce il controllo solo dopo aver completato tutta l'elaborazione associata alla chiamata e se può notificare il risultato al sottoscrittore.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory

Nome esteso dello strumento di gestione JMS : OUTCOMENOATION

Nome breve dello strumento di amministrazione JMS : NOTIFY

Accesso programmatico

Setter / getter

- MQConnectionFactory.setOutcomeNotifica ()
- MQConnectionFactory.getOutcomeNotification ()

Valori

SÌ
Quando un sottoscrittore riconosce o esegue il commit di un messaggio, IBM MQ classes for JMS non restituisce il controllo al sottoscrittore fino a quando non ha completato tutta l'elaborazione associata alla chiamata e può notificare il risultato al sottoscrittore. Questo è il valore predefinito.

No
Quando un sottoscrittore riconosce o esegue il commit di un messaggio, IBM MQ classes for JMS restituisce immediatamente il controllo al sottoscrittore, prima di aver completato tutta l'elaborazione associata alla chiamata e può notificare il risultato al sottoscrittore.

PERSISTENCE

La persistenza dei messaggi inviati ad una destinazione.

Oggetti applicabili

Coda, Argomento

JMS nome lungo dello strumento di somministrazione: PERSISTENCE

Nome breve dello strumento di amministrazione JMS : PER

Accesso programmatico

Setter / getter

- MQDestination.setPersistence()
- MQDestination.getPersistence()

Valori

APP
La persistenza è definita dall'applicazione JMS . Questo è il valore predefinito.

QDEF
La persistenza assume il valore predefinito della coda.

PER
I messaggi sono persistenti.

NON
I messaggi non sono persistenti.

ALTO
Consultare JMS messaggi persistenti per ulteriori informazioni sull'utilizzo di questo valore.

POLLINGINT

Se ciascun listener di messaggi all'interno di una sessione non dispone di un messaggio adatto sulla propria coda, questo è l'intervallo massimo, in millisecondi, che trascorre prima che ciascun listener di messaggi tenti nuovamente di ottenere un messaggio dalla propria coda. Se si verifica spesso che non è disponibile alcun messaggio adatto per nessuno dei listener di messaggi in una sessione, aumentare il valore di questa proprietà. Questa proprietà è rilevante solo se TRANSPORT ha il valore BIND o CLIENT.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : POLLINGINT

Nome breve dello strumento di amministrazione JMS : PINT

Accesso programmatico

Setter / getter

- MQConnectionFactory.setPollingIntervallo ()
- MQConnectionFactory.getPollingIntervallo ()

Valori

5000
Questo è il valore predefinito.

Qualsiasi numero intero positivo

PORTA

Per una connessione ad un gestore code, il numero della porta su cui il gestore code è in attesa oppure, per una connessione in tempo reale ad un broker, il numero della porta su cui il broker è in ascolto per le connessioni in tempo reale.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : PORT

Nome breve dello strumento di amministrazione JMS : PORT

Accesso programmatico

Setter / getter

- MQConnectionFactory.setPort()
- MQConnectionFactory.getPort()

Valori

1414
Questo è il valore predefinito se TRANSPORT è impostato su CLIENT.

1506
Questo è il valore predefinito se TRANSPORT è impostato su DIRECT o DIRECTHTTP.

Qualsiasi numero intero positivo

PRIORITY

La priorità per i messaggi inviati a una destinazione.

Oggetti applicabili

Coda, Argomento

Nome esteso dello strumento di gestione JMS : PRIORITY

Nome breve dello strumento di amministrazione JMS : PRI

Accesso programmatico

Setter / getter

- MQDestination.setPriority()
- MQDestination.getPriority()

Valori

APP
La priorità è definita dall'applicazione JMS . Questo è il valore predefinito.

QDEF
La priorità assume il valore predefinito della coda.

Qualsiasi numero intero compreso nell'intervallo 0 - 9
Dal più basso al più alto.

PROCESSDURATION

Questa proprietà determina se un sottoscrittore garantisce di elaborare rapidamente qualsiasi messaggio che riceve prima di restituire il controllo a IBM MQ classes for JMS.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory

Nome esteso dello strumento di amministrazione JMS : PROCESSDURATION

Nome breve dello strumento di amministrazione JMS : PROCDUR

Accesso programmatico

Setter / getter

- MQConnectionFactory.setProcessDurata ()
- MQConnectionFactory.getProcessDurata ()

Valori

SCONOSCIUTO
Un utente non può fornire alcuna garanzia su quanto velocemente può elaborare qualsiasi messaggio che riceve. Questo è il valore predefinito.

SHORT
Un sottoscrittore garantisce di elaborare rapidamente qualsiasi messaggio che riceve prima di restituire il controllo a IBM MQ classes for JMS.

PROVIDERVERSION

Questa proprietà differenzia le tre modalità di funzionamento della messaggistica IBM MQ : IBM MQ modalità normale del provider di messaggistica, IBM MQ modalità normale del provider di messaggistica con limitazioni e modalità di migrazione del provider di messaggistica IBM MQ .

La modalità normale del provider di messaggistica IBM MQ utilizza tutte le funzioni di un gestore code IBM MQ per implementare JMS. Questa modalità è ottimizzata per utilizzare l'API e la funzionalità JMS 2.0 . La modalità normale del provider di messaggistica IBM MQ con limitazioni utilizza l'API JMS 2.0 , ma non le nuove funzioni quali le sottoscrizioni condivise, la consegna ritardata o l'invio asincrono.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnection Factory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : PROVIDERVERSION

Nome breve dello strumento di amministrazione JMS : PVER

Accesso programmatico

Setter / getter

- MQConnectionFactory.setProviderVersione ()
- MQConnectionFactory.getProviderVersione ()

Valori

È possibile impostare la proprietà **PROVIDERVERSION** su uno qualsiasi dei valori 8 (modalità normale), 7 (modalità normale con restrizioni), 6 (modalità di migrazione) o `non specificato` (il valore predefinito). Il valore che si specifica per la proprietà **PROVIDERVERSION** deve essere una stringa. Se si specifica una delle opzioni 8, 7 o 6, è possibile farlo in uno qualsiasi dei seguenti formati:

- V.R.M.F
- V.R.M
- V.R
- V

dove V, R, M e F sono numeri interi maggiori o uguali a zero. Gli ulteriori valori R, M ed F sono facoltativi e sono disponibili per l'utilizzo qualora sia necessario un controllo più preciso. Ad esempio, se si volesse utilizzare un livello **PROVIDERVERSION** di 7, è possibile impostare **PROVIDERVERSION**=7, `7.0`, `7.0.0` o `7.0.0.0`.

8 - Modalità normale
L'applicazione JMS utilizza la modalità normale del provider di messaggistica IBM MQ. La modalità normale utilizza tutte le funzioni di un gestore code IBM MQ per implementare JMS. Questa modalità è ottimizzata per utilizzare la API e la funzionalità JMS 2.0.

Se ci si connette a un gestore code con un livello di comando di 800 o successivo, è possibile utilizzare tutte le funzioni e l'API JMS 2.0 , ad esempio l'invio asincrono, la consegna ritardata o la sottoscrizione condivisa.

Se il gestore code specificato nelle impostazioni della factory di connessione non è un gestore code IBM MQ 8.0.0 o successivo, il metodo `createConnection` ha esito negativo con un'eccezione `JMSFMQ0003`.

La modalità normale del provider di messaggistica IBM MQ utilizza la funzione di condivisione delle conversazioni e il numero di conversazioni che può essere condiviso è controllato dalla proprietà **SHARECNV()** sul canale di connessione server. Se questa proprietà è impostata su `0`, non è possibile utilizzare la modalità normale del provider di messaggistica IBM MQ e il metodo `createConnection` non riesce con un'eccezione `JMSCC5007`.

7 - Modalità normale con restrizioni
L'applicazione JMS utilizza la modalità normale con restrizioni del provider di messaggistica IBM MQ. Questa modalità utilizza l'API JMS 2.0, ma non le nuove funzioni quali le sottoscrizioni condivise, il recapito ritardato o l'invio asincrono.

Se si imposta **PROVIDERVERSION** su 7, è disponibile solo la modalità operativa normale con restrizioni del provider di messaggistica IBM MQ.

Se ci si connette utilizzando la modalità normale con limitazioni, a un gestore code con un livello di comando inferiore a 800, è possibile utilizzare l'API JMS 2.0 , ma non le funzioni di invio asincrono, di consegna ritardata o di sottoscrizione condivisa.

La modalità normale con restrizioni del provider di messaggistica IBM MQ utilizza la funzione di condivisione delle conversazioni e il numero di conversazioni che può essere condiviso è controllato dalla proprietà **SHARECNV()** sul canale di connessione server. Se questa proprietà è impostata su `0`, non è possibile utilizzare la modalità normale con restrizioni del provider di messaggistica IBM MQ e il metodo `createConnection` non riesce con un'eccezione `JMSCC5007`.

6 - modalità di migrazione
L'applicazione JMS utilizza la modalità di migrazione del provider di messaggistica IBM MQ.

È possibile connettersi a un gestore code IBM MQ 8.0o successivo utilizzando questa modalità, ma nessuna delle nuove funzioni di un gestore code IBM MQ classes for JMS viene utilizzata, ad esempio, lettura anticipata o streaming.

Se si dispone di un client IBM MQ 8.0 o successive che si sta connettendo a un gestore code IBM MQ 8.0 o successive, la selezione dei messaggi viene eseguita dal gestore code piuttosto che dal sistema client.

Se viene specificata la modalità di migrazione del provider di messaggistica IBM MQ e si prova a utilizzare una qualsiasi delle API JMS 2.0, la chiamata del metodo API non riesce con l'eccezione `JMSCC5007`.

non `specificato` (valore predefinito)
La proprietà **PROVIDERVERSION** è impostata su *non specificato* per impostazione predefinita.

Una factory di connessione che era stata creata con una versione precedente di IBM MQ classes for JMS in JNDI prende questo valore quando la factory di connessione viene utilizzata con la nuova versione di IBM MQ classes for JMS. Il seguente algoritmo viene utilizzato per determinare la modalità di funzionamento utilizzata. Questo algoritmo viene utilizzato quando il metodo `createConnection` viene richiamato e utilizza altri aspetti della factory di connessione per determinare se è necessaria la modalità normale del provider di messaggistica IBM MQ, la modalità normale con restrizioni o la modalità di migrazione del provider di messaggistica IBM MQ.

1. In primo luogo, viene effettuato un tentativo di utilizzare la modalità normale del provider di messaggistica IBM MQ.
2. Se il gestore code connesso non è IBM MQ 8.0 o successive, viene effettuato un tentativo di utilizzare la modalità normale con restrizioni del provider di messaggistica IBM MQ.
3. Se il gestore code connesso non è IBM WebSphere MQ 7.0.1 o successive, la connessione viene chiusa e viene utilizzata invece la modalità di migrazione del provider di messaggistica IBM MQ.
4. Se la proprietà **SHARECNV** sul canale di connessione server è impostata su 0, la connessione viene chiusa e viene invece utilizzata la modalità di migrazione del provider di messaggistica IBM MQ.
5. Se **BROKERVER** è impostato su V1 o sul valore *non specificato* predefinito, la modalità normale del provider di messaggistica IBM MQ continua ad essere utilizzata.

   Vedere ALTER QMGR per i dettagli del parametro PSMODE del comando ALTER QMGR per ulteriori informazioni sulla compatibilità.
6. Se **BROKERVER** è impostata su V2 l'azione intrapresa dipende dal valore di **BROKERQMGR**:
   - Se il **BROKERQMGR** è `vuoto`:

     Se la coda specificata dalla proprietà **BROKERCONQ** può essere aperta per l'output (ossia, `MQOPEN` per l'output ha esito positivo) e **PSMODE** sul gestore code è impostata su `COMPAT` o `DISABLED`, viene utilizzata la modalità di migrazione del provider di messaggistica IBM MQ.
   - Se la coda specificata dalla proprietà **BROKERCONQ** non può essere aperta per l'output o l'attributo **PSMODE** è impostato su `ENABLED`:

     Viene utilizzata la modalità normale del provider di messaggistica IBM MQ.
   - Se **BROKERQMGR** è `non vuoto`:

     Viene utilizzata la modalità di migrazione del provider di messaggistica IBM MQ.

Se non è possibile modificare la factory di connessione che si sta utilizzando, è possibile utilizzare la proprietà `com.ibm.msg.client.wmq.overrideProviderVersion` per sovrascrivere qualsiasi impostazione sulla factory di connessione. Questa sovrascrittura si applica a tutte le factory di connessione della JVM, ma gli oggetti factory di connessione effettivi non vengono modificati.

Attività correlate
Configurazione della proprietà JMS **PROVIDERVERSION**

PROXYHOSTNAME

Il nome host o l'indirizzo IP del sistema su cui è in esecuzione il server proxy quando si utilizza una connessione in tempo reale ad un broker tramite un server proxy.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory

Nome esteso dello strumento di amministrazione JMS : PROXYHOSTNAME

Nome breve dello strumento di amministrazione JMS : PHOST

Accesso programmatico

Setter / getter

- MQConnectionFactory.setProxyHostName()
- MQConnectionFactory.getProxyHostName()

Valori

vuoto
Il nome host del server proxy. Questo è il valore predefinito.

PROXYPORT

Il numero della porta su cui il server proxy è in ascolto quando utilizza una connessione in tempo reale a un broker tramite un server proxy.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory

Nome esteso dello strumento di gestione JMS : PROXYPORT

Nome breve dello strumento di amministrazione JMS : PPORT

Accesso programmatico

Setter / getter

MQConnectionFactory.setProxyPorta ()

MQConnectionFactory.getProxyPorta ()

Valori

443
Il numero di porta del server proxy. Questo è il valore predefinito.

PUBACKINT

Il numero di messaggi pubblicati dal publisher prima che IBM MQ classes for JMS richieda un riconoscimento dal broker.

Quando si riduce il valore di questa proprietà, IBM MQ classes for JMS richiede i riconoscimenti più spesso, quindi le prestazioni del publisher diminuiscono. Quando si aumenta il valore, IBM MQ classes for JMS impiega più tempo per generare un'eccezione se il broker ha esito negativo. Questa proprietà è rilevante solo se TRANSPORT ha il valore BIND o CLIENT.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : PROXYPORT

Nome breve dello strumento di amministrazione JMS : PPORT

Accesso programmatico

Setter / getter

MQConnectionFactory.setPubAckInterval()

MQConnectionFactory.getPubAckInterval()

Valori

25
Qualsiasi numero intero positivo può essere il valore predefinito.

PUTASYNCALLOWED

Questa proprietà determina se ai mittenti del messaggio è consentito utilizzare i put asincroni per inviare messaggi a questa destinazione.

Oggetti applicabili

Coda, Argomento

Nome esteso dello strumento di gestione JMS : PUTASYNCALLOWED

Nome breve dello strumento di amministrazione JMS : PAALD

Accesso programmatico

Setter / getter

MQDestination.setPutAsyncAllowed()

MQDestination.getPutAsyncAllowed()

Valori

DEST ASS
Determinare se gli inserimenti asincroni sono consentiti facendo riferimento alla definizione dell'argomento o della coda. Questo è il valore predefinito.

DED_Q_ASS
Determinare se gli inserimenti asincroni sono consentiti facendo riferimento alla definizione della coda.

DEF S_TOPIC
Determinare se gli inserimenti asincroni sono consentiti facendo riferimento alla definizione dell'argomento.

No
Le immissioni asincrone non sono consentite.

SÌ
Le immissioni asincrone sono consentite.

QMANAGER

Il nome del gestore code a cui connettersi.

Tuttavia, se la propria applicazione utilizza una tabella di definizione del canale client per connettersi a un gestore code, consultare Utilizzo di una tabella di definizione del canale client con IBM MQ classes for JMS.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, XAConnectionFactory, XAQueueConnection Factory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : QMANAGER

Nome breve dello strumento di amministrazione JMS : QMGR

Accesso programmatico

Setter / getter

- MQConnectionFactory.setQueueManager ()
- MQConnectionFactory.getQueueManager ()

Valori

"" (stringa vuota)
Qualsiasi stringa può essere il valore predefinito.

CODA

Il nome della destinazione coda JMS . Corrisponde al nome della coda utilizzata dal gestore code.

Oggetti applicabili

Coda

Nome lungo strumento di gestione JMS : QUEUE

Nome breve dello strumento di amministrazione JMS : QU

Valori

Qualsiasi stringa
Qualsiasi nome coda IBM MQ valido.

Riferimenti correlati
Regole per la denominazione degli oggetti IBM MQ >

READAHEADALLOWED

Questa proprietà determina se gli utenti dei messaggi e i browser delle code possono utilizzare la lettura anticipata per ottenere messaggi non persistenti da questa destinazione in un buffer interno prima di riceverli.

Oggetti applicabili

Coda, Argomento

Nome esteso strumento di amministrazione JMS : READAHEADALLOWED

Nome breve strumento di amministrazione JMS : RAALD

Accesso programmatico

Setter / getter

- MQDestination.setReadAheadAllowed()
- MQDestination.getReadAheadAllowed()

Valori

DEST ASS
Determinare se la lettura anticipata è consentita facendo riferimento alla definizione dell'argomento o della coda. Questo è il valore predefinito negli strumenti di gestione.

Utilizzare WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_DEST nei programmi.

DED_Q_ASS
Determinare se la lettura anticipata è consentita facendo riferimento alla definizione della coda.

Utilizzare WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF nei programmi.

DEF S_TOPIC
Determinare se la lettura anticipata è consentita facendo riferimento alla definizione dell'argomento.

Utilizzare WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_TOPIC_DEF nei programmi.

No
La lettura anticipata non è consentita.

Utilizzare WMQConstants.WMQ_READ_AHEAD_ALLOWED_DISABLED nei programmi.

SÌ
La lettura anticipata è consentita.

Utilizzare WMQConstants.WMQ_READ_AHEAD_ALLOWED_ENABLED nei programmi.

READAHEADCLOSEPOLICY

Per i messaggi consegnati a un listener di messaggi asincrono, cosa accade ai messaggi nel buffer di lettura anticipata interno quando il consumer del messaggio è chiuso.

Oggetti applicabili

Coda, Argomento

Nome esteso dello strumento di gestione JMS : READAHEADCLOSEPOLICY

Nome breve dello strumento di amministrazione JMS : RACP

Accesso programmatico

Setter / getter

- MQDestination.setReadAheadClosePolicy()
- MQDestination.getReadAheadClosePolicy()

Valori

DELIVER_ALL
Tutti i messaggi nel buffer di lettura anticipata interno vengono consegnati al listener di messaggi dell'applicazione prima della restituzione. Questo è il valore predefinito negli strumenti di gestione.

Utilizzare WMQConstants.WMQ_READ_AHEAD_DELIVERALL nei programmi.

DISTRIBUZIONE_CORRENTE
Solo la chiamata del listener dei messaggi corrente viene completata prima della restituzione, lasciando potenzialmente i messaggi nel buffer di lettura anticipata interno, che vengono quindi eliminati.

Utilizzare WMQConstants.WMQ_READ_AHEAD_DELIVERCURRENT nei programmi.

RECEIVECCSID

Proprietà di destinazione che imposta il CCSID di destinazione per la conversione del messaggio del gestore code. Il valore viene ignorato a meno che RECEIVECONVERSION non sia impostata su `WMQ_RECEIVE_CONVERSION_QMGR`

Oggetti applicabili

Coda, Argomento

Nome lungo dello strumento di gestione JMS : RICEVIECCSID

Nome breve dello strumento di amministrazione JMS : RCCS

Accesso programmatico

Setter / Getter

- `MQDestination.setReceiveCCSID`
- `MQDestination.getReceiveCCSID`

Valori

WMQConstants.WMQ_RECEIVE_CCSID_JVM_DEFAULT
0 - Utilizza JVM `Charset.defaultCharset`

1208
UTF-8

CCSID
CCSID (coded character set identifier) supportato.

RECEIVECONVERSION

Proprietà di destinazione che determina se la conversione dei dati verrà eseguita dal gestore code.

Oggetti applicabili

Coda, Argomento

Nome esteso dello strumento di gestione JMS : RECEIVECONVERSION

Nome breve dello strumento di amministrazione JMS : RCNV

Accesso programmatico

Setter / Getter

- `MQDestination.setReceiveConversion`
- `MQDestination.getReceiveConversion`

Valori

`WMQConstants.WMQ_RECEIVE_CONVERSION_CLIENT_MSG`
1 - Eseguire la conversione dei dati solo sul client JMS . Il valore predefinito da V7.0e da, e incluso, 7.0.1.5.

`WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR`
2 - Esegue la conversione dei dati sul gestore code prima di inviare un messaggio al client. Il valore predefinito (e unico) da V7.0 a V7.0.1.4 incluso, tranne se viene applicato l'APAR IC72897 .

RECEIVEISOLATION

Questa proprietà determina se un sottoscrittore può ricevere messaggi di cui non è stato eseguito il commit sulla coda del sottoscrittore.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory

Nome esteso dello strumento di gestione JMS : RECEIVEISOLATION

Nome breve dello strumento di amministrazione JMS : RCVISOL

Valori

COMMIT ESEGUITO
Un sottoscrittore riceve solo i messaggi sulla coda del sottoscrittore di cui è stato eseguito il commit. Questo è il valore predefinito negli strumenti di gestione.

Utilizzare WMQConstants.WMQ_RCVISOL_COMMITTED nei programmi.

COMMIT ANNULLATO
Un sottoscrittore può ricevere messaggi di cui non è stato eseguito il commit sulla coda del sottoscrittore.

Utilizzare WMQConstants.WMQ_RCVISOL_UNCOMMITTED nei programmi.

RECEXIT

Identifica un'uscita di ricezione del canale, o una sequenza di uscite di ricezione, da eseguire in successione.

Potrebbe essere necessaria un'altra configurazione per consentire a IBM MQ classes for JMS di individuare le uscite di ricezione. Per ulteriori informazioni, consultare Configurazione di IBM MQ classes for JMS to use channel exits.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : RECEXIT

Nome breve dello strumento di amministrazione JMS : RCX

Accesso programmatico

Setter / getter

- MQConnectionFactory.setReceiveExit ()
- MQConnectionFactory.getReceiveExit ()

Valori

- `null`. Questo è il valore predefinito.
- Una stringa che comprende uno o più elementi separati da virgole, in cui ciascun elemento è:
  - Il nome di una classe che implementa l'interfaccia `WMQReceiveExit` (per un'uscita ricezione del canale scritta in Java).
  - Una stringa nel formato *libraryName(entryPointName)* (per un'exit di ricezione canale non scritta in Java).

RECEXITINIT

I dati utente passati al canale ricevono le uscite quando vengono richiamati.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello Strumento di gestione di JMS : RECEXITINIT

Nome breve dello strumento di amministrazione JMS : RCXI

Accesso programmatico

Setter / getter

- MQConnectionFactory.setReceiveExitInit()
- MQConnectionFactory.getReceiveExitInit()

Valori

vuoto
Una stringa che comprende uno o più elementi di dati utente separati da virgole. Questo è il valore predefinito.

REPLYTOSTYLE

Determina il modo in cui viene creato il campo JMSReplyTo in un messaggio ricevuto.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome completo dello strumento di gestione JMS : REPLYTOSTYLE

Nome breve strumento di amministrazione JMS : RTOST

Accesso programmatico

Setter / getter

- MQConnectionFactory.setReplyToStyle()
- MQConnectionFactory.getReplyToStyle()

Valori

VALORE PREDEFINITO
Equivalente a `MQMD`.

RFH2
Utilizzare il valore fornito nell'intestazione RFH2 . Se un valore JMSReplyTo è stato impostato nell'applicazione di invio, utilizzare tale valore.

MQMD
Utilizzare il valore MQMD fornito. Questo comportamento è equivalente al comportamento predefinito di IBM WebSphere MQ 6.0.2.4 e 6.0.2.5.

Se il valore JMSReplyTo impostato dall'applicazione di invio non contiene un nome gestore code, il gestore code di ricezione inserisce il proprio nome in MQMD. Se si imposta questo parametro su `MQMD`, la coda di risposta utilizzata si trova sul gestore code di ricezione. Se si imposta questo parametro su RFH2, la coda di risposta utilizzata si trova sul gestore code specificato in RFH2 del messaggio inviato come originariamente impostato dall'applicazione di invio.

Se il valore JMSReplyTo impostato dall'applicazione di invio contiene un nome gestore code, il valore di questo parametro non è importante perché sia MQMD che RFH2 contengono lo stesso valore.

RESCANINT

Quando un utente di messaggi nel dominio point - to - point utilizza un selettore di messaggi per selezionare quali messaggi desidera ricevere, IBM MQ classes for JMS ricercare nella coda IBM MQ i messaggi adatti nella sequenza determinata dall'attributo `MsgDeliverySequence` della coda.

Dopo IBM MQ classes for JMS trovare un messaggio adatto e consegnarlo all'utente, IBM MQ classes for JMS riprendere la ricerca del successivo messaggio adatto dalla posizione corrente nella coda. IBM MQ classes for JMS continua a cercare la coda in questo modo fino a quando non raggiunge la fine della coda o fino a quando l'intervallo di tempo in millisecondi, come determinato dal valore di questa proprietà, non è scaduto. In ogni caso IBM MQ classes for JMS ritorna all'inizio della coda per continuare la ricerca e inizia un nuovo intervallo di tempo.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nome esteso dello Strumento di gestione JMS : RESCANINT

Nome breve dello strumento di amministrazione JMS : RINT

Accesso programmatico

Setter / getter

- MQConnectionFactory.setRescanIntervallo ()
- MQConnectionFactory.getRescanIntervallo ()

Valori

5000
Qualsiasi numero intero positivo può essere il valore predefinito.

SECEXIT

Identifica un'uscita di sicurezza del canale.

Potrebbe essere necessaria una configurazione aggiuntiva per consentire a IBM MQ classes for JMS di individuare le uscite di sicurezza. Per ulteriori informazioni, consultare Configurazione di IBM MQ classes for JMS to use channel exits.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : SECEXIT

Nome breve dello strumento di amministrazione JMS : SXC

Accesso programmatico

Setter / getter

- MQConnectionFactory.setSecurityExit ()
- MQConnectionFactory.getSecurityExit ()

Valori

- `null`. Questo è il valore predefinito.
- Una stringa che comprende uno o più elementi separati da virgole, in cui ciascun elemento è:
  - Il nome di una classe che implementa l'interfaccia `WMQSecurityExit` (per un'uscita di sicurezza del canale scritta in Java).
  - Una stringa nel formato *libraryName(entryPointName)* (per un'uscita di sicurezza canale non scritta in Java).

SECEXITINIT

I dati utente passati a un'uscita di sicurezza del canale quando viene richiamata.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : SECEXITINIT

Nome breve dello strumento di amministrazione JMS : SCXI

Accesso programmatico

Setter / getter

- MQConnectionFactory.setSecurityExitInit()
- MQConnectionFactory.getSecurityExitInit()

Valori

vuoto
Qualsiasi stringa può essere il valore predefinito.

SENDCHECKCOUNT

Il numero di chiamate di invio da consentire tra il controllo degli errori di inserimento asincrono, all'interno di una singola sessione JMS non transatta.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome completo dello strumento di amministrazione JMS : SENDCHECKCOUNT

Nome breve dello strumento di amministrazione JMS : SCC

Accesso programmatico

Setter / getter

- MQConnectionFactory.setSendCheckCount()
- MQConnectionFactory.getSendCheckCount()

Valori

vuoto
Qualsiasi stringa può essere il valore predefinito.

SENDEXIT

Identifica un'uscita di invio del canale o una sequenza di uscite di invio da eseguire in successione.

Potrebbe essere necessaria ulteriore configurazione per consentire a IBM MQ classes for JMS di individuare le uscite di invio. Per ulteriori informazioni, consultare Configurazione di IBM MQ classes for JMS to use channel exits.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : SENDEXIT

Nome breve dello strumento di amministrazione JMS : SDX

Accesso programmatico

Setter / getter

- MQConnectionFactory.setSendExit ()
- MQConnectionFactory.getSendExit ()

Valori

- `null`. Questo è il valore predefinito.
- Una stringa che comprende uno o più elementi separati da virgole, in cui ciascun elemento è:
  - Il nome di una classe che implementa l'interfaccia di `WMQSendExit` (per un'uscita di trasmissione del canale scritta in Java).
  - Una stringa nel formato *libraryName(entryPointName)* (per un'uscita di invio del canale non scritta in Java).

SENDEXITINIT

I dati utente passati alle uscite di invio del canale quando vengono richiamate.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome lungo dello strumento di gestione JMS : SENDEXITINIT

Nome breve dello strumento di amministrazione JMS : SDXI

Accesso programmatico

Setter / getter

- MQConnectionFactory.setSendExitInit()
- MQConnectionFactory.getSendExitInit()

Valori

vuoto
Qualsiasi stringa che comprende uno o più elementi di dati utente separati da virgole può essere il valore predefinito.

SHARECONVALLOWED

Per le applicazioni che utilizzano la modalità normale del provider di messaggistica IBM MQ o la modalità normale con limitazioni, questa proprietà determina se la funzione di condivisione delle conversazioni viene utilizzata per JMS connessioni, sessioni e contesti creati dalla factory di connessione.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome lungo strumento di amministrazione JMS : SHARECONVALLOWED

Nome breve dello strumento di amministrazione JMS : SCALD

Accesso programmatico

Setter / getter

- MQConnectionFactory.setShareConvAllowed()
- MQConnectionFactory.getShareConvAllowed()

Valori

SÌ

Le connessioni JMS , le sessioni e i contesti creati dal factory di connessione all'interno della stessa JVM possono condividere un'istanza del canale (che si associa a una connessione TCP/IP) dove appropriato.

Questo è il valore predefinito per gli strumenti di gestione.

Per i programmi, utilizzare WMQConstants.WMQ_SHARE_CONV_ALLOWED_YES.

No

Ogni connessione JMS creata dalla factory di connessione e ogni sessione JMS creata da quelle connessioni JMS, ha la propria istanza del canale (connessione TCP/IP) su un gestore code.

Per contesti JMS , il primo contesto creato dalla factory di connessione crea due istanze del canale (connessioni TCP/IP). Altri contesti JMS creati dal primo hanno la propria istanza del canale (connessione TCP/IP).

Per i programmi, utilizzare WMQConstants.WMQ_SHARE_CONV_ALLOWED_NO.

Concetti correlati

Modalità operative del provider di messaggistica IBM MQ

Condivisione di un collegamento TCP/IP nelle classi IBM MQ per JMS

SPARSESUBS

Controlla la politica di richiamo messaggi di un oggetto TopicSubscriber .

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory

Nome esteso dello Strumento di gestione JMS : SPARSESUBS

Nome breve dello strumento di amministrazione JMS : SSUBS

Accesso programmatico

Setter / getter

- MQConnectionFactory.setSparseSottoscrizioni ()
- MQConnectionFactory.getSparseSottoscrizioni ()

Valori

No

Le sottoscrizioni ricevono messaggi di corrispondenza frequenti. Questo è il valore predefinito per gli strumenti di gestione.

Per i programmi, utilizzare false.

SÌ

Le sottoscrizioni ricevono messaggi di corrispondenza non frequenti. Questo valore richiede che la coda di sottoscrizione possa essere aperta per la ricerca.

Per i programmi, utilizzare true.

SSLCIPHERSUITE

La CipherSuite da utilizzare per una connessione TLS.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello Strumento di gestione JMS : SSLCIPHERSUITE

Nome breve dello strumento di amministrazione JMS : SCPHS

Accesso programmatico

Setter / getter

- MQConnectionFactory.setSSLCipherSuite ()
- MQConnectionFactory.getSSLCipherSuite ()

Valori

vuoto
Questo è il valore predefinito. Per ulteriori informazioni, vedi Proprietà TLS degli oggetti JMS.

SSLCRL

Server CRL per controllare la revoca del certificato TLS.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso strumento di amministrazione JMS : SSLCRL

Nome breve dello strumento di amministrazione JMS : SCRL

Accesso programmatico

Setter / getter

- MQConnectionFactory.setSSLCertStores ()
- MQConnectionFactory.getSSLCertStores ()

Valori

vuoto
Elenco separato da spazi di URL LDAP. Questo è il valore predefinito. Per ulteriori informazioni, vedi Proprietà TLS degli oggetti JMS.

SSLFIPSREQUIRED

Questa proprietà determina se una connessione TLS deve utilizzare una CipherSuite supportata dal provider IBM Java JSSE FIPS (IBMJSSEFIPS).

Nota: Su AIX, Linux, and Windows, IBM MQ fornisce la conformità FIPS 140-2 tramite il modulo crittografico IBM Crypto for C (ICC) . Il certificato per questo modulo è stato spostato nello stato cronologico. I clienti devono visualizzare il certificatoIBM Crypto for C (ICC) ed essere a conoscenza di eventuali consigli forniti da NIST. Un modulo FIPS 140-3 di sostituzione è attualmente in corso e il relativo stato può essere visualizzato ricercandolo in NIST CMVP modules in process list.

IBM MQ Operator 3.2.0 e l'immagine del contenitore del gestore code 9.4.0.0 sono basati su UBI 9. La conformità FIPS 140-3 è attualmente in sospeso e il suo stato può essere visualizzato ricercando "Red Hat Enterprise Linux 9 - OpenSSL FIPS Provider" in NIST CMVP modules in process list.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello Strumento di gestione JMS : SSLFIPSREQUIRED

Nome breve dello strumento di amministrazione JMS : SFIPS

Accesso programmatico

Setter / getter

- MQConnectionFactory.setSSLFipsRequired ()
- MQConnectionFactory.getSSLFipsRequired ()

Valori

No
Una connessione TLS può utilizzare qualsiasi CipherSuite non supportato dal provider IBM Java JSSE FIPS (IBMJSSEFIPS).

Questo è il valore predefinito. Nei programmi, utilizzare false.

SÌ
Una connessione TLS deve utilizzare una CipherSuite supportata da IBMJSSEFIPS.

Nei programmi, utilizzare true.

SSLPEERNAME

Per TLS, una struttura *DN (distinguished name)* che deve corrispondere a quella fornita dal gestore code.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello Strumento di gestione JMS : SSLPEERNAME

Nome breve dello strumento di amministrazione JMS : SPEER

Accesso programmatico

Setter / getter

- MQConnectionFactory.setSSLPeerNome ()
- MQConnectionFactory.getSSLPeerNome ()

Valori

vuoto
Questo è il valore predefinito. Per ulteriori informazioni, vedi Proprietà TLS degli oggetti JMS.

SSLRESETCOUNT

Per TLS, il numero totale di byte inviati e ricevuti da una connessione prima che venga rinegoziata la chiave segreta utilizzata per la codifica.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : SSLRESETCOUNT

Nome breve dello strumento di amministrazione JMS : SRC

Accesso programmatico

Setter / getter

- MQConnectionFactory.setSSLResetConteggio ()
- MQConnectionFactory.getSSLResetConteggio ()

Valori

0
Zero o qualsiasi numero intero positivo minore o uguale a 999, 999, 999. Questo è il valore predefinito. Per ulteriori informazioni, vedi Proprietà TLS degli oggetti JMS.

STATREFRESHINT

L'intervallo, in millesimi di secondo, tra gli aggiornamenti della transazione di lunga durata che rileva quando un sottoscrittore perde la propria connessione al gestore code.

Questa proprietà è rilevante solo se SUBSTORE ha il valore QUEUE.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : STATREFRESHINT

Nome breve dello strumento di amministrazione JMS : SRI

Accesso programmatico

Setter / getter

- MQConnectionFactory.setStatusRefreshInterval()
- MQConnectionFactory.getStatusRefreshInterval()

Valori

60000
Qualsiasi numero intero positivo può essere il valore predefinito. Per ulteriori informazioni, vedi Proprietà TLS degli oggetti JMS.

SUBSTORE

Dove IBM MQ classes for JMS memorizza i dati persistenti relativi alle sottoscrizioni attive.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di amministrazione JMS : SUBSTORE

Nome breve dello strumento di amministrazione JMS : SS

Accesso programmatico

Setter / getter

- MQConnectionFactory.setSubscriptionStore ()

- MQConnectionFactory.getSubscriptionStore ()

Valori

BROKER
Utilizzare l'archivio sottoscrizioni basato su broker per conservare i dettagli delle sottoscrizioni. Questo è il valore predefinito per gli strumenti di gestione.

Per programmi, utilizzare WMQConstants.WMQ_SUBSTORE_BROKER.

MIGRAZIONE
Trasferire le informazioni di sottoscrizione dall'archivio di sottoscrizione basato sulla coda all'archivio di sottoscrizione basato sul broker.

Per programmi, utilizzare WMQConstants.WMQ_SUBSTORE_MIGRATE.

CODA
Utilizzare l'archivio sottoscrizioni basato sulla coda per conservare i dettagli delle sottoscrizioni.

Per programmi, utilizzare WMQConstants.WMQ_SUBSTORE_QUEUE.

SYNCPOINTALLGETS

Questa proprietà determina se tutte le operazioni di richiamo devono essere eseguite nel punto di sincronizzazione.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : SYNCPOINTALLGETS

Nome breve dello strumento di amministrazione JMS : SPAG

Accesso programmatico

Setter / getter

- MQConnectionFactory.setSyncpointAllGets()
- MQConnectionFactory.getSyncpointAllGets()

Valori

No
Questo è il valore predefinito.

Sì

TARGCLIENT

Questa proprietà determina se il formato IBM MQ RFH2 viene utilizzato per scambiare informazioni con le applicazioni di destinazione.

Oggetti applicabili

Coda, Argomento

Nome esteso dello strumento di amministrazione JMS : TARGCLIENT

Nome breve dello strumento di amministrazione JMS : TC

Accesso programmatico

Setter / getter

- MQDestination.setTargetClient()
- MQDestination.getTargetClient()

Valori

JMS
La destinazione del messaggio è un'applicazione JMS . Questo è il valore predefinito per gli strumenti di gestione.

Per i programmi, utilizzare WMQConstants.WMQ_CLIENT_JMS_COMPLIANT.

MQ
La destinazione del messaggio è un'applicazione nonJMS IBM MQ .

Per i programmi, utilizzare WMQConstants.WMQ_CLIENT_NONJMS_MQ.

TARGCLIENTMATCHING

Questa proprietà determina se un messaggio di risposta, inviato alla coda identificata dal campo di intestazione JMSReplyTo di un messaggio in ingresso, ha un'intestazione MQRFH2 solo se il messaggio in entrata ha un'intestazione MQRFH2 .

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nome esteso dello strumento di gestione JMS : TARGCLIENTMATCHING

Nome breve dello strumento di amministrazione JMS : TCM

Accesso programmatico

Setter / getter

- MQConnectionFactory.setTargetClientMatching()
- MQConnectionFactory.getTargetClientMatching()

Valori

SÌ
Se un messaggio in entrata non ha un'intestazione MQRFH2 , la proprietà TARGCLIENT dell'oggetto Queue derivata dal campo di intestazione JMSReplyTo del messaggio viene inviata a MQ. Se il messaggio non ha un'intestazione MQRFH2 , la proprietà TARGCLIENT viene invece impostata su JMS . Questo è il valore predefinito per gli strumenti di gestione.

Per i programmi, utilizzare true.

No
La proprietà TARGCLIENT dell'oggetto Queue derivato dal campo di intestazione JMSReplyTo di un messaggio in entrata è sempre impostata su JMS.

Per i programmi, utilizzare false.

TEMPMODEL

Il nome della coda modello da cui vengono create le code temporanee JMS .

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nome esteso dello strumento di amministrazione JMS : TEMPMODEL

Nome breve dello strumento di amministrazione JMS : TM

Accesso programmatico

Setter / getter

- MQConnectionFactory.setTemporaryModello ()
- MQConnectionFactory.getTemporaryModello ()

Valori

SYSTEM.DEFAULT.MODEL.QUEUE
Qualsiasi stringa può essere il valore predefinito.

TEMPQPREFIX

Il prefisso utilizzato per formare il nome di una coda dinamica IBM MQ .

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nome esteso dello strumento di amministrazione JMS : TEMPQPREFIX

Nome breve dello strumento di amministrazione JMS : TQP

Accesso programmatico

Setter / getter

- MQConnectionFactory.setTempQPrefix ()
- MQConnectionFactory.getTempQPrefix ()

Valori

'' '' (stringa vuota)
Il prefisso utilizzato è `CSQ.*` su z/OS e `AMQ.*` su tutte le altre piattaforme. Questi sono i valori predefiniti.

Prefisso coda
Il prefisso della coda è qualsiasi stringa conforme alle regole per formare il contenuto del campo *DynamicQName* in un descrittore dell'oggetto IBM MQ (struttura MQOD), ma l'ultimo carattere non vuoto deve essere un asterisco.

TEMPTOPICPREFIX

Quando si creano argomenti temporanei, JMS genera una stringa di argomenti nel formato '' TEMP */ TEMPTOPICPREFIX/unique_id* '' oppure se questa proprietà viene lasciata con il valore predefinito, solo '' TEMP */unique_id* ''. La specifica di un TEMPTOPICPREFIX non vuoto consente la definizione di code modello specifiche per la creazione delle code gestite per i sottoscrittori di argomenti temporanei creati in questa connessione.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : TEMPTOPICPREFIX

Nome breve dello strumento di amministrazione JMS : TTP

Accesso programmatico

Setter / getter

- MQConnectionFactory.setTempTopicPrefix()
- MQConnectionFactory.getTempTopicPrefix()

Valori

Qualsiasi stringa non null costituita solo da caratteri validi per una stringa di argomenti IBM MQ . Il valore predefinito è '' '' (stringa vuota).

TOPIC

Il nome della destinazione dell'argomento JMS , questo valore viene utilizzato dal gestore code come stringa di argomenti di una pubblicazione o sottoscrizione.

Oggetti applicabili

Argomento

Nome esteso dello strumento di gestione JMS : TOPIC

Nome breve dello strumento di amministrazione JMS : TOP

Valori

Qualsiasi stringa
Una stringa che forma una stringa argomento IBM MQ valida. Quando si utilizza IBM MQ come provider di messaggistica con WebSphere Application Server, specificare un valore che corrisponda al nome con cui l'argomento è noto per scopi di amministrazione in WebSphere Application Server.

Riferimenti correlati
Stringhe argomento

TRANSPORT

La natura di una connessione a un gestore code o broker.

Oggetti applicabili

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : TRANSPORT

Nome breve dello strumento di amministrazione JMS : TRAN

Accesso programmatico

Setter / getter

- MQConnectionFactory.setTransportTipo ()
- MQConnectionFactory.getTransportTipo ()

Valori

BIND
Per una connessione a un gestore code in modalità bind. Questo è il valore predefinito per gli strumenti di gestione.

Per i programmi, utilizzare WMQConstants.WMQ_CM_BINDINGS.

CLIENTE
Per una connessione ad un gestore code in modalità client.

Per i programmi, utilizzare WMQConstants.WMQ_CM_CLIENT.

DIRECT
Per una connessione in tempo reale a un broker che non utilizza il tunnelling HTTP.

Per i programmi, utilizzare WMQConstants.WMQ_CM_DIRECT_TCPIP.

DIREZIONE
Per una connessione in tempo reale a un broker utilizzando il tunnelling HTTP. È supportato solo HTTP 1.0 .

Per i programmi, utilizzare WMQConstants.WMQ_CM_DIRECT_HTTP.

Concetti correlati
“Dipendenze tra proprietà di oggetti IBM MQ classes for JMS” a pagina 1948
La validità di alcune proprietà dipende dai valori particolari di altre proprietà.

WILDCARDFORMAT

Questa proprietà determina la versione della sintassi dei caratteri jolly da utilizzare.

Oggetti applicabili

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nome esteso dello strumento di gestione JMS : WILDCARDFORMAT

Nome breve dello strumento di amministrazione JMS : WCFMT

Accesso programmatico

Setter / getter

- MQConnectionFactory.setWildCardFormat()
- MQConnectionFactory.getWildCardFormat()

Valori

SOLO argomento
Riconosce solo i caratteri jolly a livello di argomento, come utilizzato nel broker versione 2. Questo è il valore predefinito per gli strumenti di gestione.

Per programmi, utilizzare WMQConstants.WMQ_WILDCARD_TOPIC_ONLY.

SOLO GRAFICO
Riconosce solo i caratteri jolly, come utilizzato nel broker versione 1.

Per programmi, utilizzare WMQConstants.WMQ_WILDCARD_CHAR_ONLY.

La proprietà ENCODING

La proprietà ENCODING comprende tre proprietà secondarie, in dodici possibili combinazioni.

I valori validi che la proprietà `ENCODING` può assumere sono creati dalle tre proprietà secondarie:

codifica a numero intero
Normale o invertito

codifica a numero decimale
Normale o invertito

codifica a virgola mobile
IEEE normale, IEEE invertito o z/OS

La proprietà `ENCODING` è espressa come una stringa di tre caratteri con la sintassi seguente:

```
{N|R}{N|R}{N|R|3}
```

In questa stringa:

- N indica normale
- R indica l'inversione
- 3 indica z/OS
- Il primo carattere rappresenta la *codifica numero intero*
- Il secondo carattere rappresenta la *codifica decimale*
- Il terzo carattere rappresenta la *codifica a virgola mobile*

Fornisce una serie di dodici valori possibili per la proprietà ENCODING .

Esiste un ulteriore valore, la stringa NATIVE, che imposta i valori di codifica appropriati per la piattaforma Java .

I seguenti esempi mostrano combinazioni valide per ENCODING:

```
ENCODING(NNR)
ENCODING(NATIVE)
ENCODING(RR3)
```

Proprietà TLS degli oggetti JMS

Abilitare la crittografia TLS (Transport Layer Security) utilizzando la proprietà SSLCIPHERSUITE. È quindi possibile modificare le caratteristiche della codifica TLS utilizzando diverse altre proprietà.

Quando si specifica TRANSPORT (CLIENT), è possibile abilitare la comunicazione codificata TLS utilizzando la proprietà SSLCIPHERSUITE. Impostare questa proprietà su una CipherSuite valida fornita dal provider JSSE; deve corrispondere alla CipherSpec indicata sul canale SVRCONN indicato dalla proprietà CHANNEL.

Tuttavia, CipherSpecs (come specificato sul canale SVRCONN) e CipherSuites (come specificato sugli oggetti ConnectionFactory) utilizzano schemi di denominazione differenti per rappresentare gli stessi algoritmi di codifica TLS. Se viene specificato un nome CipherSpec riconosciuto nella proprietà SSLCIPHERSUITE, JMSAdmin emette un'avvertenza e associa CipherSpec alla CipherSuiteequivalente. Consultare TLS CipherSpecs e CipherSuites in IBM MQ classes for JMS per un elenco di CipherSpecs riconosciuti da IBM MQ e JMSAdmin.

Se si richiede una connessione per utilizzare una CipherSuite supportata dal provider FIPS JSSE IBM Java (IBMJSSEFIPS), impostare la proprietà SSLFIPSREQUIRED del factory di connessione su YES. Il valore predefinito di questa proprietà è NO, che significa che una connessione può utilizzare qualsiasi CipherSuitesupportata. La proprietà viene ignorata se SSLCIPHERSUITE non è impostata.

SSLPEERNAME corrisponde al formato del parametro SSLPEER, che può essere impostato sulle definizioni di canale. Si tratta di un elenco di coppie nome - valore dell'attributo separate da virgole o punti e virgola. Ad esempio:

```
SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

La serie di nomi e valori costituisce un *nome distinto*. Per ulteriori dettagli sui DN (Distinguished Name) e sul loro utilizzo con IBM MQ, consultare Protezione di IBM MQ.

L'esempio fornito controlla il certificato identificativo presentato dal server in fase di connessione. Perché la connessione abbia esito positivo, il certificato deve avere un nome comune che inizia con QMGR., e deve avere almeno due nomi di unità organizzative, il cui primo è IBM e il secondo WEBSPHERE. La verifica non è sensibile al maiuscolo / minuscolo.

Se SSLPEERNAME non è impostato, non viene eseguito alcun controllo di questo tipo. SSLPEERNAME viene ignorato se SSLCIPHERSUITE non è impostato.

La proprietà SSLCRL specifica zero o più server CRL (Certificate Revocation List). L'uso di questa propriet ... richiede una JVM all'indirizzo Java 2 v1.4. Questo è un elenco delimitato da spazi di voci del formato:

```
ldap:// hostname:[ port ]
```

facoltativamente seguito da un singolo /. Se *port* viene omesso, viene utilizzata la porta LDAP predefinita 389. In fase di connessione, il certificato TLS presentato dal server viene controllato rispetto ai server CRL specificati. Consultare Protezione di IBM MQ per ulteriori informazioni sulla sicurezza CRL.

Se SSLCRL non è impostato, non viene eseguito alcun controllo di questo tipo. SSLCRL viene ignorato se SSLCIPHERSUITE non è impostato.

La proprietà SSLRESETCOUNT rappresenta il numero totale di byte inviati e ricevuti da un collegamento prima che la chiave segreta utilizzata per la codifica venga rinegoziata. Il numero di byte inviati è il numero prima della codifica e il numero di byte ricevuti è il numero dopo la decodifica. Il numero di byte include anche le informazioni di controllo inviate e ricevute da IBM MQ classes for JMS.

Ad esempio, per configurare un oggetto ConnectionFactory che può essere utilizzato per creare una connessione su un canale MQI abilitato a TLS con una chiave segreta che viene rinegoziata dopo il flusso di 4 MB di dati, immettere il seguente comando a JMSAdmin:

```
ALTER CF(my.cf) SSLRESETCOUNT(4194304)
```

Se il valore di SSLRESETCOUNT è zero, che è il valore predefinito, la chiave segreta non viene mai rinegoziata. La proprietà SSLRESETCOUNT viene ignorata se SSLCIPHERSUITE non è impostata.

Riferimento di IBM MQ Message Service Client (XMS) for .NET

Questa sezione di riferimento fornisce informazioni sulle interfacce della classe IBM MQ Message Service Client (XMS) for .NET (XMS .NET) e sulle proprietà dell'oggetto definite da XMS.

.NETInterfacce

Questa sezione descrive le interfacce della classe .NET e le relative proprietà e metodi.

La seguente tabella riepiloga le interfacce, che sono definite all'interno dello spazio dei nomi `IBM.XMS` .

Tabella 871. Riepilogo delle interfacce della classe .NET

Interfaccia	Descrizione
“IBytesMessage” a pagina 2005	Un messaggio di byte è un messaggio il cui contenuto comprende un flusso di byte.
“IConnessione” a pagina 2014	Un oggetto Connection rappresenta la connessione attiva dell'applicazione ad un server di messaggi.
“IConnectionFactory” a pagina 2017	Un'applicazione utilizza una produzione connessioni per creare una connessione.
“Dati IConnectionMeta” a pagina 2018	Un oggetto ConnectionMetaData fornisce informazioni su una connessione.
“IDestinazione” a pagina 2019	Una destinazione è il punto in cui un'applicazione invia messaggi o è un'origine da cui un'applicazione riceve messaggi o entrambi.
“ExceptionListener” a pagina 2020	Un'applicazione utilizza un listener di eccezioni per ricevere una notifica asincrona di un problema con una connessione.

Tabella 871. Riepilogo delle interfacce della classe .NET (Continua)	
Interfaccia	**Descrizione**
“Eccezione IllegalState” a pagina 2021	XMS genera questa eccezione se un'applicazione richiama un metodo in un momento non corretto o inappropriato o se XMS non si trova in uno stato appropriato per l'operazione richiesta.
“InitialContext” a pagina 2021	Un'applicazione utilizza un oggetto InitialContext per creare oggetti dalle definizioni di oggetto richiamati da un repository di oggetti gestiti.
“IDException InvalidClient” a pagina 2023	XMS genera questa eccezione se un'applicazione tenta di impostare un identificatore client per una connessione, ma l'identificativo client non è valido o è già in uso.
“Eccezione InvalidDestination” a pagina 2023	XMS genera questa eccezione se un'applicazione specifica una destinazione non valida.
“Eccezione InvalidSelector” a pagina 2023	XMS genera questa eccezione se un'applicazione fornisce un'espressione del selettore di messaggi la cui sintassi non è valida.
“IMapMessage” a pagina 2024	Un messaggio della mappa è un messaggio il cui corpo comprende una serie di coppie nome - valore, in cui ogni valore ha un tipo di dati associato.
“IMessaggio” a pagina 2032	Un oggetto Message rappresenta un messaggio che un'applicazione invia o riceve. IMessage è una superclasse per le classi di messaggi come IMapMessage.
“IMessageConsumer” a pagina 2038	Un'applicazione utilizza un consumatore di messaggi per ricevere i messaggi inviati a una destinazione.
“MessageEOFException” a pagina 2041	XMS genera questa eccezione se XMS rileva la fine di un flusso di messaggi di byte quando un'applicazione sta leggendo il contenuto di un messaggio di byte.
“Eccezione MessageFormat” a pagina 2041	XMS genera questa eccezione se XMS rileva un messaggio con un formato non valido.
“IMessageListener (delegato)” a pagina 2041	Un'applicazione utilizza un listener di messaggi per ricevere i messaggi asincrono.
“MessageNotReadableException” a pagina 2042	XMS genera questa eccezione se un'applicazione tenta di leggere il corpo di un messaggio di sola scrittura.
“MessageNotWritableException” a pagina 2042	XMS genera questa eccezione se un'applicazione tenta di scrivere nel corpo di un messaggio di sola lettura.
“IMessageProducer” a pagina 2042	Un'applicazione utilizza un produttore di messaggi per inviare messaggi a una destinazione.
“IObjectMessage” a pagina 2048	Un messaggio oggetto è un messaggio il cui corpo comprende un oggetto Java o .NET serializzato.

Tabella 871. Riepilogo delle interfacce della classe .NET (Continua)	
Interfaccia	**Descrizione**
“IPropertyContext” a pagina 2049	IPropertyContext è una superclasse astratta che contiene metodi che richiamano e impostano le proprietà. Questi metodi vengono ereditati dalle altre classi.
“IQueueBrowser” a pagina 2057	Un'applicazione utilizza un browser della coda per ricercare i messaggi su una coda senza rimuoverli.
“Richiedente” a pagina 2059	Un'applicazione utilizza un richiedente per inviare un messaggio di richiesta e attendere e ricevere la risposta.
“Eccezione ResourceAllocation” a pagina 2060	XMS genera questa eccezione se XMS non può allocare le risorse richieste da un metodo.
“SecurityException” a pagina 2061	XMS genera questa eccezione se l'identificativo utente e la parola d'ordine forniti per autenticare un'applicazione vengono rifiutati. XMS genera questa eccezione anche se un controllo di autorizzazione non riesce e impedisce il completamento di un metodo.
“ISessione” a pagina 2061	Una sessione è un contesto a thread singolo per l'invio e la ricezione di messaggi.
“IStreamMessage” a pagina 2072	Un messaggio di flusso è un messaggio il cui contenuto comprende un flusso di valori, dove ogni valore ha un tipo di dati associato.
“ITextMessage” a pagina 2080	Un messaggio di testo è un messaggio il cui contenuto comprende una stringa.
“TransactionInProgressException” a pagina 2081	XMS genera questa eccezione se un'applicazione richiede un'operazione non valida perché è in corso una transazione.
“TransactionRolledBackException” a pagina 2082	XMS genera questa eccezione se un'applicazione richiama Session.commit() per eseguire il commit della transazione corrente, ma viene eseguito il rollback della transazione.
XMSC	Per .NET, i valori e i nomi proprietà XMS sono definiti in questa classe come costanti pubbliche. Per ulteriori dettagli, fare riferimento a “Proprietà degli oggetti XMS” a pagina 2084.
“Eccezione XMSException” a pagina 2082	Se XMS rileva un errore durante l'elaborazione di una chiamata a un metodo .NET , XMS genera un'eccezione. Un'eccezione è un oggetto che contiene informazioni sull'errore. Esistono diversi tipi di eccezione XMS e un oggetto XMSException è solo un tipo di eccezione. Tuttavia, la classe XMSException è una superclasse delle altre classi di eccezione XMS . XMS genera un oggetto XMSException in situazioni in cui nessuno degli altri tipi di eccezione è appropriato.

Tabella 871. Riepilogo delle interfacce della classe .NET (Continua)	
Interfaccia	**Descrizione**
“XMSFactoryFactory” a pagina 2083	Se un'applicazione non utilizza oggetti gestiti, utilizzare questa classe per creare factory di connessione, code e argomenti.

La definizione di ciascun metodo elenca i codici di errore che XMS potrebbe restituire se rileva un errore durante l'elaborazione di una chiamata al metodo. Ogni codice di eccezione è rappresentato dalla sua costante denominata, che ha un'eccezione corrispondente.

IBytesMessage

Un messaggio di byte è un messaggio il cui contenuto comprende un flusso di byte.

Gerarchia di eredità:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IBytesMessage
```

Proprietà di .NET

BodyLength - Richiama la lunghezza del body

Interfaccia:

```
Int64 BodyLength
    {
      get;
    }
```

Richiamare la lunghezza del corpo del messaggio in byte quando il corpo del messaggio è di sola lettura

Il valore restituito è la lunghezza dell'intero corpo indipendentemente da dove è attualmente posizionato il cursore per la lettura del messaggio.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException

Metodi

ReadBoolean - Leggi valore booleano

Interfaccia:

```
Boolean ReadBoolean();
```

Leggere un valore booleano dal flusso di messaggi byte.

Parametri:
Nessuna

Restituisce:
Il valore booleano letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

Byte ReadSigned- Byte di lettura

Interfaccia:

```
Int16   ReadSignedByte();
```

Leggere il byte successivo dal flusso di messaggi di byte come un numero intero a 8 bit con segno.

Parametri:
Nessuna

Restituisce:
Il byte letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadBytes - Byte letti

Interfaccia:

```
Int32   ReadBytes(Byte[] array);
Int32   ReadBytes(Byte[] array, Int32 length);
```

Leggere un array di byte dal flusso di messaggi di byte a partire dalla posizione corrente del cursore.

Parametri:

array (output)
Il buffer che contiene l'array di byte letti. Se il numero di byte rimanenti da leggere dal flusso prima della chiamata è maggiore o uguale alla lunghezza del buffer, il buffer viene riempito. Altrimenti, il buffer viene riempito parzialmente con tutti i restanti byte.

Se si specifica un puntatore null sull'input, il metodo ignora i byte senza leggerli. Se il numero di byte rimanenti da leggere dal flusso prima che la chiamata sia maggiore o uguale alla lunghezza del buffer, il numero di byte ignorati è uguale alla lunghezza del buffer. Altrimenti, tutti i byte rimanenti vengono ignorati. Il cursore rimane nella successiva posizione da leggere nel flusso di messaggi di byte.

lunghezza (input)
La lunghezza del buffer in byte

Restituisce:
Il numero di byte letti nel buffer. Se il buffer è parzialmente riempito, il valore è inferiore alla lunghezza del buffer, indicando che non ci sono altri byte rimanenti da leggere. Se non restano byte da leggere dal flusso prima della chiamata, il valore è XMSC_END_OF_STREAM.

Se si specifica un puntatore nullo sull'immissione, il metodo non restituisce alcun valore.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException

ReadChar - Carattere di lettura

Interfaccia:

```
Char    ReadChar();
```

Leggere i 2 byte successivi dal flusso di messaggi byte come carattere.

Parametri:
Nessuna

Restituisce:
Il carattere che viene letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadDouble - Leggi numero a virgola mobile di precisione doppia

Interfaccia:

```
Double  ReadDouble();
```

Leggere i successivi 8 byte dal flusso di messaggi byte come un numero a virgola mobile a doppia precisione.

Parametri:
Nessuna

Restituisce:
Il numero a virgola mobile di precisione doppia letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadFloat - Leggi numero a virgola mobile

Interfaccia:

```
Single  ReadFloat();
```

Leggere i successivi 4 byte dal flusso di messaggi byte come numero a virgola mobile.

Parametri:
Nessuna

Restituisce:
Il numero a virgola mobile letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadInt - Lettura numero intero

Interfaccia:

```
Int32   ReadInt();
```

Leggere i successivi 4 byte dal flusso di messaggi di byte come numero intero a 32 bit con segno.

Parametri:
Nessuna

Restituisce:
Il numero intero letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadLong - Lettura numero intero lungo

Interfaccia:

```
Int64   ReadLong();
```

Leggere i successivi 8 byte dal flusso di messaggi di byte come un numero intero a 64 bit con segno.

Parametri:
Nessuna

Restituisce:
Il numero intero lungo letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadShort - Lettura numero intero breve

Interfaccia:

```
Int16   ReadShort();
```

Leggere i 2 byte successivi dal flusso di messaggi di byte come un numero intero a 16 bit con segno.

Parametri:
Nessuna

Restituisce:
Il numero intero breve letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadByte - Lettura byte senza segno

Interfaccia:

```
Byte    ReadByte();
```

Leggere il byte successivo dal flusso di messaggi di byte come un numero intero a 8 bit senza segno.

Parametri:
Nessuna

Restituisce:
Il byte letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadUnsignedShort - Read Unsigned Short Integer

Interfaccia:

```
Int32   ReadUnsignedShort();
```

Leggere i 2 byte successivi dal flusso di messaggi di byte come un numero intero a 16 bit senza segno.

Parametri:
Nessuna

Restituisce:
Il numero intero breve senza segno che viene letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadUTF - Leggi stringa UTF

Interfaccia:

```
String  ReadUTF();
```

Leggere una stringa, codificata in UTF-8, dal flusso di messaggi di byte.

Nota: Prima di richiamare ReadUTF(), assicurarsi che il cursore del buffer punti all'inizio del flusso di messaggi di byte.

Parametri:
Nessuna

Restituisce:
Un oggetto stringa che incapsula la stringa letta.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

Reimposta - Reimposta

Interfaccia:

```
void Reset();
```

Inserire il contenuto del messaggio in modalità di sola lettura e riposizionare il cursore all'inizio del flusso di messaggi byte.

Parametri:
Nessuna

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException

WriteBoolean - Scrivi valore booleano

Interfaccia:

```
void  WriteBoolean(Boolean value);
```

Scrivere un valore booleano nel flusso di messaggi di byte.

Parametri:

valore (immissione)
Il valore booleano da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteByte - Byte di scrittura

Interfaccia:

```
void  WriteByte(Byte value);
void  WriteSignedByte(Int16 value);
```

Scrivere un byte nel flusso di messaggi byte.

Parametri:

valore (immissione)
Il byte da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteBytes - Byte scritti

Interfaccia:

```
void  WriteBytes(Byte[] value);
```

Scrivere un array di byte nel flusso di messaggi di byte.

Parametri:

valore (immissione)
La schiera di byte da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteBytes - Scrivi array di byte parziali

Interfaccia:

```
void  WriteBytes(Byte[] value, int offset, int length);
```

Scrivere un array parziale di byte nel flusso di messaggi di byte, come definito dalla lunghezza specificata.

Parametri:

valore (immissione)
La schiera di byte da scrivere.

offset (input)
Il punto di partenza per la schiera di byte da scrivere.

lunghezza (input)
Il numero di byte da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteChar - Scrivi carattere

Interfaccia:

```
void  WriteChar(Char value);
```

Scrivere un carattere nel flusso di messaggi byte come 2 byte, prima byte di ordine elevato.

Parametri:

valore (immissione)
Il carattere da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

- MessageNotWritableException

WriteDouble - Scrivere il numero a virgola mobile di precisione doppia

Interfaccia:

```
void  WriteDouble(Double value);
```

Convertire un numero a virgola mobile di precisione doppia in un numero intero lungo e scrivere il numero intero lungo nel flusso di messaggi di byte come 8 byte, primo byte di ordine elevato.

Parametri:

valore (immissione)
Il numero a virgola mobile di precisione doppia da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteFloat - Numero virgola mobile di scrittura

Interfaccia:

```
void  WriteFloat(Single value);
```

Converti un numero a virgola mobile in un intero e scrivi il numero intero nel flusso di messaggi di byte come 4 byte, primo byte di ordine elevato.

Parametri:

valore (immissione)
Il numero a virgola mobile da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteInt - Scrittura numero intero

Interfaccia:

```
void  WriteInt(Int32 value);
```

Scrivere un numero intero nel flusso di messaggi di byte come 4 byte, primo byte di ordine superiore.

Parametri:

valore (immissione)
Il numero intero da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteLong - Scrittura numero intero lungo

Interfaccia:

```
void  WriteLong(Int64 value);
```

Scrivere un numero intero lungo nel flusso di messaggi di byte come 8 byte, primo byte di ordine superiore.

Parametri:

valore (immissione)
Il numero intero lungo da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteObject - Scrivi oggetto

Interfaccia:

```
void  WriteObject(Object value);
```

Scrivere l'oggetto specificato nel flusso di messaggi di byte.

Parametri:

valore (immissione)
L'oggetto da scrivere, che deve essere un riferimento a un tipo primitivo.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteShort - Scrivi numero intero breve

Interfaccia:

```
void  WriteShort(Int16 value);
```

Scrivere un intero breve nel flusso di messaggi di byte come 2 byte, prima byte di ordine elevato.

Parametri:

valore (immissione)
Il numero intero breve da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteUTF - Scrivi stringa UTF

Interfaccia:

```
void  WriteUTF(String value);
```

Scrivere una stringa, codificata in UTF-8, nel flusso di messaggi byte.

Parametri:

valore (immissione)
Un oggetto String che incapsula la stringa da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

Proprietà e metodi ereditati

Le seguenti proprietà sono ereditate dall'interfaccia IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedeliveryMode, JMSReplyTo, JMSTimestamp, JMSType, Proprietà

I seguenti metodi sono ereditati dall'interfaccia IMessage:

clearBody, clearProperties, PropertyExists

I seguenti metodi sono ereditati dall'interfaccia IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IConnessione

Un oggetto Connection rappresenta la connessione attiva dell'applicazione ad un server di messaggi.

Gerarchia di eredità:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnection
```

Per un elenco delle XMS proprietà definite di un oggetto Connection, consultare “Proprietà della connessione” a pagina 2085.

Proprietà di .NET

ClientID - Acquisisci e imposta ID client

Interfaccia:

```
String ClientID
      {
        get;
        set;
      }
```

Richiamare e impostare l'identificativo client per la connessione

L'identificativo del client può essere preconfigurato dall'amministratore in un ConnectionFactoryo assegnato impostando ClientID.

Un identificativo client viene utilizzato solo per supportare sottoscrizioni durevoli nel dominio di pubblicazione / sottoscrizione e viene ignorato nel dominio point - to - point.

Se un'applicazione imposta un identificativo client per una connessione, l'applicazione deve farlo immediatamente dopo la creazione della connessione e prima di eseguire qualsiasi altra operazione sulla connessione. Se l'applicazione tenta di impostare un identificativo client dopo questo punto, la chiamata genera l'eccezione IllegalState.

Questa proprietà non è valida per una connessione in tempo reale a un broker.

Eccezioni:

- Eccezione XMSException
- Eccezione IllegalState
- IDException InvalidClient

ExceptionListener - Ottieni e imposta listener di eccezioni

Interfaccia:

```
    ExceptionListener ExceptionListener
        {
          get;
          set;
        }
```

Richiamare il listener di eccezione registrato con la connessione e registrare un listener di eccezione con la connessione

Se non viene registrato alcun listener di eccezione con la connessione, il metodo restituisce un valore null. Se un listener di eccezioni è già registrato con la connessione, è possibile annullare la registrazione specificando un valore null invece del listener di eccezioni.

Per ulteriori informazioni sull'utilizzo dei listener di eccezioni, consultare Utilizzo dei listener di messaggi e di eccezioni in .NET.

Eccezioni:

- Eccezione XMSException

Metadati - ottieni metadati

Interfaccia:

```
IConnectionMetaData MetaData
    {
      get;
    }
```

Ottieni i metadati per la connessione

Eccezioni:

- Eccezione XMSException

Metodi

Chiudi - Chiudi connessione

Interfaccia:

```
void  Close();
```

Chiudere la connessione

Se un'applicazione tenta di chiudere una connessione già chiusa, la chiamata viene ignorata.

Parametri:
Nessuna

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

CreateSession - Crea sessione

Interfaccia:

```
   ISession CreateSession(Boolean transacted,
                       AcknowledgeMode acknowledgeMode);
```

Creare una sessione

Parametri:

transazionali (input)
Il valore `True` indica che la sessione viene eseguita. Il valore `False` indica che la sessione non viene eseguita.

Per una connessione in tempo reale a un broker, il valore deve essere `False`.

acknowledgeMode (input)
Indica in che modo vengono riconosciuti i messaggi ricevuti da un'applicazione. Il valore deve essere uno dei seguenti dall'enumeratore AcknowledgeMode :

`AcknowledgeMode.AutoAcknowledge`
`AcknowledgeMode.ClientAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

Per una connessione in tempo reale ad un broker, il valore deve essere `AcknowledgeMode.AutoAcknowledge` o `AcknowledgeMode.DupsOkAcknowledge`

Questo parametro viene ignorato se viene eseguita la transazione della sessione. Per ulteriori informazioni sulle modalità di riconoscimento, consultare Conferma messaggio.

Restituisce:
L'oggetto Session

Eccezioni:

- Eccezione XMSException

Avvia - Avvia connessione

Interfaccia:

```
void  Start();
```

Avviare o riavviare la consegna dei messaggi in arrivo per la connessione. La chiamata viene ignorata se la connessione è già avviata.

Parametri:
Nessuna

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

Arresta - Arresta connessione

Interfaccia:

```
void  Stop();
```

Arrestare la consegna dei messaggi in entrata per la connessione. La chiamata viene ignorata se la connessione è già stata arrestata.

Parametri:
Nessuna

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IConnectionFactory

Un'applicazione utilizza una produzione connessioni per creare una connessione.

Gerarchia di eredità:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnectionFactory
```

Per un elenco delle XMS proprietà definite di un oggetto ConnectionFactory , consultare “Proprietà di ConnectionFactory” a pagina 2086.

Metodi

CreateConnection - Crea factory di connessione (utilizzando l'identità utente predefinita)

Interfaccia:

```
IConnection CreateConnection();
```

Creare una produzione connessioni con le proprietà predefinite.

Se ci si connette a IBM MQ e XMSC_USERID non è impostato, per impostazione predefinita il gestore code utilizza l' userID dell'utente collegato. Se si richiede un'ulteriore autenticazione a livello di connessione di singoli utenti, è possibile scrivere un'uscita di autenticazione client configurata in IBM MQ.

Parametri:
Nessuna

Eccezioni:

- Eccezione XMSException

CreateConnection - Crea connessione (utilizzando un'identità utente specificata)

Interfaccia:

```
IConnection CreateConnection(String userId, String password);
```

Creare un collegamento utilizzando un'identità utente specificata.

Se ci si connette a IBM MQ e XMSC_USERID non è impostato, per impostazione predefinita il gestore code utilizza l' userID dell'utente collegato. Se si richiede un'ulteriore autenticazione a livello di connessione di singoli utenti, è possibile scrivere un'uscita di autenticazione client configurata in IBM MQ.

La connessione viene creata in modalità arrestata. Nessun messaggio viene consegnato fino a quando l'applicazione non richiama **`Connection.start()`**.

Parametri:

userID (input)
Un oggetto stringa che incapsula l'identificativo utente da utilizzare per autenticare l'applicazione. Se si fornisce un valore null, viene effettuato un tentativo di creare la connessione senza autenticazione.

password (input)
Un oggetto stringa che incapsula la parola d'ordine da utilizzare per autenticare l'applicazione. Se si fornisce un valore null, viene effettuato un tentativo di creare la connessione senza autenticazione.

Restituisce:
L'oggetto Connection.

Eccezioni:

- Eccezione XMSException
- XMS_X_SECURITY_EXCEPTION

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

Dati IConnectionMeta

Un oggetto ConnectionMetaData fornisce informazioni su una connessione.

Gerarchia di eredità:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnectionMetaData
```

Per un elenco delle XMS proprietà definite di un oggetto dati ConnectionMeta, consultare “Proprietà dei dati ConnectionMeta” a pagina 2090.

Proprietà di .NET

JMSXPropertyNames - Richiama proprietà messaggio definite JMS

Interfaccia:

```
System.Collections.IEnumerator JMSXPropertyNames
    {
      get;
    }
```

Restituisce un'enumerazione dei nomi delle JMS proprietà del messaggio definite supportate dalla connessione.

Le proprietà del messaggio definite da JMS non sono supportate da una connessione in tempo reale a un broker.

Eccezioni:

- Eccezione XMSException

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IDestinazione

Una destinazione è il punto in cui un'applicazione invia messaggi o è un'origine da cui un'applicazione riceve messaggi o entrambi.

Gerarchia di eredità:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IDestination
```

Per un elenco delle XMS proprietà definite di un oggetto Destinazione, vedere “Proprietà della destinazione” a pagina 2091.

Proprietà di .NET

Nome - Ottieni nome destinazione

Interfaccia:

```
String Name
{
  get;
}
```

Ottieni il nome della destinazione Il nome è una stringa che incapsula il nome di una coda o il nome di un argomento.

Eccezioni:

- Eccezione XMSException

TypeId - Acquisisci tipo di destinazione

Interfaccia:

```
DestinationType TypeId
{
  get;
}
```

Ottieni il tipo di destinazione. Il tipo di destinazione è uno dei seguenti valori:

```
DestinationType.Queue
DestinationType.Topic
```

Eccezioni:

- Eccezione XMSException

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

ExceptionListener

Un'applicazione utilizza un listener di eccezioni per ricevere una notifica asincrona di un problema con una connessione.

Gerarchia di eredità:
Nessuna

Se un'applicazione utilizza una connessione solo per utilizzare i messaggi in modo asincrono e per nessun altro scopo, l'unico modo in cui l'applicazione può acquisire informazioni su un problema con la connessione è utilizzando un listener di eccezioni. In altre situazioni, un listener di eccezioni può fornire un modo più immediato di conoscere un problema con una connessione piuttosto che attendere la successiva chiamata sincrona a XMS.

Delegato

ExceptionListener - Listener eccezioni

Interfaccia:

```
public delegate void ExceptionListener(Exception ex)
```

Notifica l'applicazione di un problema con una connessione

I metodi che implementano questo delegato possono essere registrati con il collegamento.

Per ulteriori informazioni sull'utilizzo dei listener di eccezioni, consultare Utilizzo dei listener di messaggi e di eccezioni in .NET.

Parametri:

eccezione (input)
Un puntatore a un'eccezione creata da XMS.

Restituisce:
Azzera

Eccezione IllegalState

XMS genera questa eccezione se un'applicazione richiama un metodo in un momento non corretto o inappropriato o se XMS non si trova in uno stato appropriato per l'operazione richiesta.

Gerarchia di eredità:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.Exception
           |
           +----IBM.XMS.IllegalStateException
```

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia XMSException:

CodiceGetError, GetLinkedEccezione

InitialContext

Un'applicazione utilizza un oggetto InitialContext per creare oggetti dalle definizioni di oggetto richiamati da un repository di oggetti gestiti.

Gerarchia di eredità:
Nessuna

Proprietà di .NET

Ambiente - Ottieni l'ambiente

Interfaccia:

```
Hashtable Environment
{
     get;
}
```

Ottieni l'ambiente.

Eccezioni:

- Le eccezioni sono specifiche per il servizio di directory utilizzato.

Costruttori

InitialContext - Crea contesto iniziale

Interfaccia:

```
InitialContext(Hashtable env);
```

Creare un oggetto InitialContext .

Parametri:

Le informazioni richieste per stabilire una connessione al repository di oggetti gestiti vengono fornite al costruttore in un ambiente Hashtable.

Eccezioni:

- Eccezione XMSException

Metodi

Ambiente AddTo- Aggiungi una nuova proprietà all'ambiente

Interfaccia:

```
Object AddToEnvironment(String propName, Object propVal);
```

Aggiungere una nuova proprietà all'ambiente.

Parametri:

propName (input)
Un oggetto String che incapsula il nome della proprietà da aggiungere.

propVal (input)
Il valore della proprietà da aggiungere.

Restituisce:
Il vecchio valore della proprietà.

Eccezioni:

- Le eccezioni sono specifiche per il servizio di directory utilizzato.

Chiudi - Chiudi questo contesto

Interfaccia:

```
void Close()
```

Chiudi questo contesto.

Parametri:
Nessuna

Restituisce:
Nessuna

Eccezioni:

- Le eccezioni sono specifiche per il servizio di directory utilizzato.

Ricerca - Ricerca oggetto nel contesto iniziale

Interfaccia:

```
Object Lookup(String name);
```

Creare un oggetto da una definizione oggetto richiamata dal repository di oggetti gestiti.

Parametri:

nome (input)
Un oggetto stringa che incapsula il nome dell'oggetto gestito da richiamare. Il nome può essere un nome semplice o un nome complesso. Per ulteriori dettagli, consultare Richiamo degli oggetti gestiti.

Restituisce:
Un IConnectionFactory o un IDestination, a seconda del tipo di oggetto richiamato. Se la funzione può accedere all'indirizzario, ma non è in grado di trovare l'oggetto richiesto, viene restituito un valore null.

Eccezioni:

- Le eccezioni sono specifiche per il servizio di directory utilizzato.

Ambiente RemoveFrom- Rimuove una proprietà dall'ambiente

Interfaccia:

```
Object RemoveFromEnvironment(String propName);
```

Rimuovere una proprietà dall'ambiente.

Parametri:

propName (input)
Un oggetto stringa che racchiude il nome della proprietà da eliminare.

Restituisce:
L'oggetto che è stato rimosso.

Eccezioni:

- Le eccezioni sono specifiche per il servizio di directory utilizzato.

IDException InvalidClient

XMS genera questa eccezione se un'applicazione tenta di impostare un identificatore client per una connessione, ma l'identificativo client non è valido o è già in uso.

Gerarchia di eredità:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidClientIDException
```

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia XMSException:

CodiceGetError, GetLinkedEccezione

Eccezione InvalidDestination

XMS genera questa eccezione se un'applicazione specifica una destinazione non valida.

Gerarchia di eredità:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidDestinationException
```

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia XMSException:

CodiceGetError, GetLinkedEccezione

Eccezione InvalidSelector

XMS genera questa eccezione se un'applicazione fornisce un'espressione del selettore di messaggi la cui sintassi non è valida.

Gerarchia di eredità:

```
IBM.XMS.XMSException
   |
```

```
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidSelectorException
```

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia XMSException:

CodiceGetError, GetLinkedEccezione

IMapMessage

Un messaggio della mappa è un messaggio il cui corpo comprende una serie di coppie nome - valore, in cui ogni valore ha un tipo di dati associato.

Gerarchia di eredità:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IMapMessage
```

Quando un'applicazione ottiene il valore della coppia nome - valore, il valore può essere convertito da XMS in un altro tipo di dati. Per ulteriori informazioni su questo tipo di conversione implicita, consultare le informazioni sui messaggi della mappa in Il corpo di un messaggio XMS.

Proprietà di .NET

MapNames - Richiama nomi mappa

Interfaccia:

```
System.Collections.IEnumerator MapNames
{
  get;
}
```

Ottieni un'enumerazione dei nomi nel contenuto del messaggio della mappa.

Eccezioni:

- Eccezione XMSException

Metodi

GetBoolean - Richiama valore booleano

Interfaccia:

```
Boolean GetBoolean(String name);
```

Richiamare il valore booleano identificato dal nome dal contenuto del messaggio della mappa.

Parametri:

nome (input)
Un oggetto String che incapsula il nome che identifica il valore booleano.

Restituisce:
Il valore booleano richiamato dal contenuto del messaggio di associazione.

Eccezioni:

- Eccezione XMSException

GetByte - Acquisisci byte

Interfaccia:

```
Byte    GetByte(String name);
    Int16   GetSignedByte(String name);
```

Richiamare il byte identificato dal nome dal corpo del messaggio della mappa.

Parametri:

nome (input)
Un oggetto stringa che incapsula il nome che identifica il byte.

Restituisce:
Il byte richiamato dal corpo del messaggio di mappa. Nessuna conversione dati viene eseguita sul byte.

Eccezioni:

- Eccezione XMSException

GetBytes - Richiama byte

Interfaccia:

```
Byte[]  GetBytes(String name);
```

Richiamare l'array di byte identificati dal nome dal contenuto del messaggio di associazione.

Parametri:

nome (input)
Un oggetto stringa che incapsula il nome che identifica l'array di byte.

Restituisce:
Il numero di byte nell'array.

Eccezioni:

- Eccezione XMSException

GetChar - Acquisisci carattere

Interfaccia:

```
Char    GetChar(String name);
```

Richiamare il carattere identificato dal nome dal corpo del messaggio della mappa.

Parametri:

nome (input)
Un oggetto String che incapsula il nome che identifica il carattere.

Restituisce:
Il carattere richiamato dal contenuto del messaggio di associazione.

Eccezioni:

- Eccezione XMSException

GetDouble - Acquisisci numero a virgola mobile di precisione doppia

Interfaccia:

```
Double  GetDouble(String name);
```

Richiamare il numero a virgola mobile a doppia precisione identificato dal nome dal contenuto del messaggio della mappa.

Parametri:

nome (input)
Un oggetto String che incapsula il nome che identifica il numero a virgola mobile a doppia precisione.

Restituisce:
Il numero a virgola mobile di precisione doppia richiamato dal corpo del messaggio della mappa.

Eccezioni:

- Eccezione XMSException

GetFloat - Acquisisci numero a virgola mobile

Interfaccia:

```
Single  GetFloat(String name);
```

Richiamare il numero a virgola mobile identificato per nome dal corpo del messaggio della mappa.

Parametri:

nome (input)
Un oggetto String che incapsula il nome che identifica il numero a virgola mobile.

Restituisce:
Il numero a virgola mobile richiamato dal corpo del messaggio della mappa.

Eccezioni:

- Eccezione XMSException

GetInt - Richiamo numero intero

Interfaccia:

```
Int32   GetInt(String name);
```

Richiamare il valore intero identificato dal nome dal corpo del messaggio della mappa.

Parametri:

nome (input)
Un oggetto stringa che incapsula il nome che identifica il numero intero.

Restituisce:
Il numero intero richiamato dal contenuto del messaggio di associazione.

Eccezioni:

- Eccezione XMSException

GetLong - Richiamo numero intero lungo

Interfaccia:

```
Int64   GetLong(String name);
```

Richiamare il numero intero lungo identificato dal nome dal corpo del messaggio della mappa.

Parametri:

nome (input)
Un oggetto stringa che incapsula il nome che identifica il numero intero lungo.

Restituisce:
Il numero intero lungo richiamato dal contenuto del messaggio della mappa.

Eccezioni:

- Eccezione XMSException

GetObject - Acquisisci oggetto

Interfaccia:

```
Object  GetObject(String name);
```

Richiamare un riferimento al valore di una coppia nome - valore, dal contenuto del messaggio della mappa. La coppia nome - valore è identificata dal nome.

Parametri:

nome (input)
Un oggetto String che incapsula il nome della coppia nome - valore.

Restituisce:
Il valore, che è uno dei seguenti tipi di oggetto:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

Eccezioni:
Eccezione XMSException

GetShort - Richiama numero intero breve

Interfaccia:

```
Int16   GetShort(String name);
```

Richiamare il numero intero breve identificato per nome dal corpo del messaggio della mappa.

Parametri:

nome (input)
Un oggetto String che incapsula il nome che identifica il numero intero breve.

Restituisce:
Il numero intero breve richiamato dal corpo del messaggio di associazione.

Eccezioni:

- Eccezione XMSException

GetString - Richiama stringa

Interfaccia:

```
String  GetString(String name);
```

Ottenere la stringa identificata dal nome dal corpo del messaggio della mappa.

Parametri:

nome (input)
Un oggetto stringa che racchiude il nome che identifica la stringa nel contenuto del messaggio della mappa.

Restituisce:
Un oggetto stringa che incapsula la stringa richiamata dal contenuto del messaggio della mappa. Se è richiesta la conversione dei dati, questo valore è la stringa dopo la conversione.

Eccezioni:

- Eccezione XMSException

ItemExists - Verifica che la coppia nome - valore esista

Interfaccia:

```
Boolean ItemExists(String name);
```

Controllare se il corpo del messaggio della mappa contiene una coppia nome - valore con il nome specificato.

Parametri:

nome (input)
Un oggetto String che incapsula il nome della coppia nome - valore.

Restituisce:

- `True`, se il contenuto del messaggio della mappa contiene una coppia nome - valore con il nome specificato.
- `False`, se il corpo del messaggio della mappa non contiene una coppia nome - valore con il nome specificato.

Eccezioni:

- Eccezione XMSException

SetBoolean - Imposta valore booleano

Interfaccia:

```
void  SetBoolean(String name, Boolean value);
```

Impostare un valore booleano nel corpo del messaggio della mappa.

Parametri:

nome (input)
Un oggetto stringa che incapsula il nome per identificare il valore booleano nel contenuto del messaggio della mappa.

valore (immissione)
Il valore booleano da impostare.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

SetByte - Imposta byte

Interfaccia:

```
void  SetByte(String name, Byte value);
void  SetSignedByte(String name, Int16 value);
```

Impostare un byte nel contenuto del messaggio di mappa.

Parametri:

nome (input)
Un oggetto String che incapsula il nome per identificare il byte nel contenuto del messaggio di mappa.

valore (immissione)
Il byte da impostare.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

SetBytes - imposta byte

Interfaccia:

```
void  SetBytes(String name, Byte[] value);
```

Impostare un array di byte nel contenuto del messaggio di mappa.

Parametri:

nome (input)
Un oggetto stringa che incapsula il nome per identificare l'array di byte nel contenuto del messaggio di mappa.

valore (immissione)
L'array di byte da impostare.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

SetChar - Imposta carattere

Interfaccia:

```
void  SetChar(String name, Char value);
```

Impostare un carattere a 2 byte nel contenuto del messaggio di mappa.

Parametri:

nome (input)
Un oggetto stringa che incapsula il nome per identificare il carattere nel contenuto del messaggio della mappa.

valore (immissione)
Il carattere da impostare.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

SetDouble - Imposta numero a virgola mobile di precisione doppia

Interfaccia:

```
void  SetDouble(String name, Double value);
```

Impostare un numero a virgola mobile a precisione doppia nel corpo del messaggio della mappa.

Parametri:

nome (input)
Un oggetto String che incapsula il nome per identificare il numero a virgola mobile di precisione doppia nel contenuto del messaggio della mappa.

valore (immissione)
Il numero a virgola mobile di precisione doppia da impostare.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

SetFloat - Imposta numero a virgola mobile

Interfaccia:

```
void  SetFloat(String name, Single value);
```

Impostare un numero a virgola mobile nel corpo del messaggio della mappa.

Parametri:

nome (input)
Un oggetto String che incapsula il nome per identificare il numero a virgola mobile nel corpo del messaggio di mappa.

valore (immissione)
Il numero a virgola mobile da impostare.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

SetInt - Imposta numero intero

Interfaccia:

```
void  SetInt(String name, Int32 value);
```

Impostare un numero intero nel contenuto del messaggio di associazione.

Parametri:

nome (input)
Un oggetto stringa che incapsula il nome per identificare il intero nel corpo del messaggio della mappa.

valore (immissione)
Il numero intero da impostare.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

SetLong - Imposta numero intero lungo

Interfaccia:

```
void  SetLong(String name, Int64 value);
```

Impostare un numero intero lungo nel corpo del messaggio di mappa.

Parametri:

nome (input)
Un oggetto String che incapsula il nome per identificare il numero intero lungo nel contenuto del messaggio della mappa.

valore (immissione)
Il numero intero lungo da impostare.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

SetObject - Imposta oggetto

Interfaccia:

```
void  SetObject(String name, Object value);
```

Impostare un valore, che deve essere un tipo primitivo XMS , nel contenuto del messaggio della mappa.

Parametri:

nome (input)
Un oggetto stringa che incapsula il nome per identificare il valore nel contenuto del messaggio della mappa.

valore (immissione)
Un array di byte contenente il valore da impostare.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

SetShort - Imposta numero intero breve

Interfaccia:

```
void  SetShort(String name, Int16 value);
```

Impostare un numero intero breve nel corpo del messaggio di mappa.

Parametri:

nome (input)
Un oggetto String che incapsula il nome per identificare il numero intero breve nel contenuto del messaggio della mappa.

valore (immissione)
Il numero intero breve da impostare.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

SetString - Imposta stringa

Interfaccia:

```
void  SetString(String name, String value);
```

Impostare una stringa nel contenuto del messaggio della mappa.

Parametri:

nome (input)
Un oggetto stringa che racchiude il nome per identificare la stringa nel contenuto del messaggio della mappa.

valore (immissione)
Un oggetto String che incapsula la stringa da impostare.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

Proprietà e metodi ereditati

Le seguenti proprietà sono ereditate dall'interfaccia IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedeliveryMode, JMSReplyTo, JMSTimestamp, JMSType, Proprietà

I seguenti metodi sono ereditati dall'interfaccia IMessage:

clearBody, clearProperties, PropertyExists

I seguenti metodi sono ereditati dall'interfaccia IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IMessaggio

Un oggetto Message rappresenta un messaggio che un'applicazione invia o riceve. IMessage è una superclasse per le classi di messaggi come IMapMessage.

Gerarchia di eredità:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
```

Per un elenco dei campi di intestazione del messaggio JMS in un oggetto Messaggio, vedere Campi di intestazione di un messaggio XMS. Per un elenco delle proprietà definite da JMS di un oggetto Messaggio, vedere Proprietà definite da JMS di un messaggio. Per un elenco delle proprietà definite da IBM di

un oggetto Messaggio, vedere IBMdi un messaggio. Per un elenco di proprietà JMS_IBM_MQMD* per l'oggetto Messaggio, consultare “Proprietà JMS_IBM_MQMD*” a pagina 2095

I messaggi vengono eliminati dal raccoglitore dati inutilizzati. Quando un messaggio viene eliminato, questo libera le risorse che stava utilizzando.

Proprietà di .NET

GetJMSCorrelationID - Acquisisci e imposta JMSCorrelationID

Interfaccia:

```
String JMSCorrelationID
{
  get;
  set;
}
```

Richiamare e impostare l'identificatore di correlazione del messaggio come oggetto stringa.

Eccezioni:

- Eccezione XMSException

JMSDeliveryMode - Ottieni e imposta JMSDeliveryMode

Interfaccia:

```
DeliveryMode JMSDeliveryMode
{
  get;
  set;
}
```

Richiamare e impostare la modalità di consegna del messaggio.

La modalità di consegna del messaggio è uno dei seguenti valori:

```
DeliveryMode.Persistent
DeliveryMode.NonPersistent
```

Per un messaggio appena creato che non è stato inviato, la modalità di consegna è `DeliveryMode.Persistente`, tranne che per una connessione in tempo reale a un broker per cui la modalità di consegna è `DeliveryMode.NonPersistent`. Per un messaggio ricevuto, il metodo restituisce la modalità di consegna impostata dalla chiamata IMessageProducer.send () quando il messaggio è stato inviato a meno che l'applicazione ricevente non modifichi la modalità di consegna impostando JMSDeliveryMode.

Eccezioni:

- Eccezione XMSException

JMSDestination - Richiama e imposta JMSDestination

Interfaccia:

```
IDestination JMSDestination
{
  get;
  set;
}
```

Richiamare e impostare la destinazione del messaggio.

La destinazione viene impostata dalla chiamata IMessageProducer.send () quando il messaggio viene inviato. Il valore di JMSDestination viene ignorato. Tuttavia, è possibile utilizzare JMSDestination per modificare la destinazione di un messaggio ricevuto.

Per un messaggio appena creato che non è stato inviato, il metodo restituisce un oggetto Destinazione null, a meno che l'applicazione di invio non imposti una destinazione impostando JMSDestination. Per un messaggio ricevuto, il metodo restituisce un oggetto Destination per la destinazione impostata dalla chiamata IMessageProducer.send () quando il messaggio è stato inviato a meno che l'applicazione ricevente non modifichi la destinazione impostando JMSDestination.

Eccezioni:

- Eccezione XMSException

JMSExpiration - Richiama e imposta JMSExpiration

Interfaccia:

```
Int64 JMSExpiration
{
  get;
  set;
}
```

Richiamare e impostare l'ora di scadenza del messaggio.

L'ora di scadenza viene impostata dalla chiamata IMessageProducer.send () quando il messaggio viene inviato. Il suo valore viene calcolato aggiungendo la durata, come specificato dall'applicazione mittente, all'ora in cui viene inviato il messaggio. L'ora di scadenza è espressa in millisecondi a partire dalle 00:00 GMT del 1 ° gennaio 1970.

Per un messaggio appena creato che non è stato inviato, l'ora di scadenza è 0 a meno che l'applicazione mittente non imposti un'ora di scadenza differente impostando JMSExpiration. Per un messaggio ricevuto, il metodo restituisce l'ora di scadenza impostata dalla chiamata IMessageProducer.send () quando il messaggio è stato inviato, a meno che l'applicazione ricevente non modifichi l'ora di scadenza impostando JMSExpiration.

Se la durata è 0, la chiamata IMessageProducer.send () imposta la scadenza su 0 per indicare che il messaggio non scade.

XMS elimina i messaggi scaduti e non li consegna alle applicazioni.

Eccezioni:

- Eccezione XMSException

JMSMessageID - Richiama e imposta JMSMessageID

Interfaccia:

```
String JMSMessageID
{
  get;
  set;
}
```

Richiamare e impostare l'identificativo del messaggio come un oggetto stringa che incapsula l'identificativo del messaggio.

L'identificativo del messaggi viene impostato dalla chiamata IMessageProducer.send () quando il messaggio viene inviato. Per un messaggio ricevuto, il metodo restituisce l'identificativo del messaggio impostato dalla chiamata IMessageProducer.send () quando il messaggio è stato inviato, a meno che l'applicazione ricevente non modifichi l'identificativo del messaggio impostando JMSMessageID.

Se il messaggio non ha un identificativo di messaggio, il metodo restituisce un valore null.

Eccezioni:

- Eccezione XMSException

JMSPriority - Richiama e imposta JMSPriority

Interfaccia:

```
Int32 JMSPriority
{
  get;
  set;
}
```

Richiamare e impostare la priorità del messaggio.

La priorità viene impostata dalla chiamata IMessageProducer.send () quando il messaggio viene inviato. Il valore è un numero intero compreso nell'intervallo 0, la priorità più bassa, in 9, la priorità più alta.

Per un messaggio appena creato che non è stato inviato, la priorità è 4 a meno che l'applicazione mittente non imposti una priorità differente impostando JMSPriority. Per un messaggio ricevuto, il metodo restituisce la priorità impostata dalla chiamata IMessageProducer.send () quando il messaggio è stato inviato, a meno che l'applicazione ricevente non modifichi la priorità impostando JMSPriority.

Eccezioni:

- Eccezione XMSException

JMSRefornita - Richiama e imposta JMSReconsegnata

Interfaccia:

```
Boolean JMSRedelivered
{
  get;
  set;
}
```

Ottenere un'indicazione se il messaggio è in fase di riconsegnamento e indicare se il messaggio è in fase di riconsegnamento. L'indicazione viene impostata dalla chiamata IMessageConsumer.receive () quando viene ricevuto il messaggio.

Questa proprietà ha i seguenti valori:

- `True`, se il messaggio viene riconsegnato.
- `False`, se il messaggio non viene riconsegnato.

Per una connessione in tempo reale a un broker, il valore è sempre `False`.

Un'indicazione di riconsegna impostata da JMSResent prima dell'invio del messaggio viene ignorata dalla chiamata IMessageProducer.send () quando il messaggio viene inviato e viene ignorata e sostituita dalla chiamata IMessageConsumer.receive () quando il messaggio viene ricevuto. Tuttavia, è possibile utilizzare JMSRenei per modificare l'indicazione per un messaggio ricevuto.

Eccezioni:

- Eccezione XMSException

JMSReplyTo - Get e Set JMSReplyTo

Interfaccia:

```
IDestination JMSReplyTo
{
  get;
  set;
}
```

Richiamare e impostare la destinazione in cui deve essere inviata una risposta al messaggio.

Il valore di questa proprietà è un oggetto Destinazione per la destinazione a cui deve essere inviata una risposta al messaggio. Un oggetto di destinazione null indica che non è prevista alcuna risposta.

Eccezioni:

- Eccezione XMSException

JMSTimestamp - Richiamo e impostazione di JMSTimestamp

Interfaccia:

```
Int64 JMSTimestamp
{
  get;
  set;
}
```

Richiamare e impostare l'ora in cui è stato inviato il messaggio.

La data/ora viene impostata dalla chiamata IMessageProducer.send () quando il messaggio viene inviato ed è espressa in millisecondi a partire dalle 00:00:00 GMT del 1 ° gennaio 1970.

Per un messaggio appena creato che non è stato inviato, la data / ora è 0 a meno che l'applicazione mittente non imposti una data / ora differente impostando JMSTimestamp. Per un messaggio che è stato ricevuto, il metodo restituisce la data / ora impostata dalla chiamata IMessageProducer.send () quando il messaggio è stato inviato, a meno che l'applicazione ricevente non modifichi la data / ora impostando JMSTimestamp.

Eccezioni:

- Eccezione XMSException

Note:

1. Se la data / ora non è definita, il metodo restituisce 0 ma non genera alcuna eccezione.

JMSType - Richiama e imposta JMSType

Interfaccia:

```
String JMSType
{
  get;
  set;
}
```

Richiamare e impostare il tipo di messaggio.

Il valore di JMSType è una stringa che incapsula il tipo del messaggio. Se è richiesta la conversione dei dati, questo valore è il tipo dopo la conversione.

Eccezioni:

- Eccezione XMSException

PropertyNames - Richiama proprietà

Interfaccia:

```
System.Collections.IEnumerator PropertyNames
{
  get;
}
```

Richiamare un'enumerazione delle proprietà dei nomi del messaggio.

Eccezioni:

- Eccezione XMSException

Metodi

Riconoscimento - Riconosci

Interfaccia:

```
void  Acknowledge();
```

Riconoscere questo messaggio e tutti i messaggi precedentemente non riconosciuti ricevuti dalla sessione.

Un'applicazione può richiamare questo metodo se la modalità di riconoscimento della sessione è AcknowledgeMode.ClientAcknowledge. Le chiamate al metodo vengono ignorate se la sessione ha un'altra modalità di riconoscimento o se viene eseguita la transazione.

I messaggi ricevuti ma non riconosciuti potrebbero essere riconsegnati.

Per ulteriori informazioni sul riconoscimento dei messaggi, consultare ../develop/xms_cmesack.dita#xms_cmesack.

Parametri:
Nessuna

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- Eccezione IllegalState

ClearBody - Cancella body

Interfaccia:

```
void  ClearBody();
```

Cancella il corpo del messaggio. I campi di intestazione e le proprietà del messaggio non vengono cancellati.

Se un'applicazione cancella il contenuto di un messaggio, il corpo rimane nello stesso stato di un corpo vuoto in un messaggio appena creato. Lo stato di un corpo vuoto in un messaggio appena creato dipende dal tipo di corpo del messaggio. Per ulteriori informazioni, consultare Il corpo di un messaggio XMS.

Un'applicazione può cancellare il contenuto di un messaggio in qualsiasi momento, indipendentemente dallo stato in cui si trova il corpo. Se il corpo di un messaggio è di sola lettura, l'unico modo in cui un'applicazione può scrivere nel corpo è che l'applicazione cancella prima il corpo.

Parametri:
Nessuna

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

ClearProperties - Cancella proprietà

Interfaccia:

```
void  ClearProperties();
```

Cancella le proprietà del messaggio. I campi di intestazione e il corpo del messaggio non vengono cancellati.

Se un'applicazione cancella le proprietà di un messaggio, le proprietà diventano leggibili e scrivibili.

Un'applicazione può cancellare le proprietà di un messaggio in qualsiasi momento, indipendentemente dallo stato in cui si trovano le proprietà. Se le proprietà di un messaggio sono di sola lettura, l'unico modo in cui le proprietà possono diventare scrivibili è che l'applicazione cancella prima le proprietà.

Parametri:
Nessuna

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

PropertyExists - Verifica proprietà esistente

Interfaccia:

```
Boolean PropertyExists(String propertyName);
```

Controllare se il messaggio ha una proprietà con il nome specificato.

Parametri:

propertyName (input)
Un oggetto String che incapsula il nome della proprietà.

Restituisce:

- `True`, se il messaggio ha una proprietà con il nome specificato.
- `False`, se il messaggio non ha una proprietà con il nome specificato.

Eccezioni:

- Eccezione XMSException

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IMessageConsumer

Un'applicazione utilizza un consumatore di messaggi per ricevere i messaggi inviati a una destinazione.

Gerarchia di eredità:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessageConsumer
```

Per un elenco delle proprietà XMS definite di un oggetto MessageConsumer , consultare “Proprietà di MessageConsumer” a pagina 2098.

Proprietà di .NET

MessageListener - Richiama e imposta listener messaggi

Interfaccia:

```
MessageListener MessageListener
{
  get;
  set;
}
```

Richiamare il listener dei messaggi registrato con il consumer dei messaggi e registrare un listener dei messaggi con tale consumer.

Se nessun listener di messaggi è registrato con il consumatore di messaggi, MessageListener è null. Se un listener di messaggi è già registrato con il consumatore di messaggi, è possibile annullare la registrazione specificando invece un valore null.

Per ulteriori informazioni sull'utilizzo dei listener di messaggi, consultare Utilizzo dei listener di messaggi ed eccezioni in .NET.

Eccezioni:

- Eccezione XMSException

MessageSelector - Richiama selettore messaggi

Interfaccia:

```
String MessageSelector
{
  get;
}
```

Richiamare il selettore del messaggio per il destinatario del messaggio. Il valore di ritorno è un oggetto String che incapsula l'espressione del selettore messaggi. Se è richiesta la conversione dei dati, questo valore è l'espressione del selettore di messaggi dopo la conversione. Se il consumer del messaggio non dispone di un selettore di messaggi, il valore di MessageSelector è un oggetto String null.

Eccezioni:

- Eccezione XMSException

Metodi

Chiudi - Chiudi consumer messaggi

Interfaccia:

```
void  Close();
```

Chiudere il destinatario del messaggio

Se un'applicazione tenta di chiudere un consumatore di messaggi già chiuso, la chiamata viene ignorata.

Parametri:
Nessuna

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

Ricezione - Ricezione

Interfaccia:

```
IMessage Receive();
```

Ricevere il messaggio successivo per il destinatario del messaggio. La chiamata attende indefinitamente un messaggio o fino a quando il destinatario del messaggio non viene chiuso.

Parametri:
Nessuna

Restituisce:
Un puntatore all'oggetto Messaggio. Se il destinatario del messaggio viene chiuso mentre la chiamata è in attesa di un messaggio, il metodo restituisce un puntatore a un oggetto Messaggio null.

Eccezioni:

- Eccezione XMSException

Ricezione - Ricezione (con intervallo di attesa)

Interfaccia:

```
IMessage Receive(Int64 delay);
```

Ricevere il messaggio successivo per il destinatario del messaggio. La chiamata attende solo un periodo specificato per un messaggio o fino a quando il destinatario del messaggio non viene chiuso.

Parametri:

ritardo (input)
Il tempo, in millisecondi, in cui la chiamata attende un messaggio. Se si specifica un intervallo di attesa di 0, la chiamata attende indefinitamente un messaggio.

Restituisce:
Un puntatore all'oggetto Messaggio. Se non arriva alcun messaggio durante l'intervallo di attesa o se l'utente del messaggio è chiuso mentre la chiamata è in attesa di un messaggio, il metodo restituisce un puntatore a un oggetto Messaggio null ma non genera alcuna eccezione.

Eccezioni:

- Eccezione XMSException

ReceiveNoAttesa - Ricezione senza attesa

Interfaccia:

```
IMessage ReceiveNoWait();
```

Ricevere il messaggio successivo per il destinatario del messaggio, se disponibile immediatamente.

Parametri:
Nessuna

Restituisce:
Un puntatore ad un oggetto Messaggio. Se nessun messaggio è disponibile immediatamente, il metodo restituisce un puntatore a un oggetto Message null.

Eccezioni:

- Eccezione XMSException

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

MessageEOFException

XMS genera questa eccezione se XMS rileva la fine di un flusso di messaggi di byte quando un'applicazione sta leggendo il contenuto di un messaggio di byte.

Gerarchia di eredità:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageEOFException
```

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia XMSException:

CodiceGetError, GetLinkedEccezione

Eccezione MessageFormat

XMS genera questa eccezione se XMS rileva un messaggio con un formato non valido.

Gerarchia di eredità:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageFormatException
```

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia XMSException:

CodiceGetError, GetLinkedEccezione

IMessageListener (delegato)

Un'applicazione utilizza un listener di messaggi per ricevere i messaggi asincrono.

Gerarchia di eredità:
Nessuna

Delegato

MessageListener - Listener messaggi

Interfaccia:

```
public delegate void MessageListener(IMessage msg);
```

Consegnare un messaggio in modo asincrono al destinatario del messaggio

I metodi che implementano questo delegato possono essere registrati con il collegamento.

Per ulteriori informazioni sull'utilizzo dei listener di messaggi, consultare Utilizzo dei listener di messaggi ed eccezioni in .NET.

Parametri:

mesg (input)
L'oggetto Messaggio.

Restituisce:
Azzera

MessageNotReadableException

XMS genera questa eccezione se un'applicazione tenta di leggere il corpo di un messaggio di sola scrittura.

Gerarchia di eredità:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageNotReadableException
```

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia XMSException:

CodiceGetError, GetLinkedEccezione

MessageNotWritableException

XMS genera questa eccezione se un'applicazione tenta di scrivere nel corpo di un messaggio di sola lettura.

Gerarchia di eredità:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageNotWritableException
```

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia XMSException:

CodiceGetError, GetLinkedEccezione

IMessageProducer

Un'applicazione utilizza un produttore di messaggi per inviare messaggi a una destinazione.

Gerarchia di eredità:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessageProducer
```

Per un elenco delle XMS proprietà definite di un oggetto MessageProducer , consultare “Proprietà di MessageProducer” a pagina 2098.

Proprietà di .NET

DeliveryMode - Richiama e imposta la modalità di consegna predefinita

Interfaccia:

```
DeliveryMode DeliveryMode
{
  get;
  set;
}
```

Richiamare e impostare il modo di consegna predefinito per i messaggi inviati dal produttore del messaggio.

La modalità di consegna predefinita ha uno dei seguenti valori:

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

Per una connessione in tempo reale a un broker, il valore deve essere `DeliveryMode.NonPersistent`.

Il valore predefinito è `DeliveryMode.Persistent`, tranne per una connessione in tempo reale a un broker per cui il valore predefinito è `DeliveryMode.NonPersistent`.

Eccezioni:

- Eccezione XMSException

Destinazione - Ottieni destinazione

Interfaccia:

```
IDestination Destination
{
  get;
}
```

Ottieni la destinazione per il produttore del messaggio.

Parametri:
Nessuna

Restituisce:
L'oggetto Destinazione. Se il produttore del messaggio non dispone di una destinazione, il metodo restituisce un oggetto Destinazione null.

Eccezioni:

- Eccezione XMSException

DisableMsgID - Richiama e imposta indicatore di disabilitazione ID messaggio

Interfaccia:

```
Boolean DisableMessageID
{
  get;
  set;
}
```

Indica se un'applicazione ricevente richiede che gli identificativi dei messaggi siano inclusi nei messaggi inviati dal produttore del messaggio e se un'applicazione ricevente richiede che gli identificativi dei messaggi siano inclusi nei messaggi inviati dal produttore del messaggio.

In una connessione a un gestore code o in una connessione in tempo reale a un broker, questo indicatore viene ignorato. Su una connessione a un bus di integrazione servizi, l'indicatore viene rispettato.

L'ID DisabledMsgha i valori seguenti:

- `True`, se un'applicazione di ricezione non richiede che gli identificativi di messaggio siano inclusi nei messaggi inviati dal produttore del messaggio.
- `False`, se un'applicazione ricevente richiede che gli identificativi dei messaggi siano inclusi nei messaggi inviati dal produttore del messaggio.

Eccezioni:

- Eccezione XMSException

DisableMsgTS - Richiama e imposta indicatore di disabilitazione data/ora

Interfaccia:

```
Boolean DisableMessageTimestamp
{
  get;
  set;
}
```

Indica se un'applicazione di ricezione richiede che la data / ora sia inclusa nei messaggi inviati dal produttore del messaggio e indica se un'applicazione di ricezione richiede che la data / ora sia inclusa nei messaggi inviati dal produttore del messaggio.

In una connessione in tempo reale a un broker, questo indicatore viene ignorato. Su una connessione a un gestore code o su una connessione a un SIB (service integration bus), l'indicatore viene rispettato.

DisableMsgTS ha i valori seguenti:

- `True`, se un'applicazione ricevente non richiede che le date / ore siano incluse nei messaggi inviati dal produttore del messaggio.
- `False`, se un'applicazione ricevente richiede che le date / ore siano incluse nei messaggi inviati dal produttore del messaggio.

Restituisce:

Eccezioni:

- Eccezione XMSException

Priorità - Ottieni e imposta priorità predefinita

Interfaccia:

```
Int32 Priority
{
  get;
  set;
}
```

Richiamare e impostare la priorità predefinita per i messaggi inviati dal produttore del messaggio.

Il valore della priorità predefinita del messaggio è un numero intero compreso tra `0`, la priorità più bassa, e `9`, la priorità più elevata.

In una connessione in tempo reale a un broker, la priorità di un messaggio viene ignorata.

Eccezioni:

- Eccezione XMSException

TimeToLive - Ottieni e imposta il TTL (Time to Live) predefinito

Interfaccia:

```
Int64 TimeToLive
```

```
{
  get;
  set;
}
```

Richiamare e impostare il periodo di tempo predefinito per cui un messaggio esiste prima che scada.

Il tempo viene misurato dal momento in cui il mittente del messaggio invia il messaggio ed è il tempo di vita predefinito in millisecondi. Il valore 0 indica che un messaggio non scade mai.

Per una connessione in tempo reale a un broker, questo valore è sempre 0.

Eccezioni:

- Eccezione XMSException

Metodi

Chiudi - Chiudi produttore messaggi

Interfaccia:

```
void  Close();
```

Chiudere il mittente del messaggio.

Se un'applicazione tenta di chiudere un produttore di messaggi che è già chiuso, la chiamata viene ignorata.

Parametri:
Nessuna

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

Invia - Invia

Interfaccia:

```
void Send(IMessage msg) ;
```

Inviare un messaggio alla destinazione specificata quando è stato creato il produttore del messaggio. Inviare il messaggio utilizzando la modalità di distribuzione predefinita del produttore del messaggio, la priorità e il time to live.

Parametri:

msg (input)
L'oggetto Messaggio.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- Eccezione MessageFormat
- Eccezione InvalidDestination

Invia - Invia (specificando una modalità di consegna, una priorità e un TTL (time to live))

Interfaccia:

```
void Send(IMessage msg,
          DeliveryMode deliveryMode,
          Int32 priority,
          Int64 timeToLive);
```

Inviare un messaggio alla destinazione specificata quando è stato creato il produttore del messaggio. Inviare il messaggio utilizzando la modalità di consegna, la priorità e la durata specificate.

Parametri:

msg (input)
L'oggetto Messaggio.

deliveryMode (input)
La modalità di consegna per il messaggio, che deve essere uno dei seguenti valori:

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

Per una connessione in tempo reale a un broker, il valore deve essere `DeliveryMode.NonPersistent`.

priorità (input)
La priorità del messaggio. Il valore può essere un numero intero compreso nell'intervallo `0`, per la priorità più bassa, per 9, per la priorità più alta. In una connessione in tempo reale a un broker, il valore viene ignorato.

timeToLive (input)
La durata del messaggio in millisecondi. Il valore 0 indica che il messaggio non scade mai. Per una connessione in tempo reale a un broker, il valore deve essere 0.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- Eccezione MessageFormat
- Eccezione InvalidDestination
- Eccezione IllegalState

Invia - Invia (a una destinazione specificata)

Interfaccia:

```
void Send(IDestination dest, IMessage msg) ;
```

Inviare un messaggio a una destinazione specificata se si sta utilizzando un produttore di messaggi per cui non è stata specificata alcuna destinazione quando è stato creato il produttore di messaggi. Inviare il messaggio utilizzando la modalità di distribuzione predefinita del produttore del messaggio, la priorità e il time to live.

Generalmente, si specifica una destinazione quando si crea un produttore di messaggi ma, in caso contrario, è necessario specificare una destinazione ogni volta che si invia un messaggio.

Parametri:

dest (input)
L'oggetto Destinazione.

msg (input)
L'oggetto Messaggio.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- Eccezione MessageFormat
- Eccezione InvalidDestination

Invia - Invia (a una destinazione specificata, specificando una modalit ... di consegna, una priorit ... e la durata)

Interfaccia:

```
void Send(IDestination dest,
          IMessage msg,
          DeliveryMode deliveryMode,
          Int32 priority,
          Int64 timeToLive) ;
```

Inviare un messaggio a una destinazione specificata se si sta utilizzando un produttore di messaggi per cui non è stata specificata alcuna destinazione quando è stato creato il produttore di messaggi. Inviare il messaggio utilizzando la modalità di consegna, la priorità e la durata specificate.

Generalmente, si specifica una destinazione quando si crea un produttore di messaggi ma, in caso contrario, è necessario specificare una destinazione ogni volta che si invia un messaggio.

Parametri:

dest (input)
L'oggetto Destinazione.

msg (input)
L'oggetto Messaggio.

deliveryMode (input)
La modalità di consegna per il messaggio, che deve essere uno dei seguenti valori:

```
DeliveryMode.Persistent
DeliveryMode.NonPersistent
```

Per una connessione in tempo reale a un broker, il valore deve essere `DeliveryMode.NonPersistent`.

priorità (input)
La priorità del messaggio. Il valore può essere un numero intero compreso nell'intervallo 0, per la priorità più bassa, per 9, per la priorità più alta. In una connessione in tempo reale a un broker, il valore viene ignorato.

timeToLive (input)
La durata del messaggio in millisecondi. Il valore 0 indica che il messaggio non scade mai. Per una connessione in tempo reale a un broker, il valore deve essere 0.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- Eccezione MessageFormat
- Eccezione InvalidDestination
- Eccezione IllegalState

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IObjectMessage

Un messaggio oggetto è un messaggio il cui corpo comprende un oggetto Java o .NET serializzato.

Gerarchia di eredità:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IObjectMessage
```

Proprietà di .NET

Oggetto - Ottieni e imposta oggetto come byte

Interfaccia:

```
System.Object Object
    {
      get;
      set;
    }

    Byte[] GetObject();
```

Richiamare e impostare l'oggetto che costituisce il corpo del messaggio oggetto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException
- MessageNotWritableException

Proprietà e metodi ereditati

Le seguenti proprietà sono ereditate dall'interfaccia IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedeliveryMode, JMSReplyTo, JMSTimestamp, JMSType, Proprietà

I seguenti metodi sono ereditati dall'interfaccia IMessage:

clearBody, clearProperties, PropertyExists

I seguenti metodi sono ereditati dall'interfaccia IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IPropertyContext

IPropertyContext è una superclasse astratta che contiene metodi che richiamano e impostano le proprietà. Questi metodi vengono ereditati dalle altre classi.

Gerarchia di eredità:
Nessuna

Metodi

Proprietà GetBoolean- Richiama proprietà booleana

Interfaccia:

```
Boolean GetBooleanProperty(String property_name);
```

Richiama il valore della proprietà booleana con il nome specificato.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

Restituisce:
Il valore della proprietà.

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException

Proprietà GetByte- Proprietà Get Byte

Interfaccia:

```
Byte    GetByteProperty(String property_name) ;
Int16   GetSignedByteProperty(String property_name) ;
```

Richiamare il valore della proprietà byte identificata dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

Restituisce:
Il valore della proprietà.

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException

Proprietà GetBytes- proprietà Richiamo matrice di byte

Interfaccia:

```
Byte[]  GetBytesProperty(String property_name) ;
```

Richiama il valore della proprietà dell'array di byte identificato dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

Restituisce:
Il numero di byte nell'array.

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException

Proprietà GetChar- Acquisisci proprietà carattere

Interfaccia:

```
Char    GetCharProperty(String property_name) ;
```

Richiamare il valore della proprietà di caratteri a 2 byte identificata dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

Restituisce:
Il valore della proprietà.

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException

Proprietà GetDouble- Richiamo proprietà a virgola mobile di precisione doppia

Interfaccia:

```
Double  GetDoubleProperty(String property_name) ;
```

Richiama il valore della proprietà a virgola mobile a doppia precisione identificata per nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

Restituisce:
Il valore della proprietà.

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException

Proprietà GetFloat- Acquisisci proprietà a virgola mobile

Interfaccia:

```
Single  GetFloatProperty(String property_name) ;
```

Ottieni il valore della proprietà a virgola mobile identificata per nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

Restituisce:
Il valore della proprietà.

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException

Proprietà GetInt- Proprietà GetInt

Interfaccia:

```
Int32   GetIntProperty(String property_name) ;
```

Richiamare il valore della proprietà integer identificata dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

Restituisce:
Il valore della proprietà.

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException

Proprietà GetLong- Richiamo proprietà numero intero lungo

Interfaccia:

```
Int64   GetLongProperty(String property_name) ;
```

Richiamare il valore della proprietà long integer identificata dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

Restituisce:
Il valore della proprietà.

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException

Proprietà GetObject- Acquisisci proprietà oggetto

Interfaccia:

```
Object  GetObjectProperty( String property_name) ;
```

Ottieni il valore e il tipo di dati della proprietà identificata per nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

Restituisce:
Il valore della proprietà, che è uno dei seguenti tipi di oggetto:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException

Proprietà GetShort- Richiamo proprietà numero intero breve

Interfaccia:

```
Int16   GetShortProperty(String property_name) ;
```

Richiama il valore della proprietà numero intero breve identificata dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

Restituisce:
Il valore della proprietà.

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException

Proprietà GetString- GetString

Interfaccia:

```
String  GetStringProperty(String property_name) ;
```

Richiamare il valore della proprietà della stringa identificata per nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

Restituisce:
Un oggetto String che incapsula la stringa che è il valore della proprietà. Se è richiesta la conversione dei dati, questo valore è la stringa dopo la conversione.

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException

Proprietà SetBoolean- Imposta proprietà booleana

Interfaccia:

```
void SetBooleanProperty( String property_name,  Boolean value) ;
```

Impostare il valore della proprietà booleana identificata dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

valore (immissione)
Il valore della proprietà.

Restituisce:
Azzera

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

Proprietà SetByte- Imposta proprietà byte

Interfaccia:

```
void SetByteProperty( String property_name,  Byte value) ;
      void SetSignedByteProperty( String property_name,  Int16 value) ;
```

Impostare il valore della proprietà byte identificata dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

valore (immissione)
Il valore della proprietà.

Restituisce:
Azzera

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

Proprietà SetBytes- Imposta proprietà matrice di byte

Interfaccia:

```
void SetBytesProperty( String property_name, Byte[] value ) ;
```

Impostare il valore della proprietà della matrice di byte identificata dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

valore (immissione)
Il valore della proprietà, che è un array di byte.

Restituisce:
Azzera

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

Proprietà SetChar- Imposta proprietà carattere

Interfaccia:

```
void SetCharProperty( String property_name,  Char value) ;
```

Impostare il valore della proprietà di caratteri a 2 byte identificata dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

valore (immissione)
Il valore della proprietà.

Restituisce:
Azzera

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

Proprietà SetDouble- Imposta proprietà a virgola mobile di precisione doppia

Interfaccia:

```
void SetDoubleProperty( String property_name,  Double value) ;
```

Impostare il valore della proprietà a virgola mobile di precisione doppia identificata dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

valore (immissione)
Il valore della proprietà.

Restituisce:
Azzera

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

Proprietà SetFloat- Imposta proprietà a virgola mobile

Interfaccia:

```
void SetFloatProperty( String property_name,  Single value) ;
```

Impostare il valore della proprietà a virgola mobile identificato dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

valore (immissione)
Il valore della proprietà.

Restituisce:
Azzera

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

Proprietà SetInt- Imposta proprietà numero intero

Interfaccia:

```
void SetIntProperty( String property_name,  Int32 value) ;
```

Impostare il valore della proprietà integer identificata dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

valore (immissione)
Il valore della proprietà.

Restituisce:
Azzera

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

Proprietà SetLong- Imposta proprietà numero intero lungo

Interfaccia:

```
void SetLongProperty( String property_name,  Int64 value) ;
```

Impostare il valore della proprietà long integer identificata dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

valore (immissione)
Il valore della proprietà.

Restituisce:
Azzera

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

Proprietà SetObject- Imposta proprietà oggetto

Interfaccia:

```
void SetObjectProperty( String property_name,  Object value) ;
```

Impostare il valore e il tipo di dati di una proprietà identificata per nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

objectType (input)
Il valore della proprietà, che deve essere uno dei seguenti tipi di oggetto:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

valore (immissione)
Il valore della proprietà come un array di byte.

lunghezza (input)
Il numero di byte nell'array.

Restituisce:
Azzera

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

Proprietà SetShort- Imposta proprietà numero intero breve

Interfaccia:

```
void SetShortProperty( String property_name,  Int16 value) ;
```

Impostare il valore della proprietà short integer identificata dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

valore (immissione)
Il valore della proprietà.

Restituisce:
Azzera

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

Proprietà SetString- Imposta proprietà stringa

Interfaccia:

```
void SetStringProperty( String property_name,  String value);
```

Impostare il valore della proprietà stringa identificata dal nome.

Parametri:

nome_proprietà (input)
Un oggetto String che incapsula il nome della proprietà.

valore (immissione)
Un oggetto String che incapsula la stringa che è il valore della proprietà.

Restituisce:
Azzera

Contesto thread:
Determinato dalla sottoclasse

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

IQueueBrowser

Un'applicazione utilizza un browser della coda per ricercare i messaggi su una coda senza rimuoverli.

Gerarchia di eredità:

```
IBM.XMS.IPropertyContext
System.Collections.IEnumerable
   |
   +----IBM.XMS.IQueueBrowser
```

Proprietà di .NET

MessageSelector - Richiama selettore messaggi

Interfaccia:

```
String MessageSelector
{
  get;
}
```

Richiamare il selettore messaggi per il browser della coda.

Il selettore messaggi è un oggetto stringa che incapsula l'espressione del selettore messaggi. Se è richiesta la conversione dei dati, questo valore è l'espressione del selettore di messaggi dopo la conversione. Se il browser della coda non dispone di un selettore di messaggi, il metodo restituisce un oggetto String null.

Eccezioni:

- Eccezione XMSException

Coda - Coda di ricezione

Interfaccia:

```
IDestination Queue
{
  get;
}
```

Richiamare la coda associata al browser della coda come oggetto di destinazione che rappresenta la coda.

Eccezioni:

- Eccezione XMSException

Metodi

Chiudi - Chiudi browser coda

Interfaccia:

```
void  Close();
```

Chiudere il browser della coda.

Se un'applicazione tenta di chiudere un browser della coda già chiuso, la chiamata viene ignorata.

Parametri:
Nessuna

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

GetEnumerator - Acquisisci messaggi

Interfaccia:

```
IEnumerator GetEnumerator();
```

Richiamare un elenco dei messaggi sulla coda.

Il metodo restituisce un enumeratore che incapsula un elenco di oggetti Messaggio. L'ordine degli oggetti Messaggio è uguale a quello in cui i messaggi vengono richiamati dalla coda. L'applicazione può quindi utilizzare l'enumeratore per esaminare ogni messaggio a turno.

L'enumeratore viene aggiornato dinamicamente quando i messaggi vengono inseriti nella coda e rimossi dalla coda. Ogni volta che l'applicazione richiama IEnumerator.MoveNext() per sfogliare il messaggio successivo nella coda, il messaggio riflette il contenuto corrente della coda.

Se un'applicazione richiama questo metodo più di una volta per un browser della coda, ogni chiamata restituisce un nuovo enumeratore. L'applicazione può quindi utilizzare più di un enumeratore per esaminare i messaggi su una coda e mantenere più posizioni all'interno della coda.

Parametri:
Nessuna

Restituisce:
L'oggetto iteratore.

Eccezioni:

- Eccezione XMSException

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

Richiedente

Un'applicazione utilizza un richiedente per inviare un messaggio di richiesta e attendere e ricevere la risposta.

Gerarchia di eredità:
Nessuna

Costruttori

Richiedente - Crea richiedente

Interfaccia:

```
Requestor(ISession sess, IDestination dest);
```

Creare un richiedente.

Parametri:

sess (input)
Un oggetto Session. La sessione non deve essere sottoposta a transazione e deve avere una delle seguenti modalità di riconoscimento:

`AcknowledgeMode.AutoAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

dest (input)
Un oggetto Destinazione che rappresenta la destinazione in cui l'applicazione può inviare i messaggi di richiesta.

Contesto thread:
La sessione associata al richiedente

Eccezioni:

- Eccezione XMSException

Metodi

Chiudi - Chiudi richiedente

Interfaccia:

```
void  Close();
```

Chiudere il richiedente.

Se un'applicazione tenta di chiudere un richiedente già chiuso, la chiamata viene ignorata.

Nota: Quando un'applicazione chiude un richiedente, anche la sessione associata non viene chiusa. A questo proposito, XMS si comporta in modo diverso rispetto a JMS.

Parametri:
Nessuna

Restituisce:
Azzera

Contesto thread:
Qualsiasi

Eccezioni:

- Eccezione XMSException

Richiesta - Richiesta risposta

Interfaccia:

```
IMessage Request(IMessage requestMessage);
```

Inviare un messaggio di richiesta, quindi attendere e ricevere una risposta dall'applicazione che riceve il messaggio di richiesta.

Una chiamata a questo metodo si blocca fino a quando non viene ricevuta una risposta o fino alla fine della sessione, a seconda di quale sia la prima.

Parametri:

requestMessage (input)
L'oggetto Messaggio che incapsula il messaggio di richiesta.

Restituisce:
Un puntatore all'oggetto Messaggio che incapsula il messaggio di risposta.

Contesto thread:
La sessione associata al richiedente

Eccezioni:

- Eccezione XMSException

Eccezione ResourceAllocation

XMS genera questa eccezione se XMS non può allocare le risorse richieste da un metodo.

Gerarchia di eredità:

```
IBM.XMS.XMSException
   |
```

```
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.ResourceAllocationException
```

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia XMSException:

CodiceGetError, GetLinkedEccezione

SecurityException

XMS genera questa eccezione se l'identificativo utente e la parola d'ordine forniti per autenticare un'applicazione vengono rifiutati. XMS genera questa eccezione anche se un controllo di autorizzazione non riesce e impedisce il completamento di un metodo.

Gerarchia di eredità:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.SecurityException
```

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia XMSException:

CodiceGetError, GetLinkedEccezione

ISessione

Una sessione è un contesto a thread singolo per l'invio e la ricezione di messaggi.

Gerarchia di eredità:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.ISession
```

Per un elenco delle proprietà definite XMS di un oggetto Session, vedere “Proprietà della sessione” a pagina 2098.

Proprietà di .NET

AcknowledgeMode - Modalità ricezione riconoscimento

Interfaccia:

```
AcknowledgeMode AcknowledgeMode
    {
      get;
    }
```

Richiamare la modalità di riconoscimento per la sessione

La modalità di riconoscimento viene specificata quando la sessione viene creata.

Se la sessione non è sottoposta a transazione, la modalità di riconoscimento è uno dei seguenti valori:

`AcknowledgeMode.AutoAcknowledge`
`AcknowledgeMode.ClientAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

Per ulteriori informazioni sulle modalità di riconoscimento, consultare Conferma messaggio.

Una sessione sottoposta a transazione non dispone di una modalità di riconoscimento. Se la sessione viene eseguita, il metodo restituisce `AcknowledgeMode.SessionTransacted` .

Eccezioni:

- Eccezione XMSException

Transazionali - Determinare se transazionali

Interfaccia:

```
Boolean Transacted
    {
      get;
    }
```

Determinare se la sessione è sottoposta a transazione.

La transazione indicata è:

- True, se la sessione è sottoposta a transazione.
- False, se la sessione non viene eseguita.

Per una connessione in tempo reale a un broker, il metodo restituisce sempre False.

Eccezioni:

- Eccezione XMSException

Metodi

Chiudi - Chiudi sessione

Interfaccia:

```
void Close();
```

Chiudere la sessione. Se la sessione viene sottoposta a transazione, viene eseguito il rollback di qualsiasi transazione in corso.

Se un'applicazione tenta di chiudere una sessione già chiusa, la chiamata viene ignorata.

Parametri:
Nessuna

Restituisce:
Azzera

Contesto thread:
Qualsiasi

Eccezioni:

- Eccezione XMSException

Commit - Commit

Interfaccia:

```
void Commit();
```

Sincronizzare tutti i messaggi elaborati nella transazione corrente

La sessione deve essere una sessione transattiva.

Parametri:
Nessuna

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- Eccezione IllegalState
- TransactionRolledBackException

CreateBrowser - Crea browser coda

Interfaccia:

```
IQueueBrowser CreateBrowser(IDestination queue) ;
```

Creare un browser di code per la coda specificata.

Parametri:

coda (input)
Un oggetto di destinazione che rappresenta la coda.

Restituisce:
L'oggetto QueueBrowser .

Eccezioni:

- Eccezione XMSException
- Eccezione InvalidDestination

CreateBrowser - Crea browser della coda (con selettore messaggi)

Interfaccia:

```
IQueueBrowser CreateBrowser(IDestination queue, String selector) ;
```

Creare un browser di code per la coda specificata utilizzando un selettore di messaggi.

Parametri:

coda (input)
Un oggetto di destinazione che rappresenta la coda.

selettore (input)
Un oggetto stringa che incapsula un'espressione del selettore messaggi. Solo i messaggi con proprietà che corrispondono all'espressione del selettore messaggi vengono consegnati al browser della coda.

Un oggetto stringa null indica che non è presente alcun selettore di messaggi per il browser della coda.

Restituisce:
L'oggetto QueueBrowser .

Eccezioni:

- Eccezione XMSException
- Eccezione InvalidDestination
- Eccezione InvalidSelector

Messaggio CreateBytes- Crea messaggio di byte

Interfaccia:

```
IBytesMessage CreateBytesMessage();
```

Creare un messaggio byte.

Parametri:
Nessuna

Restituisce:
L'oggetto BytesMessage .

Eccezioni:

- Eccezione XMSException
- Eccezione IllegalState(la sessione è chiusa)

CreateConsumer - Crea consumer

Interfaccia:

```
IMessageConsumer CreateConsumer(IDestination dest) ;
```

Creare un utente del messaggio per la destinazione specificata

Parametri:

dest (input)
L'oggetto Destinazione.

Restituisce:
L'oggetto MessageConsumer .

Eccezioni:

- Eccezione XMSException
- Eccezione InvalidDestination

CreateConsumer - Crea consumer (con selettore messaggi)

Interfaccia:

```
IMessageConsumer CreateConsumer(IDestination dest,
                                String selector) ;
```

Creare un consumatore di messaggi per la destinazione specificata utilizzando un selettore di messaggi.

Parametri:

dest (input)
L'oggetto Destinazione.

selettore (input)
Un oggetto stringa che incapsula un'espressione del selettore messaggi. Solo i messaggi con proprietà che corrispondono all'espressione del selettore messaggi vengono consegnati al consumatore di messaggi.

Un oggetto String null indica che non esiste alcun selettore di messaggi per il consumer del messaggio.

Restituisce:
L'oggetto MessageConsumer .

Eccezioni:

- Eccezione XMSException
- Eccezione InvalidDestination
- Eccezione InvalidSelector

CreateConsumer - Crea consumer (con selettore di messaggi e indicatore di messaggi locali)

Interfaccia:

```
IMessageConsumer CreateConsumer(IDestination dest,
                                String selector,
                                Boolean noLocal) ;
```

Creare un consumatore di messaggi per la destinazione specificata utilizzando un selettore di messaggi e, se la destinazione è un argomento, specificando se il consumatore di messaggi riceve i messaggi pubblicati dalla propria connessione.

Parametri:

dest (input)
L'oggetto Destinazione.

selettore (input)
Un oggetto stringa che incapsula un'espressione del selettore messaggi. Solo i messaggi con proprietà che corrispondono all'espressione del selettore messaggi vengono consegnati al consumatore di messaggi.

Un oggetto String null indica che non esiste alcun selettore di messaggi per il consumer del messaggio.

noLocal (input)
Il valore True indica che l'utente del messaggio non riceve i messaggi pubblicati dalla propria connessione. Il valore False indica che il destinatario del messaggio riceve i messaggi pubblicati dalla propria connessione. Il valore predefinito è False.

Restituisce:
L'oggetto MessageConsumer .

Eccezioni:

- Eccezione XMSException
- Eccezione InvalidDestination
- Eccezione InvalidSelector

CreateDurableSottoscrittore - Crea sottoscrittore durevole

Interfaccia:

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,
                                         String subscription) ;
```

Creare un sottoscrittore durevole per l'argomento specificato.

Questo metodo non è valido per una connessione in tempo reale a un broker.

Per ulteriori informazioni sui sottoscrittori durevoli, consultare Sottoscrittori durevoli.

Parametri:

dest (input)
Un oggetto Destinazione che rappresenta l'argomento. L'argomento non deve essere un argomento temporaneo.

sottoscrizione (input)
Un oggetto stringa che incapsula un nome che identifica la sottoscrizione durevole. Il nome deve essere univoco all'interno dell'identificativo client per la connessione.

Restituisce:
L'oggetto MessageConsumer che rappresenta il sottoscrittore durevole.

Eccezioni:

- Eccezione XMSException
- Eccezione InvalidDestination

CreateDurableSottoscrittore - Crea sottoscrittore durevole (con selettore di messaggi e indicatore di messaggio locale)

Interfaccia:

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,
                                         String subscription,
                                         String selector,
                                         Boolean noLocal) ;
```

Creare un sottoscrittore (subscriber) durevole per l'argomento specificato utilizzando un selettore messaggi e specificando se il sottoscrittore (subscriber) durevole riceve i messaggi pubblicati dalla propria connessione.

Questo metodo non è valido per una connessione in tempo reale a un broker.

Per ulteriori informazioni sui sottoscrittori durevoli, consultare Sottoscrittori durevoli.

Parametri:

dest (input)
Un oggetto Destinazione che rappresenta l'argomento. L'argomento non deve essere un argomento temporaneo.

sottoscrizione (input)
Un oggetto stringa che incapsula un nome che identifica la sottoscrizione durevole. Il nome deve essere univoco all'interno dell'identificativo client per la connessione.

selettore (input)
Un oggetto stringa che incapsula un'espressione del selettore messaggi. Solo i messaggi con proprietà che corrispondono all'espressione del selettore messaggi vengono consegnati al sottoscrittore durevole.

Un oggetto String null indica che non esiste alcun selettore di messaggi per il sottoscrittore durevole.

noLocal (input)
Il valore True indica che il sottoscrittore durevole non riceve i messaggi pubblicati dalla propria connessione. Il valore False indica che il sottoscrittore durevole riceve i messaggi pubblicati dalla propria connessione. Il valore predefinito è False.

Restituisce:
L'oggetto MessageConsumer che rappresenta il sottoscrittore durevole.

Eccezioni:

- Eccezione XMSException
- Eccezione InvalidDestination
- Eccezione InvalidSelector

Messaggio CreateMap- Crea messaggio mappa

Interfaccia:

```
IMapMessage CreateMapMessage();
```

Creare un messaggio di associazione.

Parametri:
Nessuna

Restituisce:
L'oggetto MapMessage .

Eccezioni:

- Eccezione XMSException
- Eccezione IllegalState(la sessione è chiusa)

CreateMessage - Crea messaggio

Interfaccia:

```
IMessage CreateMessage();
```

Creare un messaggio senza corpo.

Parametri:
Nessuna

Restituisce:
L'oggetto Messaggio.

Eccezioni:

- Eccezione XMSException
- Eccezione IllegalState(la sessione è chiusa)

Messaggio CreateObject- Crea messaggio oggetto

Interfaccia:

```
IObjectMessage CreateObjectMessage();
```

Creare un messaggio oggetto.

Parametri:
Nessuna

Restituisce:
L'oggetto ObjectMessage .

Eccezioni:

- Eccezione XMSException
- Eccezione IllegalState(la sessione è chiusa)

CreateProducer - Crea Producer

Interfaccia:

```
IMessageProducer CreateProducer(IDestination dest) ;
```

Creare un produttore di messaggi per inviare i messaggi alla destinazione specificata

Parametri:

dest (input)
L'oggetto Destinazione.

Se si specifica un oggetto Destinazione null, il produttore del messaggio viene creato senza una destinazione. In questo caso, l'applicazione deve specificare una destinazione ogni volta che utilizza il mittente del messaggio per inviare un messaggio.

Restituisce:
L'oggetto MessageProducer .

Eccezioni:

- Eccezione XMSException
- Eccezione InvalidDestination

CreateQueue - Crea coda

Interfaccia:

```
IDestination CreateQueue(String queue) ;
```

Creare un oggetto Destinazione per rappresentare una coda nel server di messaggistica.

Questo metodo non crea la coda nel server di messaggistica. È necessario creare la coda prima che un'applicazione possa richiamare questo metodo.

Parametri:

coda (input)
Un oggetto stringa che incapsula il nome della coda o un URI (uniform resource identifier) che identifica la coda.

Restituisce:
L'oggetto Destinazione che rappresenta la coda.

Eccezioni:

- Eccezione XMSException

Messaggio CreateStream- Crea messaggio di flusso

Interfaccia:

```
IStreamMessage CreateStreamMessage();
```

Creare un messaggio di flusso.

Parametri:
Nessuna

Restituisce:
L'oggetto StreamMessage .

Eccezioni:

- Eccezione XMSException
- XMS_ILLEGAL_STATE_EXCEPTION

Coda CreateTemporary- Crea coda temporanea

Interfaccia:

```
IDestination CreateTemporaryQueue() ;
```

Creare una coda temporanea

L'ambito della coda temporanea è la connessione. Solo le sessioni create dalla connessione possono utilizzare la coda temporanea.

La coda temporanea rimane fino a quando non viene esplicitamente eliminata o fino al termine della connessione, a seconda di quale sia la prima.

Per ulteriori informazioni sulle code temporanee, consultare Destinazioni temporanee.

Parametri:
Nessuna

Restituisce:
L'oggetto Destinazione che rappresenta la coda temporanea.

Eccezioni:

- Eccezione XMSException

Argomento CreateTemporary- Crea argomento temporaneo

Interfaccia:

```
IDestination CreateTemporaryTopic() ;
```

Creare un argomento temporaneo.

L'ambito dell'argomento temporaneo è la connessione. Solo le sessioni create dalla connessione possono utilizzare l'argomento temporaneo.

L'argomento temporaneo rimane fino a quando non viene eliminato esplicitamente o fino al termine della connessione, a seconda di quale sia la data più breve.

Per ulteriori informazioni sugli argomenti temporanei, consultare Destinazioni temporanee.

Parametri:
Nessuna

Restituisce:
L'oggetto Destinazione che rappresenta l'argomento temporaneo.

Eccezioni:

- Eccezione XMSException

Messaggio CreateText- Crea messaggio di testo

Interfaccia:

```
ITextMessage CreateTextMessage();
```

Creare un messaggio di testo con un corpo vuoto.

Parametri:
Nessuna

Restituisce:
L'oggetto TextMessage .

Eccezioni:

- Eccezione XMSException

CreateTextMessaggio - Crea messaggio di testo (inizializzato)

Interfaccia:

```
ITextMessage CreateTextMessage(String initialValue);
```

Creare un messaggio di testo il cui corpo sia inizializzato con il testo specificato.

Parametri:

initialValue (input)
Un oggetto String che incapsula il testo per inizializzare il contenuto del messaggio di testo.

Nessuna

Restituisce:
L'oggetto TextMessage .

Eccezioni:

- Eccezione XMSException

CreateTopic - Crea argomento

Interfaccia:

```
IDestination CreateTopic(String topic) ;
```

Creare un oggetto Destinazione per rappresentare un argomento.

Parametri:

argomento (input)
Un oggetto Stringa che incapsula il nome dell'argomento o un URI (uniform resource identifier) che identifica l'argomento.

Restituisce:
L'oggetto Destinazione che rappresenta l'argomento.

Eccezioni:

- Eccezione XMSException

Recupera - Ripristina

Interfaccia:

```
void Recover();
```

Ripristinare la sessione La consegna del messaggio viene arrestata e riavviata con il messaggio non riconosciuto più vecchio.

La sessione non deve essere una sessione con transazioni.

Per ulteriori informazioni sul ripristino di una sessione, consultare Conferma messaggio.

Parametri:
Nessuna

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- Eccezione IllegalState

Rollback - Rollback

Interfaccia:

```
void Rollback();
```

Eseguire il rollback di tutti i messaggi elaborati nella transazione corrente

La sessione deve essere una sessione transattiva.

Parametri:
Nessuna

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- Eccezione IllegalState

Annulla sottoscrizione - Annulla sottoscrizione

Interfaccia:

```
void Unsubscribe(String subscription);
```

Eliminare una sottoscrizione durevole. Il server di messaggistica elimina il record della sottoscrizione durevole che sta gestendo e non invia ulteriori messaggi al sottoscrittore durevole.

Un'applicazione non può eliminare una sottoscrizione durevole in nessuna delle seguenti circostanze:

- Mentre è presente un consumer di messaggi attivo per la sottoscrizione durevole
- Mentre un messaggio utilizzato fa parte di una transazione in sospeso
- Mentre un messaggio utilizzato non è stato riconosciuto

Questo metodo non è valido per una connessione in tempo reale a un broker.

Parametri:

sottoscrizione (input)
Un oggetto stringa che incapsula il nome che identifica la sottoscrizione durevole.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- Eccezione InvalidDestination
- Eccezione IllegalState

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IStreamMessage

Un messaggio di flusso è un messaggio il cui contenuto comprende un flusso di valori, dove ogni valore ha un tipo di dati associato. Il contenuto del corpo viene scritto e letto in modo sequenziale.

Gerarchia di eredità:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IStreamMessage
```

Quando un'applicazione legge un valore dal flusso di messaggi, il valore può essere convertito da XMS in un altro tipo di dati. Per ulteriori informazioni su questa forma di conversione implicita, consultare Il corpo di un messaggio XMS.

Metodi

ReadBoolean - Leggi valore booleano

Interfaccia:

```
Boolean ReadBoolean();
```

Leggere un valore booleano dal flusso di messaggi.

Parametri:
Nessuna

Restituisce:
Il valore booleano letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadByte - Byte di lettura

Interfaccia:

```
Int16   ReadSignedByte();
    Byte    ReadByte();
```

Leggere un numero intero a 8 bit dal flusso di messaggi.

Parametri:
Nessuna

Restituisce:
Il byte letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadBytes - Byte letti

Interfaccia:

```
Int32   ReadBytes(Byte[] array);
```

Leggere un array di byte dal flusso di messaggi.

Parametri:

array (immissione)
Il buffer contenente l'array di byte letti e la lunghezza del buffer in byte.

Se il numero di byte nella schiera è inferiore o uguale alla lunghezza del buffer, l'intera schiera viene letta nel buffer. Se il numero di byte nella schiera è maggiore della lunghezza del buffer, il buffer viene riempito con parte della schiera e un cursore interno contrassegna la posizione del byte successivo da leggere. Una chiamata successiva a readBytes() legge i byte dall'array iniziando dalla posizione corrente del cursore.

Se si specifica un puntatore null sull'input, la chiamata salta l'array di byte senza leggerlo.

Restituisce:
Il numero di byte letti nel buffer. Se il buffer è parzialmente riempito, il valore è inferiore alla lunghezza del buffer, indicando che non ci sono più byte nell'array rimanenti da leggere. Se non ci sono byte rimanenti da leggere dall'array prima della chiamata, il valore è `XMSC_END_OF_BYTEARRAY`.

Se si specifica un puntatore nullo sull'immissione, il metodo non restituisce alcun valore.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadChar - Carattere di lettura

Interfaccia:

```
Char    ReadChar();
```

Leggere un carattere a 2 byte dal flusso di messaggi.

Parametri:
Nessuna

Restituisce:
Il carattere che viene letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadDouble - Leggi numero a virgola mobile di precisione doppia

Interfaccia:

```
Double  ReadDouble();
```

Leggere un numero a virgola mobile a doppia precisione a 8 byte dal flusso di messaggi.

Parametri:
Nessuna

Restituisce:
Il numero a virgola mobile di precisione doppia letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadFloat - Leggi numero a virgola mobile

Interfaccia:

```
Single  ReadFloat();
```

Leggere un numero a virgola mobile a 4 byte dal flusso di messaggi.

Parametri:
Nessuna

Restituisce:
Il numero a virgola mobile letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadInt - Lettura numero intero

Interfaccia:

```
Int32   ReadInt();
```

Leggere un intero a 32 bit con segno dal flusso di messaggi.

Parametri:
Nessuna

Restituisce:
Il numero intero letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadLong - Lettura numero intero lungo

Interfaccia:

```
Int64   ReadLong();
```

Leggere un intero a 64 bit con segno dal flusso di messaggi.

Parametri:
Nessuna

Restituisce:
Il numero intero lungo letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadObject - Lettura oggetto

Interfaccia:

```
Object  ReadObject();
```

Leggere un valore dal flusso di messaggi e restituirne il tipo di dati.

Parametri:
Nessuna

Restituisce:
Il valore, che è uno dei seguenti tipi di oggetto:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

Eccezioni:
Eccezione XMSException

ReadShort - Lettura numero intero breve

Interfaccia:

```
Int16   ReadShort();
```

Leggere un numero intero a 16 bit dal flusso di messaggi.

Parametri:
Nessuna

Restituisce:
Il numero intero breve letto.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

ReadString - Lettura stringa

Interfaccia:

```
String  ReadString();
```

Leggere una stringa dal flusso di messaggi. Se richiesto, XMS converte i caratteri nella stringa nella codepage locale.

Parametri:
Nessuna

Restituisce:
Un oggetto stringa che incapsula la stringa letta. Se è richiesta la conversione dei dati, questa è la stringa dopo la conversione.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

Reimposta - Reimposta

Interfaccia:

```
void  Reset();
```

Inserire il corpo del messaggio in modalità di sola lettura e riposizionare il cursore all'inizio del flusso di messaggi.

Parametri:
Nessuna

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageEOFException

WriteBoolean - Scrivi valore booleano

Interfaccia:

```
void  WriteBoolean(Boolean value);
```

Scrivere un valore booleano nel flusso di messaggi.

Parametri:

valore (immissione)
Il valore booleano da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteByte - Byte di scrittura

Interfaccia:

```
void  WriteByte(Byte value);
void  WriteSignedByte(Int16 value);
```

Scrivere un byte nel flusso di messaggi.

Parametri:

valore (immissione)
Il byte da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteBytes - Byte scritti

Interfaccia:

```
void  WriteBytes(Byte[] value);
```

Scrivere un array di byte nel flusso di messaggi

Parametri:

valore (immissione)
La schiera di byte da scrivere.

lunghezza (input)
Il numero di byte nell'array.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteChar - Scrivi carattere

Interfaccia:

```
void  WriteChar(Char value);
```

Scrivere un carattere nel flusso di messaggi come 2 byte, prima byte di ordine superiore.

Parametri:

valore (immissione)
Il carattere da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteDouble - Scrivere il numero a virgola mobile di precisione doppia

Interfaccia:

```
void  WriteDouble(Double value);
```

Convertire un numero a virgola mobile di precisione doppia in numero intero lungo e scrivere il numero intero lungo nel flusso di messaggi come byte 8, byte di ordine superiore per primo.

Parametri:

valore (immissione)
Il numero a virgola mobile di precisione doppia da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteFloat - Numero virgola mobile di scrittura

Interfaccia:

```
void  WriteFloat(Single value);
```

Converti un numero a virgola mobile in un numero intero e scrivi il numero intero nel flusso di messaggi come 4 byte, primo byte di ordine elevato.

Parametri:

valore (immissione)
Il numero a virgola mobile da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteInt - Scrittura numero intero

Interfaccia:

```
void  WriteInt(Int32 value);
```

Scrivere un numero intero nel flusso di messaggi come 4 byte, prima byte di ordine superiore.

Parametri:

valore (immissione)
Il numero intero da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteLong - Scrittura numero intero lungo

Interfaccia:

```
void  WriteLong(Int64 value);
```

Scrivere un numero intero lungo nel flusso di messaggi come 8 byte, prima byte di ordine superiore.

Parametri:

valore (immissione)
Il numero intero lungo da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteObject - Scrivi oggetto

Interfaccia:

```
void  WriteObject(Object value);
```

Scrivere un valore, con un tipo di dati specificato, nel flusso di messaggi.

Parametri:

objectType (input)
Il valore, che deve essere uno dei seguenti tipi di oggetto:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

valore (immissione)
Un array di byte contenente il valore da scrivere.

lunghezza (input)
Il numero di byte nell'array.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException

WriteShort - Scrivi numero intero breve

Interfaccia:

```
void  WriteShort(Int16 value);
```

Scrivere un numero intero breve nel flusso di messaggi come 2 byte, prima byte di ordine superiore.

Parametri:

valore (immissione)
Il numero intero breve da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

WriteString - Scrivi stringa

Interfaccia:

```
void  WriteString(String value);
```

Scrivere una stringa nel flusso di messaggi

Parametri:

valore (immissione)
Un oggetto String che incapsula la stringa da scrivere.

Restituisce:
Azzera

Eccezioni:

- Eccezione XMSException
- MessageNotWritableException

Proprietà e metodi ereditati

Le seguenti proprietà sono ereditate dall'interfaccia IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedeliveryMode, JMSReplyTo, JMSTimestamp, JMSType, Proprietà

I seguenti metodi sono ereditati dall'interfaccia IMessage:

clearBody, clearProperties, PropertyExists

I seguenti metodi sono ereditati dall'interfaccia IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

ITextMessage

Un messaggio di testo è un messaggio il cui contenuto comprende una stringa.

Gerarchia di eredità:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.ITextMessage
```

Proprietà di .NET

Testo - Richiama e imposta testo

Interfaccia:

```
String Text
{
  get;
  set;
}
```

Richiamare e impostare la stringa che costituisce il corpo del messaggio di testo.

Se richiesto, XMS converte i caratteri nella stringa nella codepage locale.

Eccezioni:

- Eccezione XMSException
- MessageNotReadableException
- MessageNotWritableException
- MessageEOFException

Proprietà e metodi ereditati

Le seguenti proprietà sono ereditate dall'interfaccia IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedeliveryMode, JMSReplyTo, JMSTimestamp, JMSType, Proprietà

I seguenti metodi sono ereditati dall'interfaccia IMessage:

clearBody, clearProperties, PropertyExists

I seguenti metodi sono ereditati dall'interfaccia IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

TransactionInProgressException

XMS genera questa eccezione se un'applicazione richiede un'operazione non valida perché è in corso una transazione.

Gerarchia di eredità:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.TransactionInProgressException
```

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia XMSException:

CodiceGetError, GetLinkedEccezione

TransactionRolledBackException

XMS genera questa eccezione se un'applicazione richiama Session.commit() per eseguire il commit della transazione corrente, ma viene eseguito il rollback della transazione.

Gerarchia di eredità:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.TransactionRolledBackException
```

Proprietà e metodi ereditati

I seguenti metodi sono ereditati dall'interfaccia XMSException:

CodiceGetError, GetLinkedEccezione

Eccezione XMSException

Se XMS rileva un errore durante l'elaborazione di una chiamata a un metodo .NET , XMS genera un'eccezione. Un'eccezione è un oggetto che contiene informazioni sull'errore.

Gerarchia di eredità:

```
System.Exception
   |
   +----IBM.XMS.XMSException
```

Esistono diversi tipi di eccezione XMS e un oggetto XMSException è solo un tipo di eccezione. Tuttavia, la classe XMSException è una superclasse delle altre classi di eccezione XMS . XMS genera un oggetto XMSException in situazioni in cui nessuno degli altri tipi di eccezione è appropriato.

Proprietà di .NET

ErrorCode - Richiama codice di errore

Interfaccia:

```
public String ErrorCode
{
  get {return errorCode_;}
}
```

Ottieni il codice di errore.

Eccezioni:

- Eccezione XMSException

LinkedException - Acquisisci eccezione collegata

Interfaccia:

```
public Exception LinkedException
{
  get { return linkedException_;}
  set { linkedException_ = value;}
}
```

Richiama l'eccezione successiva nella catena di eccezioni

Il metodo restituisce un valore null se non ci sono più eccezioni nella catena.

Eccezioni:

- Eccezione XMSException

XMSFactoryFactory

Se un'applicazione non utilizza oggetti gestiti, utilizzare questa classe per creare factory di connessione, code e argomenti.

Gerarchia di eredità:
Nessuna

Proprietà di .NET

Metadati - Richiama metadati

Interfaccia:

```
IConnectionMetaData MetaData
```

Richiamare i metadati appropriati per il tipo di connessione dell'oggetto XMSFactoryFactory .

Eccezioni:
Nessuna

Metodi

CreateConnectionFactory - Crea factory di connessione

Interfaccia:

```
IConnectionFactory CreateConnectionFactory();
```

Creare un oggetto ConnectionFactory del tipo dichiarato.

Parametri:
Nessuna

Restituisce:
L'oggetto ConnectionFactory .

Eccezioni:

- Eccezione XMSException

CreateQueue - Crea coda

Interfaccia:

```
IDestination CreateQueue(String name);
```

Creare un oggetto Destinazione per rappresentare una coda nel server di messaggistica.

Questo metodo non crea la coda nel server di messaggistica. È necessario creare la coda prima che un'applicazione possa richiamare questo metodo.

Parametri:

nome (input)
Un oggetto stringa che incapsula il nome della coda o un URI (uniform resource identifier) che identifica la coda.

Restituisce:
L'oggetto Destinazione che rappresenta la coda.

Eccezioni:

- Eccezione XMSException

CreateTopic - Crea argomento

Interfaccia:

```
IDestination CreateTopic(String name);
```

Creare un oggetto Destinazione per rappresentare un argomento.

Parametri:

nome (input)
Un oggetto Stringa che incapsula il nome dell'argomento o un URI (uniform resource identifier) che identifica l'argomento.

Restituisce:
L'oggetto Destinazione che rappresenta l'argomento.

Eccezioni:

- Eccezione XMSException

GetInstance - Richiamare un'istanza di XMSFactoryFactory

Interfaccia:

```
static XMSFactoryFactory GetInstance(int connectionType);
```

Creare un'istanza di XMSFactoryFactory. Un'applicazione XMS utilizza un oggetto XMSFactoryFactory per ottenere un riferimento a un oggetto ConnectionFactory appropriato per il tipo di protocollo richiesto. Questo oggetto ConnectionFactory può quindi produrre connessioni solo per quel tipo di protocollo.

Parametri:

connectionType (input)
Il tipo di connessione per cui l'oggetto ConnectionFactory produce connessioni:

- XMSC.CT_WPM
- XMSC.CT_RTT
- XMSC.CT_WMQ

Restituisce:
L'oggetto XMSFactoryFactory dedicato al tipo di connessione dichiarato.

Eccezioni:

- Eccezione NotSupported

Proprietà degli oggetti XMS

Questa sezione documenta le proprietà dell'oggetto definite da XMS.

Questa sezione contiene informazioni sui seguenti tipi di oggetto:

- “Proprietà della connessione” a pagina 2085
- “Proprietà di ConnectionFactory” a pagina 2086
- “Proprietà dei dati ConnectionMeta” a pagina 2090
- “Proprietà della destinazione” a pagina 2091
- “Proprietà di InitialContext” a pagina 2092
- “Proprietà del messaggio” a pagina 2093

- “Proprietà di MessageConsumer” a pagina 2098
- “Proprietà di MessageProducer” a pagina 2098
- “Proprietà della sessione” a pagina 2098

La descrizione di ciascun tipo di oggetto elenca le proprietà di un oggetto del tipo specificato e fornisce una breve descrizione di ciascuna proprietà.

Questa sezione fornisce anche una definizione di ciascuna proprietà (consultare “Definizioni proprietà” a pagina 2099).

Se un'applicazione definisce le proprie proprietà degli oggetti descritti in questa sezione, non causa un errore, ma potrebbe causare risultati imprevedibili.

Nota: I nomi e i valori delle proprietà in questa sezione sono mostrati nel formato XMSC.*NAME*, che è il modulo utilizzato per C e C + +. Tuttavia, in .NET, il formato del nome della proprietà può essere XMSC.*NAME* o XMSC_*NAME*, a seconda di come viene utilizzato:

- Se si sta specificando una proprietà, il nome della proprietà deve essere nel formato XMSC.*NAME* come mostrato nel seguente esempio:

  ```
  cf.SetStringProperty(XMSC.WMQ_CHANNEL, "DOTNET.SVRCONN");
  ```

- Se si sta specificando una stringa, il nome della proprietà deve essere nel formato XMSC_*NAME* come mostrato nel seguente esempio:

  ```
  cf.SetStringProperty("XMSC_WMQ_CHANNEL", "DOTNET.SVRCONN");
  ```

  In .NET, i nomi e i valori delle proprietà vengono forniti come costanti nella classe XMSC. Queste costanti identificano le stringhe e vengono utilizzate da qualsiasi applicazione XMS .NET . Se si utilizzano queste costanti predefinite, i nomi e i valori delle proprietà sono nel formato XMSC.*NOME*, quindi, ad esempio, si utilizza XMSC.USERID, piuttosto che XMSC_USERID.

I tipi di dati sono anche nel formato utilizzato per C/C + +. È possibile trovare i corrispondenti valori per .NET in Tipi di dati per .NET.

Proprietà della connessione

Una panoramica delle proprietà dell'oggetto Connection, con collegamenti a informazioni di riferimento più dettagliate.

Tabella 872. Proprietà della connessione

Nome della proprietà	Descrizione
“XMSC_WMQ_RESOLVED_QUEUE_MANAGER” a pagina 2133	Questa proprietà viene utilizzata per ottenere il nome del gestore code a cui è connesso.
“ID_MANAGER_XMSC_WMQ_RESOLVED_QUEUE_” a pagina 2133	Questa proprietà viene popolata con l'ID del gestore code dopo la connessione.
XMSC_WPM_CONNECTION_PROTOCOL	Il protocollo di comunicazione utilizzato per la connessione al motore di messaggistica. Questa proprietà è di sola lettura.
Nome_HOST_WPM_XMSC	Il nome host o l'indirizzo IP del sistema che contiene il motore di messaggistica a cui è connessa l'applicazione. Questa proprietà è di sola lettura.
NOME_ME_WPM XMSC	Il nome del motore di messaggistica a cui è connessa l'applicazione. Questa proprietà è di sola lettura.
XMSC_WPM_PORT	Il numero della porta di ascolto da parte del motore di messaggistica a cui è connessa l'applicazione. Questa proprietà è di sola lettura.

Un oggetto Connection ha anche proprietà di sola lettura derivate dalle proprietà del factory di connessione utilizzato per creare la connessione. Queste proprietà derivano non solo dalle proprietà del factory di connessione impostate al momento della creazione della connessione, ma anche dai valori predefiniti delle proprietà non impostate. Le proprietà includono solo quelle rilevanti per il tipo di server di messaggistica a cui è connessa l'applicazione. I nomi delle proprietà sono uguali ai nomi delle proprietà del factory di connessione.

Proprietà di ConnectionFactory

Una panoramica delle proprietà dell'oggetto ConnectionFactory , con link a informazioni di riferimento più dettagliate.

Tabella 873. Proprietà di ConnectionFactory

Nome della proprietà	**Descrizione**
“XMSC_ASYNC_EXCEPTIONS” a pagina 2108	Questa proprietà determina se XMS informa un ExceptionListener solo quando una connessione viene interrotta o quando si verifica qualsiasi eccezione in modo asincrono per una chiamata API XMS. Questa proprietà si applica a tutte le connessioni create da questa ConnectionFactory che hanno un ExceptionListener registrato.
“TIPO_APPLICAZIONE XMSC_WMQ_BALANCING_” a pagina 2117	Tipo di opzione di bilanciamento
“OPZIONI_BILANCIAMENTE_WMQ_XMSC_” a pagina 2117	Opzioni di bilanciamento impostate dall'applicazione emittente
“TIMEOUT_BILANCIAMENTO_WMQ_XMSC_” a pagina 2118	Timeout dopo il quale il ribilanciamento potrebbe interrompere l'attività dell'applicazione.
ID_CLI_XMSC	L'identificativo client per una connessione.
XMSC_CONNECTION_TYPE	Il tipo di server di messaggistica a cui si connette un'applicazione.
PASSWORD_XMSC_	Una password che può essere utilizzata per autenticare l'applicazione quando prova a connettersi a un server di messaggistica.
“XMSC_RTT_BROKER_PING_INTERVAL” a pagina 2114	L'intervallo di tempo, in millisecondi, dopo il quale XMS .NET controlla la connessione a un server di messaggistica in tempo reale per rilevare eventuale attività.
XMSC_RTT_CONNECTION_PROTOCOL	Il protocollo di comunicazione utilizzato per una connessione in tempo reale a un broker.
XMSC_RTT_NOME_HOST	Il nome host o l'indirizzo IP del sistema su cui viene eseguito un broker.
XMSC_RTT_LOCAL_ADDRESS	Il nome host o l'indirizzo IP dell'interfaccia di rete locale da utilizzare per una connessione in tempo reale a un broker.
XMSC_RTT_MULTICAST	L'impostazione multicast per una destinazione o una factory di connessione.
PORTA_RTT_XMSC	Il numero della porta su cui un broker è in ascolto per le richieste in entrata.

Tabella 873. Proprietà di ConnectionFactory (Continua)	
Nome della proprietà	**Descrizione**
IDUSER_XMSC	Un identificativo utente che può essere utilizzato per autenticare l'applicazione quando prova a connettersi a un server di messaggistica.
XMSC_WMQ_BROKER_CONTROLQ	Il nome della coda di controllo utilizzata da un Broker.
XMSC_WMQ_BROKER_PUBQ	Il nome della coda monitorata da un broker dove le applicazioni inviano i messaggi che pubblicano.
XMSC_WMQ_BROKER_QMGR	Il nome del gestore code a cui è connesso un broker.
XMSC_WMQ_BROKER_SUBQ	Il nome della coda sottoscrittore per un utilizzatore di messaggi non durevole.
XMSC_WMQ_BROKER_VERSION	Il tipo di broker utilizzato dall'applicazione per una connessione o per la destinazione.
“URL CCDT WMQ_XMSC” a pagina 2120	Un URL (A Uniform Resource Locator) che identifica il nome e l'ubicazione del file che contiene la tabella di definizione del canale client e che specifica anche come è possibile accedere al file.
XMSC_WMQ_CHALLEGATO	Il nome del canale da utilizzare per una connessione.
“XMSC_WMQ_CLIENT_RECONNECT_OPTIONS” a pagina 2121	Questa proprietà specifica le opzioni di riconnessione client per le nuove connessioni create da questo factory
“XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT” a pagina 2121	Questa proprietà specifica il periodo di tempo, in secondi, in cui una connessione client tenta di riconnettersi.
XMSC_WMQ_CONNECTION_MODE	La modalità in base alla quale un'applicazione si connette a un gestore code.
“ELENCO_NOMI_CONNESSIONI_WMQ_XMSC_” a pagina 2122	Questa proprietà specifica gli host a cui il client tenta di riconnettersi dopo che la connessione è stata interrotta.
XMSC_WMQ_FAIL_IF_QUIESCE	Indicazione della mancata riuscita delle chiamate a specifici metodi se il gestore code a cui è connessa l'applicazione è in uno stato di non attivo.
NOME_HOST WMQ_XMSC	Il nome host o l'indirizzo IP del sistema su cui viene eseguito un gestore code.
XMSC_WMQ_LOCAL_ADDRESS	Per una connessione a un gestore code, questa proprietà specifica l'interfaccia di rete locale da utilizzare oppure la porta locale o l'intervallo di porte locali da utilizzare oppure entrambe le cose.
XMSC_WMQ_MESSAGE_SELECTION	Determina se la selezione del messaggio viene effettuata dal client XMS o dal broker.
XMSC_WMQ_MSG_BATCH_XX_ENCODE_CASE_ONE dimensione	Il numero massimo di messaggi da richiamare da una coda in un singolo batch quando si utilizza la consegna di messaggi asincrona.
XMSC_WMQ_POLLING_INTERVAL	Se ciascun listener messaggi in una sessione non dispone di alcun messaggio adatto nella propria coda, questo valore è l'intervallo massimo, in millisecondi, che trascorre prima che ciascun listener messaggi provi di nuovo ad ottenere un messaggio dalla propria coda.

Tabella 873. Proprietà di ConnectionFactory (Continua)	
Nome della proprietà	**Descrizione**
“XMSC_WMQ_PROVIDER_VERSION” a pagina 2131	La versione, la release, il livello di modifica e il fix pack del gestore code al quale l'applicazione intende connettersi.
XMSC_WMQ_PORT	Il numero della porta su cui un gestore code è in ascolto per le richieste in entrata.
XMSC_WMQ_PUB_ACK_INTERVAL	Il numero di messaggi pubblicati da un publisher prima che il client XMS richieda un riconoscimento dal broker.
“XMSC_WMQ_PUT_ASYNC_ALLOWED” a pagina 2126	Questa proprietà determina se ai mittenti del messaggio è consentito utilizzare i put asincroni per inviare messaggi a questa destinazione.
XMSC_WMQ_QMGR_CCSID	L'identificativo (CCSID) della serie di caratteri codificati, o codepage, in cui i campi di dati carattere definiti in MQI (Message Queue Interface) vengono scambiati tra il client XMS e quello IBM MQ .
XMSC_WMQ_QUEUE_MANAGER	Il nome del gestore code a cui connettersi.
XMSC_WMQ_RECEIVE_EXIT	Identifica un'uscita di ricezione del canale da eseguire.
XMSC_WMQ_RECEIVE_EXIT_INIT	I dati utente passati a un'uscita di ricezione del canale quando viene richiamata.
XMSC_WMQ_SECURITY_EXIT	Identifica un'uscita di sicurezza del canale.
XMSC_WMQ_SECURITY_EXIT_INIT	I dati utente passati a un'uscita di sicurezza del canale quando viene richiamata.
“WMQ_XMSC_SEND_CHECK_COUNT” a pagina 2135	Il numero di chiamate di invio da consentire tra i controlli per rilevare eventuali errori di put asincrono all'interno di una singola sessione XML non sottoposta a transazione.
XMSC_WMQ_SEND_EXIT	Identifica un'uscita di invio del canale.
XMSC_WMQ_SEND_EXIT_INIT	I dati utente passati alle uscite di invio del canale quando vengono richiamate.
“XMSC_WMQ_SHARE_CONV_ALLOWED” a pagina 2135	Se una connessione client può condividere il proprio socket con altre connessioni XMS di livello superiore dallo stesso processo allo stesso gestore code, se le definizioni di canale corrispondono. Questa proprietà viene fornita per consentire un isolamento completo delle connessioni in socket separati se necessario per motivi correlati allo sviluppo, alla manutenzione e alla gestione delle applicazioni.
XMSC_WMQ_SSL_CERT_STORES	L'ubicazione dei server che contengono i CRL (cettificate revocation list) da utilizzare su una connessione SSL a un gestore code.
XMSC_WMQ_SSL_CIPHER_SPEC	Il nome della CipherSpec da utilizzare su una connessione protetta a un gestore code.
XMSC_WMQ_SSL_CIPHER_SUITE	Il nome della CipherSuite da utilizzare su una connessione TLS a un gestore code. Il protocollo utilizzato nella negoziazione della connessione protetta dipende dalla CipherSuite specificata.

Tabella 873. Proprietà di ConnectionFactory (Continua)	
Nome della proprietà	**Descrizione**
XMSC_WMQ_SSL_CRYPTO_HW	I dettagli di configurazione per l'hardware di crittografia connesso al sistema client.
XMSC_WMQ_SSL_FIPS_REQUIRED	Il valore della proprietà determina se un'applicazione può o meno utilizzare delle suite di crittografia non conformi a FIPS. Se questa proprietà è impostata su true, per la connessione client-server vengono utilizzati solo gli algoritmi FIPS.
XMSC_WMQ_SSL_KEY_REPOSITORY	L'ubicazione del file del database delle chiavi in cui sono memorizzati chiavi e certificati.
XMSC_WMQ_SSL_KEY_RESETCOUNT	Il KeyResetCount rappresenta il numero totale di byte non crittografati inviati e ricevuti in una conversazione SSL prima che venga rinegoziata la chiave segreta.
XMSC_WMQ_SSL_PEER_NAME	Il nome peer da utilizzare su una connessione SSL a un gestore code.
XMSC_WMQ_SYNCPOINT_ALL_GETS	Indica se tutti i messaggi devono essere richiamati dalle code entro il controllo del punto di sincronizzazione.
“XMSC_WMQ_TARGET_CLIENT” a pagina 2142	
XMSC_WMQ_TEMP_Q_PREFIX	Il prefisso utilizzato per formare il nome della coda dinamica IBM MQ creata quando l'applicazione crea una coda temporanea XMS .
XMSC_WMQ_TEMP_TOPIC_PREFIX	Quando si creano argomenti temporanei, XMS genera una stringa di argomenti nel formato "`TEMP/TEMPTOPICPREFIX/unique_id`" oppure, se questa proprietà contiene il valore predefinito, viene generata questa stringa, "`TEMP/unique_id`". La specifica di un valore non vuoto consente la definizione di specifiche code modello per la creazione delle code gestite per i sottoscrittori di argomenti temporanei creati in questa connessione.
XMSC_WMQ_TEMPORARY_MODEL	Il nome della coda modello IBM MQ da cui viene creata una coda dinamica quando l'applicazione crea una coda temporanea XMS .
XMSC_WPM_BUS_NAME	Per una factory di connessione, il nome del bus di integrazione del servizio a cui si connette l'applicazione oppure, per una destinazione, il nome del bus di integrazione del servizio in cui esiste la destinazione.
XMSC_WPM_CONNECTION_PROSSIMITÀ	L'impostazione di prossimità della connessione per la connessione.
XMSC_WPM_DUR_SUB_HOME	Il nome del motore di messaggistica dove vengono gestite tutte le sottoscrizioni durevoli per una connessione o una destinazione.
XMSC_WPM_LOCAL_ADDRESS	Per una connessione a un bus di integrazione del servizio, questa proprietà specifica l'interfaccia di rete locale da utilizzare oppure la porta locale o l'intervallo di porte locali da utilizzare, oppure entrambe le cose.

Tabella 873. Proprietà di ConnectionFactory (Continua)	
Nome della proprietà	**Descrizione**
XMSC_WPM_NON_PERSISTENT_MAP	Il livello di affidabilità dei messaggi non persistenti inviati utilizzando la connessione.
XMSC_WPM_PERSISTENT_MAP	Il livello di affidabilità dei messaggi persistenti inviati utilizzando la connessione.
XMSC_WPM_PROVIDER_ENDPOINTS	Una sequenza di uno o più indirizzi endpoint di server di avvio.
XMSC_WPM_XX_ENCODE_CASE_ONE gruppo_destinazione	Il nome del gruppo di destinazione dei motori di messaggistica.
XMSC_WPM_TARGET_SIGNIFICATIVITÀ	La significatività del gruppo di destinazione dei motori di messaggistica.
XMSC_WPM_TARGET_TRANSPORT_CHAIN	Il nome della catena di trasporto in entrata che l'applicazione deve utilizzare per connettersi a un motore di messaggistica.
XMSC_WPM_XX_ENCODE_CASE_ONE tipo_destinazione	Il tipo di gruppo di destinazione dei motori di messaggistica.
XMSC_WPM_TEMP_Q_PREFIX	Il prefisso utilizzato per formare il nome della coda temporanea creata nel SIB (service integration bus) quando l'applicazione crea una coda temporanea XMS .
XMSC_WPM_TEMP_TOPIC_PREFIX	Il prefisso utilizzato per formare il nome di un argomento temporaneo creato dall'applicazione.

Proprietà dei dati ConnectionMeta

Una panoramica delle proprietà dell'oggetto dati ConnectionMeta, con link a informazioni di riferimento più dettagliate.

Tabella 874. Proprietà dei dati ConnectionMeta	
Nome della proprietà	**Descrizione**
XMSC_JMS_MAJOR_VERSION	Il numero di versione principale della specifica JMS su cui si basa XMS . Questa proprietà è di sola lettura.
XMSC_JMS_MINOR_VERSION	Il numero di versione minore della specifica JMS su cui si basa XMS . Questa proprietà è di sola lettura.
VERSIONE JMS XMSC	L'identificativo della versione della specifica JMS su cui si basa XMS . Questa proprietà è di sola lettura.
VERSIONE_MAGGIORE_XMSCR	Il numero di versione del client XMS . Questa proprietà è di sola lettura.
XMSC_VERSIONE_MINORE	Il numero di release del client XMS . Questa proprietà è di sola lettura.
NOME_PROVIDER_XMSCD	Il fornitore del client XMS . Questa proprietà è di sola lettura.
VERSIONE XMSC	L'identificativo della versione della cliXMSent. Questa proprietà è di sola lettura.

Proprietà della destinazione

Una panoramica delle proprietà dell'oggetto Destinazione, con link a informazioni di riferimento più dettagliate.

Tabella 875. Proprietà della destinazione

Nome della proprietà	Descrizione
MODALITÀ_DISTRIBUZIONE_XMSCD	La modalità di consegna del messaggio inviato alla destinazione.
PRIORITÀ_XMSC_XX_ENCODE_CASE_CAPS_LOCK_OFF	La priorità dei messaggi inviati alla destinazione.
XMSC_RTT_MULTICAST	L'impostazione multicast per una destinazione o una factory di connessione.
XMSC_TIME_TO_LIVE	La durata (TTL, Time To Live) per i messaggi inviati alla destinazione.
XMSC_WMQ_BROKER_VERSION	Il tipo di broker utilizzato dall'applicazione per una connessione o per la destinazione.
XMSC_WMQ_CCSID	L'identificativo (CCSID) della serie di caratteri codificati, o codepage, in cui si trovano le stringhe di dati carattere nel corpo di un messaggio quando il client XMS inoltra il messaggio alla destinazione.
XMSC_WMQ_DUR_SUBQ	il nome della coda sottoscrittore per un sottoscrittore durevole che sta ricevendo messaggi dalla destinazione. **Nota:** Questa proprietà può essere utilizzata con la versione 2.0 di IBM Message Service Client per .NET , ma non ha alcun effetto per un'applicazione connessa a un gestore code IBM WebSphere MQ 7.0 a meno che la proprietà XMSC_WMQ_PROVIDER_VERSION della factory di connessione non sia impostata su un numero di versione inferiore a 7.
XMSC_WMQ_ENCODING	Modalità di rappresentazione dei dati numerici nel corpo di un messaggio quando il client XMS inoltra il messaggio alla destinazione.
XMSC_WMQ_FAIL_IF_QUIESCE	Indicazione della mancata riuscita delle chiamate a specifici metodi se il gestore code a cui è connessa l'applicazione è in uno stato di non attivo.
“CORPO messaggio wmq_xmsc_” a pagina 2124	Questa proprietà determina se un'applicazione XMS elabora MQRFH2 di un messaggio IBM MQ come parte del payload del messaggio (ovvero, come parte del contenuto del messaggio).
“XMSC_WMQ_MQMD_MESSAGE_CONTEXT” a pagina 2125	Determina quale livello di contesto del messaggio deve essere impostato dall'applicazione XMS . L'applicazione deve essere in esecuzione con l'autorizzazione di contesto appropriata perché questa proprietà diventi effettiva.
“XMSC_WMQ_MQMD_READ_ENABLED” a pagina 2126	Questa proprietà determina se un'applicazione XMS può estrarre o meno i valori dei campi MQMD.
“XMSC_WMQ_MQMD_WRITE_ENABLED” a pagina 2126	Questa proprietà determina se un'applicazione XMS può impostare o meno i valori dei campi MQMD.

Tabella 875. Proprietà della destinazione (Continua)	
Nome della proprietà	**Descrizione**
“XMSC_WMQ_READ_AHEAD_ALLOWED” a pagina 2127	Questa proprietà determina se ai destinatari dei messaggi e ai browser della coda è consentito utilizzare la lettura anticipata per ottenere i messaggi non persistenti e non transazionali da questa destinazione in un buffer interno prima di riceverli.
“XMSC_WMQ_READ_AHEAD_CLOSE_POLICY” a pagina 2127	Questa proprietà determina, per i messaggi di cui si sta eseguendo la consegna a un listener di messaggi asincroni, cosa succede ai messaggi nel buffer di lettura anticipata interno quando viene chiuso il destinatario del messaggio.
“XMSC_WMQ_RECEIVE_CCSID” a pagina 2132	Proprietà di destinazione che imposta il CCSID di destinazione per la conversione del messaggio del gestore code. Il valore viene ignorato a meno che XMSC_WMQ_RECEIVE_CONVERSION non sia impostato su WMQ_RECEIVE_CONVERSION_QMGR.
“XMSC_WMQ_RECEIVE_CONVERSIONE” a pagina 2132	Proprietà di destinazione che determina se la conversione dati verrà eseguita dal gestore code.
XMSC_WMQ_TARGET_CLIENT	Indica se i messaggi inviati alla destinazione contengono un'intestazione MQRFH2.
XMSC_WMQ_TEMP_TOPIC_PREFIX	Quando si creano argomenti temporanei, XMS genera una stringa di argomenti nel formato "TEMP/TEMPTOPICPREFIX/unique_id" oppure, se questa proprietà contiene il valore predefinito, viene generata questa stringa, "TEMP/unique_id". La specifica di un valore non vuoto consente la definizione di specifiche code modello per la creazione delle code gestite per i sottoscrittori di argomenti temporanei creati in questa connessione.
XMSC_WPM_BUS_NAME	Per una factory di connessione, il nome del bus di integrazione del servizio a cui si connette l'applicazione oppure, per una destinazione, il nome del bus di integrazione del servizio in cui esiste la destinazione.
SPACE XMSC_WPM_TOPIC_XX_ENCODE_CASE_ONE spazio	Il nome dello spazio argomento che contiene l'argomento.

Proprietà di InitialContext

Una panoramica delle proprietà dell'oggetto InitialContext , con link a informazioni di riferimento più dettagliate.

Tabella 876. Proprietà di InitialContext	
Nome della proprietà	**Descrizione**
URL_PROVIDER_IC_XMSC	Utilizzata per individuare la directory di denominazione JNDI in modo che il servizio di denominazione COS non debba necessariamente trovarsi sullo stesso server del servizio web.

Tabella 876. Proprietà di InitialContext (Continua)	
Nome della proprietà	**Descrizione**
XMSC_IC_SECURITY_AUTHENTICATION	Basato su SECURITY_AUTHENTICATION dell'interfaccia di contesto Java . Questa proprietà è applicabile solo al contesto di denominazione COS.
CREDENZIALI XMSC_IC_SECURITY_CREDENTIALS	Basato sull'interfaccia di contesto Java SECURITY_CREDENTIALS. Questa proprietà è applicabile solo al contesto di denominazione COS.
XMSC_IC_SECURITY_PRINCIPAL	In base all'interfaccia di contesto Java SECURITY_PRINCIPAL. Questa proprietà è applicabile solo al contesto di denominazione COS.
PROTOCOLLO_SICUREZZA_XMSC_ICA	In base all'interfaccia di contesto Java SECURITY_PROTOCOL Questa proprietà è applicabile solo al contesto di denominazione COS.
URL_IC_XMSC	Per i contesti LDAP e FileSystem, l'indirizzo del repository che contiene gli oggetti amministrati. Per i contesti di denominazione COS, l'indirizzo del servizio web che cerca gli oggetti nella directory.

Proprietà del messaggio

Una panoramica delle proprietà dell'oggetto Messaggio, con link a informazioni di riferimento più dettagliate.

Tabella 877. Proprietà del messaggio	
Nome della proprietà	**Descrizione**
IMPOSTAZIONE_IBM_JMS	L'identificativo (CCSID) della serie di caratteri codificati, o codepage, in cui si trovano le stringhe di dati di caratteri nel contenuto del messaggio quando il client XMS inoltra il messaggio alla destinazione desiderata. In XMS, questa proprietà dispone di un valore numerico ed esegue l'associazione al CCSID. Tuttavia, questa proprietà si basa su una proprietà JMS per cui dispone di un valore di tipo stringa ed esegue l'associazione al set di caratteri Java che rappresenta questo CCSID numerico.
JMS_IBM_Encoding	Modalità di rappresentazione dei dati numerici nel corpo del messaggio quando il client XMS inoltra il messaggio alla destinazione desiderata.
MESSAGGIO_IBM_JMS	Testo che descrive perché il messaggio è stato inviato alla destinazione eccezioni. Questa proprietà è di sola lettura.
JMS_IBM_EXCEPTIONPROBLEMDESTINATION	Il nome della destinazione in cui si trovava il messaggio prima che il messaggio venisse inviato alla destinazione eccezioni.
JMS_IBM_EXCEPTIONREASON	Un codice motivo che indica il motivo per cui il messaggio è stato inviato alla destinazione eccezioni.
JMS_IBM_EXCEPTIONTIMESTAMP	L'ora in cui il messaggio è stato inviato alla destinazione eccezioni.
JMS_IBM_FEEDBACK	Un codice che indica la natura di un messaggio di report.

Tabella 877. Proprietà del messaggio (Continua)	
Nome della proprietà	**Descrizione**
FORM_IBM_JMS	La natura dei dati dell'applicazione nel messaggio.
JMS_IBM_LAST_MSG_IN_GROUP	Indicare se il messaggio è l'ultimo messaggio in un gruppo di messaggi.
TIPO_IBM_JMS	Il tipo del messaggio.
TIPO_IBM_JMS	Il tipo di applicazione che ha inviato il messaggio.
DATA_IBM_JMS	La data in cui il messaggio è stato inviato.
ORA_IBM_JMS	L'ora in cui il messaggio è stato inviato.
COA IBM REPORT_JMS	Richiedere i messaggi di report di 'conferma all'arrivo', specificando quanti dati dell'applicazione dal messaggio originale devono essere inclusi in un messaggio di report.
COD REPORT_IBM_JMS	Richiedere i messaggi di report di 'conferma alla consegna', specificando quanti dati dell'applicazione dal messaggio originale devono essere inclusi in un messaggio di report.
MSG_DISCARD_IBM_REPORT_JMS JMS	Richiedere che il messaggio venga eliminato se non può essere consegnato alla sua destinazione prevista.
JMS_IBM_REPORT_ECCEZIONE	Richiedere dei messaggi di report di eccezione, specificando quanti dati dell'applicazione dal messaggio originale devono essere inclusi in un messaggio di report.
JMS_IBM_REPORT_EXPIRATION	Richiedere dei messaggi di report di scadenza, specificando quanti dati dell'applicazione dal messaggio originale devono essere inclusi in un messaggio di report.
NAN_IBM_REPORT_JMS	Richiedere i messaggi di report di notifica di azione non eseguita con esito positivo.
PAN IBM_REPORT_JMS	Richiedere i messaggi di report di notifica di azione eseguita con esito positivo.
ID_CORREL_IBM_REPORT_JMS	Richiedere che l'identificativo correlazioni di qualsiasi messaggio di report o di risposta sia uguale all'identificativo correlazioni del messaggio originale.
ID_MSG_PASS_IBM_REPORT_JMS	Richiedere che l'identificativo del messaggio di qualsiasi messaggio di report o di risposta sia uguale all'identificativo del messaggio originale.
RETAIN IBM JMS	L'impostazione di questa proprietà indica al gestore code di trattare un messaggio come Pubblicazione conservata.
ID_MESSAGGIO_SISTEMA_JMS	Un identificativo che identifica il messaggio in modo univoco all'interno del bus di integrazione del servizio. Questa proprietà è di sola lettura.
IDAPP_JMS	Il nome dell'applicazione che ha inviato il messaggio.
CONTEGGIO_DISTRIBUZIONE_JMSX	Il numero di tentativi di consegna del messaggio.
IDGROUP_JMS	L'identificativo del gruppo messaggi a cui appartiene il messaggio.
GROUPSEQ JMSX	Il numero di sequenza del messaggio all'interno di un gruppo messaggi.

Tabella 877. Proprietà del messaggio (Continua)	
Nome della proprietà	**Descrizione**
IDUSER_JMS	L'identificativo utente associato all'applicazione che ha inviato il messaggio.

Proprietà JMS_IBM_MQMD*

IBM Message Service Client for .NET consente alle applicazioni client di leggere / scrivere i campi MQMD utilizzando API. Consente inoltre l'accesso ai dati dei messaggi di MQ . Per impostazione predefinita, l'accesso a MQMD è disattivato e deve essere abilitato esplicitamente dall'applicazione utilizzando le proprietà di destinazione XMSC_WMQ_MQMD_WRITE_ENABLED e XMSC_WMQ_MQMD_READ_ENABLED. Queste due proprietà sono indipendenti l'una dall'altra.

Tutti i campi MQMD tranne StrucId e Version sono esposti come proprietà aggiuntive dell'oggetto Message e hanno come prefisso JMS_IBM_MQMD.

Le proprietà JMS_IBM_MQMD* hanno la precedenza su altre proprietà come JMS_IBM* descritte nella tabella precedente.

invio di messaggi

Tutti i campi MQMD tranne StrucId e Version sono rappresentati. Queste proprietà fanno riferimento solo ai campi MQMD; dove una proprietà si verifica sia nell'intestazione MQMD che MQRFH2 , la versione in MQRFH2 non è impostata o estratta. È possibile impostare una qualsiasi di queste proprietà, ad eccezione di JMS_IBM_MQMD_BackoutCount. Qualsiasi valore impostato per JMS_IBM_MQMD_BackoutCount viene ignorato.

Se una proprietà ha una lunghezza massima e si fornisce un valore troppo lungo, il valore viene troncato.

Per alcune proprietà, è necessario impostare anche la proprietà XMSC_WMQ_MQMD_MESSAGE_CONTEXT sull'oggetto Destinazione. L'applicazione deve essere in esecuzione con l'autorizzazione di contesto appropriata perché questa proprietà diventi effettiva. Se non si imposta XMSC_WMQ_MQMD_MESSAGE_CONTEXT su un valore appropriato, il valore della proprietà viene ignorato. Se si imposta XMSC_WMQ_MQMD_MESSAGE_CONTEXT su un valore appropriato ma non si dispone dell'autorizzazione di contesto sufficiente per il gestore code, viene emessa un'eccezione. Le proprietà che richiedono valori specifici di XMSC_WMQ_MQMD_MESSAGE_CONTEXT sono le seguenti.

Le seguenti proprietà richiedono che XMSC_WMQ_MQMD_MESSAGE_CONTEXT sia impostato su XMSC WMQ_MDCTX_SET_IDENTITY_CONTEXT o XMSC_WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_UserIdentifier
- JMS_IBM_MQMD_AccountingToken
- Dati JMS_IBM_MQMD_ApplIdentity

Le seguenti proprietà richiedono che XMSC_WMQ_MQMD_MESSAGE_CONTEXT sia impostato su XMSC_WMQ_MDCTX_SET_ALL_CONTEXT:

- Tipo JMS_IBM_MQMD_PutAppl
- Nome JMS_IBM_MQMD_PutAppl
- JMS_IBM_MQMD_PutDate
- PutTime JMS_IBM_MQMD_
- JMS_IBM_MQMD_ApplOriginDati

ricezione di messaggi

Tutte queste proprietà sono disponibili su un messaggio ricevuto se la proprietà XMSC_WMQ_MQMD_READ_ENABLED è impostata su true, indipendentemente dalle proprietà effettive impostate dall'applicazione di produzione. Un'applicazione non può modificare le proprietà di un

messaggio ricevuto a meno che tutte le proprietà non vengano prima cancellate, in base alla specifica JMS. Il messaggio ricevuto può essere inoltrato senza modificare le proprietà.

Nota: Se la propria applicazione riceve un messaggio da una destinazione con la proprietà XMSC_WMQ_MQMD_READ_ENABLED impostata su true e lo inoltra a una destinazione con XMSC_WMQ_MQMD_WRITE_ENABLED impostata su true, tutti i valori del campo MQMD del messaggio ricevuto vengono copiati nel messaggio inoltrato. Tabella delle proprietà

Tabella 878. Proprietà dell'oggetto Message che rappresentano i campi MQMD

Proprietà	**Descrizione**	**Tipo**
REPORT IBM_MQMD_JMS	Opzioni per messaggi di report	System.Int32
JMS_IBM_MQMD_MSGTIPO	Tipo messaggio	System.Int32
EXPIRY IBM_MQMD_JMS	Durata messaggio	System.Int32
IBM_MQMD_FEEDBACK JMS	Feedback o codice motivo	System.Int32
JMS_IBM_MQMD_ENCODING	Codifica numerica dei dati di messaggio	System.Int32
ID_CODICE_IBM_MQMD_JMS	CSID (Character set identifier) dei dati del messaggio	System.Int32
JMS_IBM_MQMD_FORMATO	Nome formato dei dati di messaggio	System.String
JMS_IBM_MQMD_PRIORITY **Nota:** Se si assegna un valore a JMS_IBM_MQMD_PRIORITY non compreso nell'intervallo 0-9, questo valore viola la specifica JMS.	Priorità messaggio	System.Int32
JMS_IBM_MQMD_PERSISTENZA	Persistenza messaggio	System.Int32
IDMSG_IBM_MQMD_JMS **Nota:** La specifica JMS indica che l'ID messaggio deve essere impostato dal provider JMS e che deve essere univoco o null. Se si assegna un valore a JMS_IBM_MQMD_MSGID, questo valore viene copiato in JMSMessageID. Quindi non è impostato dal provider JMS e potrebbe non essere univoco: questo valore viola la specifica JMS.	ID messaggio	Matrice byte **Nota:** L'utilizzo delle proprietà della matrice di byte su un messaggio viola le specifiche JMS.
IDCORREL_IBM_MQMD_JMS **Nota:** Se si assegna un valore a JMS_IBM_MQMD_CORRELID che inizia con la stringa 'ID:', questo valore viola la specifica JMS.	Identificativo di correlazione	Matrice byte **Nota:** L'utilizzo delle proprietà della matrice di byte su un messaggio viola le specifiche JMS.
JMS_IBM_MQMD_BACKOUTCOUNT	Contatore backout	System.Int32
QMS_IBM_MQMD_REPLYTOQ	Nome della coda di risposta	System.String
JMS_IBM_MQMD_REPLYTOQMGR	Nome del gestore code di risposta	System.String
JMS_IBM_MQMD_USERIDENTIFIER	Identificativo utente	System.String

Tabella 878. Proprietà dell'oggetto Message che rappresentano i campi MQMD (Continua)		
Proprietà	**Descrizione**	**Tipo**
JMS_IBM_MQMD_ACCOUNTINGTOKEN	Token account	Matrice byte **Nota:** L'utilizzo delle proprietà della matrice di byte su un messaggio viola le specifiche JMS.
JMS_IBM_MQMD_APPLIDENTITYDATA	Dati dell'applicazione correlati all'identità	System.String
PUTAPPLTYPE IBM_MQMD_JMS	Tipo di applicazione che inserisce il messaggio	System.Int32
NomePUTAPPL_IBM_MQMD_JMS	Nome dell'applicazione che ha inserito il messaggio	System.String
PUTDATE MQMD_IBM_JMS	Data in cui il messaggio è stato inserito	System.String
PUTORA_IBM_MQMD_JMS	Ora in cui il messaggio è stato inserito	System.String
JMS_IBM_MQMD_APPLORIGINDATA	Dati dell'applicazione correlati all'origine	System.String
IDGROUP_IBM_MQMD_JMS	ID gruppo	Matrice byte **Nota:** L'utilizzo delle proprietà della matrice di byte su un messaggio viola le specifiche JMS.
JMS_IBM_MQMD_MSGSEQNUMBER	Numero di sequenza del messaggio locale all'interno del gruppo	System.Int32
OFFSET IBM_MQMD_JMS	Offset di dati nel messaggio fisico dall'inizio del messaggio logico	System.Int32
JMS_IBM_MQMD_MSGFLAGS	Indicatori di messaggio	System.Int32
JMS_IBM_MQMD_ORIGINALLENGTH	Lunghezza del messaggio originale	System.Int32

Per ulteriori informazioni, vedere MQMD .

Esempi

Questo esempio determina l'inserimento di un messaggio in una coda o in un argomento con MQMD.UserIdentifier impostato suJoeBloggs".

```
// Create a ConnectionFactory, connection, session, producer, message
// ...

// Create a destination
// ...

// Enable MQMD write
dest.setBooleanProperty(XMSC_WMQ_MQMD_WRITE_ENABLED,
      XMSC_WMQ_MQMD_WRITE_ENABLED_YES);

// Optionally, set a message context if applicable for this MD field
dest.setIntProperty(XMSC_WMQ_MQMD_MESSAGE_CONTEXT,
  XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT);
```

```
// On the message, set property to provide custom UserId
msg.setStringProperty(JMS_IBM_MQMD_USERIDENTIFIER, "JoeBloggs");

// Send the message
// ...
```

È necessario impostare XMSC_WMQ_MQMD_MESSAGE_CONTEXT prima di impostare JMS_IBM_MQMD_USERIDENTIFIER. Per ulteriori informazioni relative all'utilizzo di XMSC_WMQ_MQMD_MESSAGE_CONTEXT, consultare Proprietà dell'oggetto Messaggio.

Allo stesso modo, è possibile estrarre il contenuto dei campi MQMD impostando XMSC_WMQ_MQMD_READ_ENABLED su true prima di ricevere un messaggio e quindi utilizzando i metodi get del messaggio, ad esempio la proprietà getString. Tutte le proprietà ricevute sono di sola lettura.

Questo esempio risulta nel campo del valore che contiene il valore di MQMD.ApplIdentityData di un messaggio ricevuto da una coda o da un argomento.

```
// Create a ConnectionFactory, connection, session, consumer
// ...

// Create a destination
// ...

// Enable MQMD read
dest.setBooleanProperty(XMSC_WMQ_MQMD_READ_ENABLED, XMSC_WMQ_MQMD_READ_ENABLED_YES);

// Receive a message
// ...

// Get required MQMD field value using a property
System.String value = rcvMsg.getStringProperty(JMS_IBM_MQMD_APPLIDENTITYDATA);
```

Proprietà di MessageConsumer

Una panoramica delle proprietà dell'oggetto MessageConsumer , con link a informazioni di riferimento più dettagliate.

Tabella 879. Proprietà di MessageConsumer

Nome della proprietà	Descrizione
XMSC_IS_SUBSCRIPTION_MULTICAST	Indica se i messaggi vengono consegnati al consumatore di messaggi utilizzando WebSphere MQ Multicast Transport. Questa proprietà è di sola lettura.
XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST	Indica se i messaggi vengono consegnati al consumatore di messaggi utilizzando WebSphere MQ Multicast Transport con una QoS (quality of service) affidabile. Questa proprietà è di sola lettura.

Fare riferimento a .Proprietà NET di IMessageConsumer per ulteriori dettagli.

Proprietà di MessageProducer

Una panoramica delle proprietà dell'oggetto MessageProducer , con collegamenti a informazioni di riferimento più dettagliate.

Consultare .Proprietà NET di IMessageProducer per ulteriori dettagli.

Proprietà della sessione

Una panoramica delle proprietà dell'oggetto Session, con link a informazioni di riferimento più dettagliate.

Consultare .Proprietà NET di ISession per ulteriori dettagli.

Definizioni proprietà

Questa sezione fornisce una definizione di ogni proprietà oggetto.

Ogni definizione di proprietà include le seguenti informazioni:

- Il tipo di dati della proprietà
- I tipi di oggetto che hanno la proprietà
- Per una proprietà di Destinazione, il nome che può essere utilizzato in un URI (uniform resource identifier)
- Una descrizione più dettagliata dell'immobile
- I valori validi della proprietà
- Il valore predefinito della proprietà

Le proprietà i cui nomi iniziano con uno dei seguenti prefissi sono rilevanti solo per il tipo di collegamento specificato:

RTT XMSC
Le proprietà sono rilevanti solo per una connessione in tempo reale a un broker. I nomi delle proprietà vengono definiti come costanti denominate nel file di intestazione `xmsc_rtt.h`.

WMQ XMSC
Le proprietà sono rilevanti solo quando un'applicazione si connette ad un gestore code IBM MQ . I nomi delle proprietà vengono definiti come costanti denominate nel file di intestazione `xmsc_wmq.h`.

WPM XMSC
Le proprietà sono pertinenti solo quando un'applicazione si connette a un SIB (service integration bus) WebSphere . I nomi delle proprietà vengono definiti come costanti denominate nel file di intestazione `xmsc_wpm.h`.

Se non diversamente specificato nelle loro definizioni, le restanti proprietà sono rilevanti per tutti i tipi di connessione. I nomi delle proprietà vengono definiti come costanti denominate nel file di intestazione `xmsc.h`. Le proprietà i cui nomi iniziano con il prefisso JMSX sono JMS proprietà definite di un messaggio e le proprietà i cui nomi iniziano con il prefisso JMS_IBM sono IBM proprietà definite di un messaggio. Per ulteriori informazioni sulle proprietà dei messaggi, consultare Proprietà di un messaggio XMS.

Se non diversamente specificato nella sua definizione, ogni proprietà è rilevante sia nei domini point - to - point che in quelli di pubblicazione - sottoscrizione.

Un'applicazione può ottenere e impostare il valore di qualsiasi proprietà, a meno che la proprietà non sia di sola lettura.

JMS_IBM_CHARACTER_SET

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

L'identificativo (CCSID) della serie di caratteri codificati, o codepage, in cui si trovano le stringhe di dati di caratteri nel contenuto del messaggio quando il client XMS inoltra il messaggio alla destinazione desiderata. In XMS, questa proprietà dispone di un valore numerico ed esegue l'associazione al CCSID. Tuttavia, questa proprietà si basa su una proprietà JMS per cui dispone di un valore di tipo stringa ed esegue l'associazione al set di caratteri Java che rappresenta questo CCSID numerico. Questa proprietà sovrascrive qualsiasi CCSID specificato per la destinazione dalla proprietà XMSC_WMQ_CCSID .

Per default, la proprietà non è impostata.

Questa proprietà non è rilevante quando un'applicazione si connette a un SIB (service integration bus).

JMS_IBM_Encoding

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Modalità di rappresentazione dei dati numerici nel corpo del messaggio quando il client XMS inoltra il messaggio alla destinazione desiderata. Questa proprietà sovrascrive qualsiasi codifica specificata per la destinazione dalla proprietà XMSC_WMQ_ENCODING . La proprietà specifica la rappresentazione di numeri interi binari, interi decimali compressi e numeri a virgola mobile.

I valori validi della proprietà sono gli stessi valori che possono essere specificati nel campo **Encoding** di un descrittore di messaggi.

Un'applicazione può utilizzare le seguenti costanti denominate per impostare la proprietà:

Costante con nome	Significato
MQENC_INTEGER_NORMAL	Codifica numero intero normale
MQENC_INTEGER_REVERSED	Codifica numero intero inversa
MQENC_DECIMAL_NORMAL	Codifica decimale compresso normale
MQENC_DECIMAL_REVERSED	Codifica decimale compresso inversa
MQENC_FLOAT_IEEE_NORMAL	Codifica a virgola mobile IEEE normale
MQENC_FLOAT_IEEE_REVERSED	Codifica a virgola mobile IEEE inversa
z/OS MQENC_FLOAT_S390	Codifica a virgola mobile dell'architettura z/OS
MQEN_NATIVE	Codifica macchina nativa

Per formare un valore per la proprietà, l'applicazione può aggiungere tre di queste costanti come segue:

- Una costante il cui nome inizia con MQENC_INTEGER, per specificare la rappresentazione di numeri interi binari
- Una costante il cui nome inizia con MQENC_DECIMAL, per specificare la rappresentazione di numeri interi decimali compressi
- Una costante il cui nome inizia con MQENC_FLOAT, per specificare la rappresentazione dei numeri a virgola mobile

In alternativa, l'applicazione può impostare la proprietà su MQENC_NATIVE, il cui valore dipende dall'ambiente.

Per default, la proprietà non è impostata.

Questa proprietà non è rilevante quando un'applicazione si connette a un SIB (service integration bus).

MESSAGGIO_IBM_JMS

Tipo di dati:
Stringa

Proprietà di:
Messaggio

Testo che descrive perché il messaggio è stato inviato alla destinazione eccezioni. Questa proprietà è di sola lettura.

Questa proprietà è rilevante solo quando un'applicazione si connette a un SIB (service integration bus) e riceve un messaggio da una destinazione eccezioni.

JMS_IBM_EXCEPTIONPROBLEMDESTINATION

Tipo di dati:
Stringa

Proprietà di:
Messaggio

Il nome della destinazione in cui si trovava il messaggio prima che il messaggio venisse inviato alla destinazione eccezioni.

Questa proprietà è rilevante solo quando un'applicazione si connette a un SIB (service integration bus) e riceve un messaggio da una destinazione eccezioni.

JMS_IBM_EXCEPTIONREASON

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Un codice motivo che indica il motivo per cui il messaggio è stato inviato alla destinazione eccezioni.

Questa proprietà è rilevante solo quando un'applicazione si connette a un SIB (service integration bus) e riceve un messaggio da una destinazione eccezioni.

JMS_IBM_EXCEPTIONTIMESTAMP

Tipo di dati:
System.Int64

Proprietà di:
Messaggio

L'ora in cui il messaggio è stato inviato alla destinazione eccezioni.

L'ora è espressa in millisecondi a partire dalle 00:00:00 GMT del 1 ° gennaio 1970.

Questa proprietà è rilevante solo quando un'applicazione si connette a un SIB (service integration bus) e riceve un messaggio da una destinazione eccezioni.

JMS_IBM_FEEDBACK

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Un codice che indica la natura di un messaggio di report.

I valori validi della proprietà sono i codici di feedback e i codici motivo che possono essere specificati nel campo **Feedback** di un descrittore di messaggi.

Per default, la proprietà non è impostata.

JMS_IBM_FORMAT

Tipo di dati:
Stringa

Proprietà di:
Messaggio

La natura dei dati dell'applicazione nel messaggio.

I valori validi della proprietà sono gli stessi valori che possono essere specificati nel campo **Format** di un descrittore di messaggi.

Per default, la proprietà non è impostata.

Questa proprietà non è rilevante quando un'applicazione si connette a un SIB (service integration bus).

JMS_IBM_LAST_MSG_IN_GROUP

Tipo di dati:
System.Boolean

Proprietà di:
Messaggio

Indicare se il messaggio è l'ultimo messaggio in un gruppo di messaggi.

Impostare la proprietà su true se il messaggio è l'ultimo messaggio in un gruppo di messaggi. Altrimenti, impostare la proprietà su false o non impostare la proprietà. Per default, la proprietà non è impostata.

Il valore true corrisponde all'indicatore di stato MQMF_LAST_MSG_IN_GROUP, che può essere specificato nel campo **MsgFlags** di un descrittore di messaggi.

Questa proprietà viene ignorata nel dominio di pubblicazione / sottoscrizione e non è rilevante quando un'applicazione si connette a un SIB (Service Integration Bus).

TIPO_IBM_JMS

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Il tipo del messaggio.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
MQMT_DATAGRAM	Il messaggio non richiede una risposta.
MQMT_REQUEST	Il messaggio richiede una risposta.
MQMT_REPLY	Il messaggio è un messaggio di risposta.
REPORT MQMT	Il messaggio è un messaggio di report.

Questi valori corrispondono ai tipi di messaggi che è possibile specificare nel campo **MsgType** di un descrittore di messaggi.

Per default, la proprietà non è impostata.

Questa proprietà non è rilevante quando un'applicazione si connette a un SIB (service integration bus).

TIPO_PUTAPPL_IBM_JMS

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Il tipo di applicazione che ha inviato il messaggio.

I valori validi della proprietà sono i tipi di applicazione che è possibile specificare nel campo **PutApplType** di un descrittore di messaggi.

Per default, la proprietà non è impostata.

Questa proprietà non è rilevante quando un'applicazione si connette a un SIB (service integration bus).

DATA_IBM_JMS

Tipo di dati:
Stringa

Proprietà di:
Messaggio

La data in cui il messaggio è stato inviato.

I valori validi della proprietà sono gli stessi valori che possono essere specificati nel campo **PutDate** di un descrittore di messaggi.

Per default, la proprietà non è impostata.

Questa proprietà non è rilevante quando un'applicazione si connette a un SIB (service integration bus).

PUTORA_IBM_JMS

Tipo di dati:
Stringa

Proprietà di:
Messaggio

L'ora in cui il messaggio è stato inviato.

I valori validi della proprietà sono gli stessi valori che possono essere specificati nel campo **PutTime** di un descrittore di messaggi.

Per default, la proprietà non è impostata.

Questa proprietà non è rilevante quando un'applicazione si connette a un SIB (service integration bus).

COA IBM_REPORT_JMS

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Richiedere i messaggi di report di 'conferma all'arrivo', specificando quanti dati dell'applicazione dal messaggio originale devono essere inclusi in un messaggio di report.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
COA MQRO	Richiedere i messaggi di report 'conferma all'arrivo', senza dati dell'applicazione dal messaggio originale inclusi in un messaggio di report.
DATA_COA_WITH_MQRO	Richiedere i messaggi di report 'conferma all'arrivo', con i primi 100 byte di dati dell'applicazione dal messaggio originale inclusi in un messaggio di report.
DATI_COA_WITH_MQRO_FULL_DATA	Richiedere i messaggi di report 'conferma all'arrivo', con tutti i dati dell'applicazione del messaggio originale inclusi in un messaggio di report.

Questi valori corrispondono alle opzioni di report che possono essere specificate nel campo **Report** di un descrittore di messaggi. Per ulteriori informazioni su queste opzioni, consultare Report (MQLONG).

Per default, la proprietà non è impostata.

COD IBM REPORT_JMS

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Richiedere i messaggi di report di 'conferma alla consegna', specificando quanti dati dell'applicazione dal messaggio originale devono essere inclusi in un messaggio di report.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
COD MQRO	Richiedere i messaggi di report 'conferma alla consegna', senza dati dell'applicazione dal messaggio originale inclusi in un messaggio di report.
DATI MQRO_COD_WITH_	Richiedere i messaggi di report 'conferma alla consegna', con i primi 100 byte di dati dell'applicazione dal messaggio originale inclusi in un messaggio di report.
DAD_COD MQRO_WITH_FULL_DATA	Richiedere i messaggi di report 'conferma alla consegna', con tutti i dati dell'applicazione dal messaggio originale inclusi in un messaggio di report.

Questi valori corrispondono alle opzioni di report che è possibile specificare nel campo **Report** di un descrittore di messaggi.

Per default, la proprietà non è impostata.

JMS_IBM_REPORT_DISCARD_MSG

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Richiedere che il messaggio venga eliminato se non può essere consegnato alla sua destinazione prevista.

Impostare la proprietà su MQRO_DISCARD_MSG per richiedere che il messaggio venga eliminato se non può essere consegnato alla destinazione prevista. Se invece si richiede che il messaggio venga inserito in una coda di messaggi non recapitabili o inviato a una destinazione eccezioni, non impostare la proprietà. Per default, la proprietà non è impostata.

Il valore MQRO_DISCARD_MSG corrisponde a un'opzione di report che può essere specificata nel campo **Report** di un descrittore di messaggi.

JMS_IBM_REPORT_EXCEPTION

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Richiedere dei messaggi di report di eccezione, specificando quanti dati dell'applicazione dal messaggio originale devono essere inclusi in un messaggio di report.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
MQRO_ECCEZIONE	Richiedere i messaggi di report di eccezioni, senza dati dell'applicazione dal messaggio originale inclusi in un messaggio di report.
MQRO_EXCEPTION_WITH_DATA	Richiedere i messaggi di report di eccezione, con i primi 100 byte di dati dell'applicazione dal messaggio originale inclusi in un messaggio di report.
MQRO_EXCEPTION_WITH_FULL_DATA	Richiedere i messaggi di report di eccezione, con tutti i dati dell'applicazione dal messaggio originale inclusi in un messaggio di report.

Questi valori corrispondono alle opzioni di report che è possibile specificare nel campo **Report** di un descrittore di messaggi.

Per default, la proprietà non è impostata.

SCADENZA_REPORT_IBM_JMS

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Richiedere dei messaggi di report di scadenza, specificando quanti dati dell'applicazione dal messaggio originale devono essere inclusi in un messaggio di report.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
SCADENZA_MQRO	Richiedere i messaggi di report di scadenza, senza dati dell'applicazione dal messaggio originale inclusi in un messaggio di report.
MQRO_EXPIRATION_WITH_DATA	Messaggi di report di scadenza della richiesta, con i primi 100 byte di dati dell'applicazione dal messaggio originale inclusi in un messaggio di report.
MQRO_EXPIRATION_WITH_FULL_DATA	Richiedere i messaggi di report di scadenza, con tutti i dati dell'applicazione del messaggio originale inclusi in un messaggio di report.

Questi valori corrispondono alle opzioni di report che è possibile specificare nel campo **Report** di un descrittore di messaggi.

Per default, la proprietà non è impostata.

NAN_IBM_REPORT_JMS

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Richiedere i messaggi di report di notifica di azione non eseguita con esito positivo.

Impostare la proprietà su MQRO_NAN per richiedere i messaggi di report di notifica azione negativa. Se non si richiedono messaggi di report di notifica azione negativa, non impostare la proprietà. Per default, la proprietà non è impostata.

Il valore MQRO_NAN corrisponde ad un'opzione di report che può essere specificata nel campo **Report** di un descrittore di messaggi.

PAN IBM REPORT_JMS

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Richiedere i messaggi di report di notifica di azione eseguita con esito positivo.

Impostare la proprietà su MQRO_PAN per richiedere i messaggi di report di notifica azione positiva. Se non si richiedono messaggi di report di notifica azione positiva, non impostare la proprietà. Per default, la proprietà non è impostata.

Il valore MQRO_PAN corrisponde ad un'opzione di report che può essere specificata nel campo **Report** di un descrittore di messaggi.

ID_CORREL_PASS_IBM_REPORT_JMS

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Richiedere che l'identificativo correlazioni di qualsiasi messaggio di report o di risposta sia uguale all'identificativo correlazioni del messaggio originale.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
ID CORREL_PASS_MQRO_	Richiedere che l'identificativo correlazioni di qualsiasi messaggio di report o di risposta sia uguale all'identificativo correlazioni del messaggio originale.
ID_COPY_MQRO_MSG_TO_CORREL_ID	Richiedere che l'identificativo di correlazione di qualsiasi report o messaggio di risposta sia uguale all'identificativo del messaggio originale.

Questi valori corrispondono alle opzioni di report che è possibile specificare nel campo **Report** di un descrittore di messaggi.

Il valore predefinito della proprietà è MQRO_COPY_MSG_ID_TO_CORREL_ID.

ID_IBM_REPORT_PASS_MSG_JMS

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Richiedere che l'identificativo del messaggio di qualsiasi messaggio di report o di risposta sia uguale all'identificativo del messaggio originale.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
MQRO_PASS_MSG_ID	Richiedere che l'identificativo del messaggio di qualsiasi messaggio di report o di risposta sia uguale all'identificativo del messaggio originale.

Valore valido	Significato
ID_MSG_NEW_MQRO	Richiedere che venga generato un nuovo identificativo di messaggio per ogni report o messaggio di risposta.

Tali valori corrispondono alle opzioni del prospetto che è possibile specificare nel campo Prospetto di un descrittore di messaggi.

Il valore predefinito della proprietà è MQRO_NEW_MSG_ID.

RETAIN IBM_JMS

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

L'impostazione di questa proprietà indica al gestore code di trattare un messaggio come Pubblicazione conservata. Quando un sottoscrittore riceve messaggi dagli argomenti, potrebbe ricevere ulteriori messaggi immediatamente dopo la sottoscrizione, oltre ai messaggi ricevuti nelle release precedenti. Questi messaggi sono le pubblicazioni facoltative conservate per gli argomenti sottoscritti. Per ogni argomento che corrisponde alla sottoscrizione, se è presente una pubblicazione conservata, la pubblicazione viene resa disponibile per la consegna al destinatario del messaggio di sottoscrizione.

RETAIN_PUBLICATION è l'unico valore valido per questa proprietà. Per default questa proprietà non è impostata.

Nota: Questa proprietà è rilevante solo nel dominio di pubblicazione / sottoscrizione

IDMESSAGGIO_IBM_SISTEMA_JMS

Tipo di dati:
Stringa

Proprietà di:
Messaggio

Un identificativo che identifica il messaggio in modo univoco all'interno del bus di integrazione del servizio. Questa proprietà è di sola lettura.

Questa proprietà è rilevante solo quando un'applicazione si connette a un SIB (service integration bus).

IDAPP_JMSX

Tipo di dati:
Stringa

Proprietà di:
Messaggio

Il nome dell'applicazione che ha inviato il messaggio.

Questa proprietà è la proprietà JMS definita con il JMS nome JMSXAppID. Per ulteriori informazioni sulla proprietà, consultare *Java Message Service Specification, Versione 1.1*.

Per default, la proprietà non è impostata.

Questa proprietà non è valida per una connessione in tempo reale a un broker.

CONTEGGIO_DISTRIBUZIONE_JMSX

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Il numero di tentativi di consegna del messaggio.

Questa proprietà è la proprietà JMS definita con il JMS nome JMSXDeliveryCount. Per ulteriori informazioni sulla proprietà, consultare *Java Message Service Specification, Version 1.1*.

Per default, la proprietà non è impostata.

Questa proprietà non è valida per una connessione in tempo reale a un broker.

IDGRUPPO_JMS

Tipo di dati:
Stringa

Proprietà di:
Messaggio

L'identificativo del gruppo messaggi a cui appartiene il messaggio.

Questa proprietà è la proprietà JMS definita con il JMS nome JMSXGroupID. Per ulteriori informazioni sulla proprietà, consultare *Java Message Service Specification, Version 1.1*.

Per default, la proprietà non è impostata.

Questa proprietà non è valida per una connessione in tempo reale a un broker.

GROUPSEQ JMS

Tipo di dati:
System.Int32

Proprietà di:
Messaggio

Il numero di sequenza del messaggio all'interno di un gruppo messaggi.

Questa proprietà è la proprietà JMS definita con il JMS nome JMSXGroupSeq. Per ulteriori informazioni sulla proprietà, consultare *Java Message Service Specification, Version 1.1*.

Per default, la proprietà non è impostata.

Questa proprietà non è valida per una connessione in tempo reale a un broker.

IDUSER_JMS

Tipo di dati:
Stringa

Proprietà di:
Messaggio

L'identificativo utente associato all'applicazione che ha inviato il messaggio.

Questa proprietà è la proprietà JMS definita con il JMS nome JMSXUserID. Per ulteriori informazioni sulla proprietà, consultare *Java Message Service Specification, Version 1.1*.

Per default, la proprietà non è impostata.

Questa proprietà non è valida per una connessione in tempo reale a un broker.

XMSC_ASYNC_EXCEPTIONS

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Oggetti applicabili:
Nome completo strumento di gestione JMS: ASYNCEXCEPTION

Nome breve dello strumento di amministrazione JMS: AEX

Questa proprietà determina se XMS informa un ExceptionListener solo quando una connessione viene interrotta o quando si verifica qualsiasi eccezione in modo asincrono per una chiamata API XMS. Questa proprietà si applica a tutte le connessioni create da questa ConnectionFactory che hanno un ExceptionListener registrato.

I valori validi per questa proprietà sono:

XMSC_ASYNC_EXCEPTIONS_ALL
Qualsiasi eccezione rilevata in modo asincrono, al di fuori dell'ambito di una chiamata API sincrona e tutte le eccezioni di connessione interrotta vengono inviate a ExceptionListener.

XMSC_ASYNC_EXCEPTIONS_CONNECTIONBROKEN
Solo le eccezioni che indicano una connessione interrotta vengono inviate a ExceptionListener. Tutte le altre eccezioni che si verificano durante l'elaborazione asincrona non vengono notificate a ExceptionListenere quindi l'applicazione non viene informata di tali eccezioni.

Per impostazione predefinita, questa proprietà è impostata su XMSC_ASYNC_EXCEPTIONS_ALL.

ID_CLI_XMSC

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Oggetti applicabili:
Nome esteso strumento di amministrazione JMS: CLIENTID

Nome breve strumento di amministrazione JMS: CID

L'identificativo client per una connessione.

Un identificativo client viene utilizzato solo per supportare sottoscrizioni durevoli nel dominio di pubblicazione / sottoscrizione e viene ignorato nel dominio point - to - point. Per ulteriori informazioni sull'impostazione degli identificativi client, consultare ConnectionFactories e oggetti Connection.

Questa proprietà non è rilevante per una connessione in tempo reale a un broker.

XMSC_CONNECTION_TYPE

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Il tipo di server di messaggistica a cui si connette un'applicazione.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
RTT CT_XMSC	Una connessione in tempo reale a un broker.
CT_XMSC - WMQ	Una connessione a un gestore code IBM MQ .
WPM CT_XMSC	Una connessione a un WebSphere Application Server service integration bus.

Per default, la proprietà non è impostata.

*MODALITÀ_DISTRIBUZIONE_XMSC_*

Tipo di dati:
System.Int32

Proprietà di:
Destinazione

Nome utilizzato in un URI:

`persistenza` (per una destinazione IBM MQ)
`deliveryMode` (per una destinazione del provider di messaggistica predefinita WebSphere)

Oggetti applicabili:
Nome lungo strumento di amministrazione JMS: PERSISTENCE

Nome breve strumento di amministrazione JMS: PER

La modalità di consegna del messaggio inviato alla destinazione.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
XMSC_DELIVERY_NON_PERSISTENTE	Un messaggio inviato alla destinazione è provvisorio. La modalità di consegna predefinita del mittente del messaggio o qualsiasi modalità di consegna specificata nella chiamata di invio viene ignorata. Se la destinazione è una coda IBM MQ , anche il valore dell'attributo della coda *DefPersistence* viene ignorato.
XMSC_DELIVERY_PERSISTENT	Un messaggio inviato alla destinazione è permanente. La modalità di consegna predefinita del mittente del messaggio o qualsiasi modalità di consegna specificata nella chiamata di invio viene ignorata. Se la destinazione è una coda IBM MQ , anche il valore dell'attributo della coda *DefPersistence* viene ignorato.
XMSC_DELIVERY_AS_APP	Un messaggio inviato alla destinazione ha la modalità di consegna specificata nella chiamata di invio. Se la chiamata di invio non specifica alcuna modalità di consegna, viene utilizzata la modalità di distribuzione predefinita del produttore del messaggio. Se la destinazione è una coda IBM MQ , il valore dell'attributo della coda *DefPersistence* viene ignorato.
XMSC_DELIVERY_AS_DEST	Se la destinazione è una coda IBM MQ , un messaggio inserito nella coda ha la modalità di consegna specificata dal valore dell'attributo della coda *DefPersistence*. La modalità di consegna predefinita del mittente del messaggio o qualsiasi modalità di consegna specificata nella chiamata di invio viene ignorata. Se la destinazione non è una coda IBM MQ , il significato è lo stesso di XMSC_DELIVERY_AS_APP.

Il valore predefinito è XMSC_DELIVERY_AS_APP.

*URL PROVIDER_IC_XMSC_*

Tipo di dati:
Stringa

Proprietà di:
InitialContext

Utilizzata per individuare la directory di denominazione JNDI in modo che il servizio di denominazione COS non debba necessariamente trovarsi sullo stesso server del servizio web.

*AUTENTICAZIONE_SICUREZZA_IC_XMSC_*

Tipo di dati:
Stringa

Proprietà di:
InitialContext

Basato su SECURITY_AUTHENTICATION dell'interfaccia di contesto Java . Questa proprietà è applicabile solo al contesto di denominazione COS.

CREDENZIALI XMSC_IC_SECURITY_CREDENTIALS

Tipo di dati:
Stringa

Proprietà di:
InitialContext

Basato sull'interfaccia di contesto Java SECURITY_CREDENTIALS. Questa proprietà è applicabile solo al contesto di denominazione COS.

*PRINCIPALE_SICUREZZA_ICA_XMSC_*

Tipo di dati:
Stringa

Proprietà di:
InitialContext

In base all'interfaccia di contesto Java SECURITY_PRINCIPAL. Questa proprietà è applicabile solo al contesto di denominazione COS.

*PROTOCOLLO_SICUREZZA_IC_XMSC_*

Tipo di dati:
Stringa

Proprietà di:
InitialContext

In base all'interfaccia di contesto Java SECURITY_PROTOCOL Questa proprietà è applicabile solo al contesto di denominazione COS.

URL IC_XMSC

Tipo di dati:
Stringa

Proprietà di:
InitialContext

Per i contesti LDAP e FileSystem, l'indirizzo del repository che contiene gli oggetti amministrati.

Per i contesti LDAP e FileSystem, l'indirizzo del repository che contiene gli oggetti amministrati.

XMSC_IS_SUBSCRIPTION_MULTICAST

Tipo di dati:
System.Boolean

Proprietà di:
MessageConsumer

Indica se i messaggi vengono consegnati al consumatore di messaggi utilizzando WebSphere MQ Multicast Transport. Questa proprietà è di sola lettura.

Il valore della proprietà è true se i messaggi vengono consegnati al consumatore di messaggi utilizzando WebSphere MQ Multicast Transport. Altrimenti, il valore è false.

Questa proprietà è rilevante solo per una connessione in tempo reale a un broker.

XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST

Tipo di dati:
System.Boolean

Proprietà di:
MessageConsumer

Indica se i messaggi vengono consegnati al consumatore di messaggi utilizzando WebSphere MQ Multicast Transport con una QoS (quality of service) affidabile. Questa proprietà è di sola lettura.

Il valore della proprietà è true se i messaggi vengono consegnati al consumatore di messaggi utilizzando WebSphere MQ Multicast Transport con una qualità di servizio affidabile. Altrimenti, il valore è false.

Questa proprietà è rilevante solo per una connessione in tempo reale a un broker.

XMSC_JMS_MAJOR_VERSION

Tipo di dati:
System.Int32

Proprietà di:
Dati ConnectionMeta

Il numero di versione principale della specifica JMS su cui si basa XMS . Questa proprietà è di sola lettura.

VERSIONE_MINORE_JMS_XMSC_XX_ENCODE_CASE_CAPS_LOCK_OFF

Tipo di dati:
System.Int32

Proprietà di:
Dati ConnectionMeta

Il numero di versione minore della specifica JMS su cui si basa XMS . Questa proprietà è di sola lettura.

VERSIONE JMS XMSC

Tipo di dati:
Stringa

Proprietà di:
Dati ConnectionMeta

L'identificativo della versione della specifica JMS su cui si basa XMS . Questa proprietà è di sola lettura.

*VERSIONE_MAGGIORE_XMSC_*

Tipo di dati:
System.Int32

Proprietà di:
Dati ConnectionMeta

Il numero di versione del client XMS . Questa proprietà è di sola lettura.

XMSC_VERSIONE_MINORE

Tipo di dati:
System.Int32

Proprietà di:
Dati ConnectionMeta

Il numero di release del client XMS . Questa proprietà è di sola lettura.

*PASSWORD_XMSC_*

Tipo di dati:
Array di byte

Proprietà di:
ConnectionFactory

Una password che può essere utilizzata per autenticare l'applicazione quando prova a connettersi a un server di messaggistica. La password viene utilizzata con la proprietà XMSC_USERID .

Per default, la proprietà non è impostata.

Multi Se ci si connette a IBM MQ su Multipiattaformee si imposta la proprietà XMSC_USERID della factory di connessione, deve corrispondere al **userid** dell'utente collegato. Se non si impostano queste proprietà, il gestore code utilizza il **userid** dell'utente collegato per impostazione predefinita. Se si richiede un'ulteriore autenticazione a livello di connessione di singoli utenti, è possibile scrivere un'uscita di autenticazione client configurata in IBM MQ. Per ulteriori informazioni sulla creazione di un'uscita di autenticazione client, consultare Pianificazione dell'autenticazione per un'applicazione client.

z/OS Per autenticare l'utente durante la connessione a IBM MQ for z/OS è necessario utilizzare un'uscita di sicurezza.

*PRIORITÀ_XMSC_*

Tipo di dati:
System.Int32

Proprietà di:
Destinazione

Nome utilizzato in un URI:
priorità

La priorità dei messaggi inviati alla destinazione.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
Un numero intero compreso nell'intervallo 0, la priorità più bassa, a 9, la priorità più alta	Un messaggio inviato alla destinazione ha la priorità specificata. La priorità predefinita del mittente del messaggio o qualsiasi priorità specificata nella chiamata di invio viene ignorata. Se la destinazione è una coda IBM MQ , anche il valore dell'attributo della coda **DefPriority** viene ignorato.
APPLICAZIONE_PRIORITÀ_XMSC_	Un messaggio inviato alla destinazione ha la priorità specificata nella chiamata di invio. Se la chiamata di invio non specifica alcuna priorità, viene utilizzata la priorità predefinita del mittente del messaggio. Se la destinazione è una coda IBM MQ , il valore dell'attributo della coda **DefPriority** viene ignorato.
XMSC_PRIORITÀ_AS_DEST	Se la destinazione è una coda IBM MQ , un messaggio inserito nella coda ha la priorità specificata dal valore dell'attributo della coda **DefPriority**. La priorità predefinita del mittente del messaggio o qualsiasi priorità specificata nella chiamata di invio viene ignorata. Se la destinazione non è una coda IBM MQ , il significato è lo stesso di XMSC_PRIORITY_AS_APP.

Il valore predefinito è XMSC_PRIORITY_AS_APP.

WebSphere MQ Real-Time Transport e WebSphere MQ Multicast Transport non intrapprendono alcuna azione in base alla priorità di un messaggio.

NOME_PROVIDER_XMSCO

Tipo di dati:
Stringa

Proprietà di:
Dati ConnectionMeta

Il fornitore del client XMS . Questa proprietà è di sola lettura.

XMSC_RTT_BROKER_PING_INTERVAL

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

L'intervallo di tempo, in millisecondi, dopo il quale XMS .NET controlla la connessione a un server di messaggistica in tempo reale per rilevare eventuale attività. Se non viene rilevata alcuna attività, il client avvia un ping; la connessione viene chiusa se non viene rilevata alcuna risposta al ping.

Il valore predefinito della proprietà è 30000.

*PROTOCOLLO_COLLEGAMENTO_RTT_XMSC_*

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Il protocollo di comunicazione utilizzato per una connessione in tempo reale a un broker.

Il valore della proprietà deve essere XMSC_RTT_CP_TCP, che indica una connessione in tempo reale a un broker su TCP/IP. Il valore predefinito è XMSC_RTT_CP_TCP.

*NOME_HOST_RTT_XMSC_*

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il nome host o l'indirizzo IP del sistema su cui viene eseguito un broker.

Questa proprietà viene utilizzata con la proprietà XMSC_RTT_PORT per identificare il broker.

Per default, la proprietà non è impostata.

XMSC_RTT_LOCAL_ADDRESS

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il nome host o l'indirizzo IP dell'interfaccia di rete locale da utilizzare per una connessione in tempo reale a un broker.

Questa proprietà è utile solo se il sistema su cui è in esecuzione l'applicazione ha due o più interfacce di rete ed è necessario poter specificare quale interfaccia deve essere utilizzata per una connessione in

tempo reale. Se il sistema dispone di una sola interfaccia di rete, è possibile utilizzare solo tale interfaccia. Se il sistema ha due o più interfacce di rete e la proprietà non è impostata, l'interfaccia viene selezionata in modo casuale.

Per default, la proprietà non è impostata.

XMSC_RTT_MULTICAST

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory e Destinazione

Nome utilizzato in un URI:
mulicast

L'impostazione multicast per una destinazione o una factory di connessione. Solo una destinazione che è un argomento può avere questa proprietà.

Un'applicazione utilizza questa proprietà per abilitare il multicast in associazione con una connessione in tempo reale ad un broker e, se il multicast è abilitato, per specificare il modo preciso in cui il multicast viene utilizzato per consegnare i messaggi dal broker ad un consumatore di messaggi. La proprietà non ha alcun effetto sul modo in cui un produttore di messaggi invia i messaggi al broker.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
XMSC_RTT_MULTICAST_DISABLED	I messaggi non vengono consegnati a un consumatore di messaggi utilizzando WebSphere MQ Multicast Transport. Questo è il valore predefinito per un oggetto ConnectionFactory .
XMSC_RTT_MULTICAST_ASCF	I messaggi vengono consegnati ad un consumatore di messaggi in base all'impostazione multicast per la produzione connessioni associata al consumatore di messaggi. L'impostazione multicast per il factory di connessione viene annotato al momento della creazione della connessione. Questo valore è valido solo per un oggetto Destinazione ed è il valore predefinito per un oggetto Destinazione.
XMSC_RTT_MULTICAST_ENABLED	Se l'argomento è configurato per multicast nel broker, i messaggi vengono consegnati a un consumatore di messaggi utilizzando WebSphere MQ Multicast Transport. Viene utilizzata una QoS (quality of service) affidabile se l'argomento è configurato per il multicast affidabile.
XMSC_RTT_MULTICAST_AFFIDABILE	Se l'argomento è configurato per il multicast affidabile nel broker, i messaggi vengono consegnati a un consumatore di messaggi utilizzando WebSphere MQ Multicast Transport con una QoS (quality of service) affidabile. Se l'argomento non è configurato per il multicast affidabile, non è possibile creare un consumer di messaggi per l'argomento.

Valore valido	Significato
XMSC_RTT_MULTICAST_NON_AFFIDABILE	Se l'argomento è configurato per multicast nel broker, i messaggi vengono consegnati a un consumatore di messaggi utilizzando WebSphere MQ Multicast Transport. Una QoS (quality of service) affidabile non viene utilizzata anche se l'argomento è configurato per il multicast affidabile.

PORTA_RTT_XMSC

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Il numero della porta su cui un broker è in ascolto per le richieste in entrata. Sul broker, è necessario configurare un nodo di elaborazione dei messaggi Real -timeInput o Real -timeOptimizedFlow per essere in ascolto su questa porta.

Questa proprietà viene utilizzata con la proprietà XMSC_RTT_HOST_NAME per identificare il broker.

Il valore predefinito della proprietà è XMSC_RTT_DEFAULT_PORT o 1506.

XMSC_TIME_TO_LIVE

Tipo di dati:
System.Int32

Proprietà di:
Destinazione

Nome utilizzato in un URI:

scadenza (per una destinazione IBM MQ)
timeToLive (per una destinazione del fornitore di messaggistica predefinito WebSphere)

La durata (TTL, Time To Live) per i messaggi inviati alla destinazione.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
0	Un messaggio inviato alla destinazione non scade mai.
Un numero intero positivo	Un messaggio inviato alla destinazione ha la durata specificata in millisecondi. Il TTL (time to live) predefinito del produttore del messaggio o qualsiasi TTL (time to live) specificato nella chiamata di invio viene ignorato.
XMSC_TIME_TO_LIVE_AS_APP	Un messaggio inviato alla destinazione ha la durata specificata nella chiamata di invio. Se la chiamata di invio non specifica alcuna durata, viene utilizzata la durata predefinita del produttore del messaggio.

Il valore predefinito è XMSC_TIME_TO_LIVE_AS_APP.

IDUSER_XMSC

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Un identificativo utente che può essere utilizzato per autenticare l'applicazione quando prova a connettersi a un server di messaggistica. L'identificativo utente viene utilizzato con la propriet ... XMSC_PASSWORD .

Per default, la proprietà non è impostata.

Multi Se ci si connette a IBM MQ for Multiplatformse si imposta la proprietà XMSC_USERID della factory di connessione, deve corrispondere al **userid** dell'utente collegato. Se non si impostano queste proprietà, il gestore code utilizza il **userid** dell'utente collegato per impostazione predefinita. Se si richiede un'ulteriore autenticazione a livello di connessione di singoli utenti, è possibile scrivere un'uscita di autenticazione client configurata in IBM MQ. Per ulteriori informazioni sulla creazione di un'uscita di autenticazione client, consultare Pianificazione dell'autenticazione per un'applicazione client.

z/OS Per autenticare l'utente durante la connessione a IBM MQ for z/OS è necessario utilizzare un'uscita di sicurezza.

VERSIONE XMSC

Tipo di dati:
Stringa

Proprietà di:
Dati ConnectionMeta

L'identificativo della versione della cliXMSent. Questa proprietà è di sola lettura.

*TIPO_APPLICAZIONE XMSC_WMQ_BALANCING_*

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
XMSC_WMQ_BALANCING_APPLICATION_TYPE_SIMPLE	Bilanciamento semplice; non vengono applicate regole specifiche oltre a quelle descritte in Influencing application re-balancing in uniform clusters. Questo è il valore predefinito.
XMSC_WMQ_BALANCING_APPLICATION_TYPE_REQUEST_REPLY	Bilanciamento richiesta - risposta; dopo ogni chiamata MQPUT, è prevista una chiamata MQGET corrispondente per un messaggio di risposta. Il bilanciamento viene ritardato fino a quando non viene ricevuto un messaggio di questo tipo o fino a quando non viene superato il messaggio di richiesta SCADENZA

Inoltre, questa proprietà può essere impostata nel file `client.ini` . L'ordine di preferenza è:

1. Proprietà impostate nell'applicazione
2. Corrispondenza della stanza dell'applicazione denominata nel file `mqclient.ini` .
3. Stanza dei valori predefiniti dell'applicazione nel file `mqclient.ini` .

*OPZIONI_BILANCIAMENTE_WMQ_XMSC_*

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Valore corrispondente
XMSC_WMQ_BALANCING_OPTIONS_NONE	Nessuna opzione impostata. Questo è il valore predefinito
XMSC_WMQ_BALANCING_OPTIONS_IGNORE_TRANSACTIONS	L'impostazione di questa opzione consente alle applicazioni di essere ribilanciate anche se nel mezzo di una transazione.

Inoltre, questa proprietà può essere impostata nel file `client.ini` . L'ordine di preferenza è:

1. Proprietà impostate nell'applicazione
2. Corrispondenza della stanza dell'applicazione denominata nel file `mqclient.ini` .
3. Stanza dei valori predefiniti dell'applicazione nel file `mqclient.ini` .

*TIMEOUT_BILANCIAMENTO_WMQ_XMSC_*

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
XMSC_WMQ_BALANCING_TIMEOUT_IMMEDIATE	Si verifica un timeout immediato
XMSC_WMQ_BALANCING_TIMEOUT_AS_DEFAULT	Il valore di timeout predefinito impostato. Questo è il valore predefinito
XMSC_WMQ_BALANCING_TIMEOUT_NEVER	Non si verifica alcun timeout

Nota: È necessario fornire un solo valore dai valori definiti oppure un valore compreso tra 0 e 999999999 secondi.

Inoltre, questa proprietà può essere impostata nel file `client.ini` . L'ordine di preferenza è:

1. Proprietà impostate nell'applicazione
2. Corrispondenza della stanza dell'applicazione denominata nel file `mqclient.ini` .
3. Stanza dei valori predefiniti dell'applicazione nel file `mqclient.ini` .

BROKER_CONTROLQ WMQ_XMSC

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il nome della coda di controllo utilizzata da un Broker.

Il valore predefinito della proprietà è SYSTEM.BROKER.CONTROL.QUEUE.

Questa proprietà è rilevante solo nel dominio di pubblicazione / sottoscrizione.

BROKER_PUBQ WMQ_XMSC

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il nome della coda monitorata da un broker dove le applicazioni inviano i messaggi che pubblicano.

Il valore predefinito della proprietà è SYSTEM.BROKER.DEFAULT.STREAM.

Questa proprietà è rilevante solo nel dominio di pubblicazione / sottoscrizione.

Gestore code BROKER_WMQ_XMSC

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il nome del gestore code a cui è connesso un broker.

Per default, la proprietà non è impostata.

Questa proprietà è rilevante solo nel dominio di pubblicazione / sottoscrizione.

BROKER_SUBQ WMQ_XMSC

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il nome della coda sottoscrittore per un utilizzatore di messaggi non durevole.

Il nome della coda del sottoscrittore deve iniziare con i seguenti caratteri:

SYSTEM.JMS.ND.

Se si desidera che tutti gli utenti di messaggi non durevoli condividano una coda del sottoscrittore, specificare il nome completo della coda condivisa. Una coda con il nome specificato deve esistere prima che un'applicazione possa creare un consumer di messaggi non durevoli.

Se si desidera che ogni utente di messaggi non durevoli richiami i messaggi dalla propria coda di sottoscrittori esclusiva, specificare un nome coda che termini con un asterisco (*). Quindi, quando un'applicazione crea un consumer di messaggi non duraturo, il client XMS crea una coda dinamica per l'uso esclusivo da parte del consumer di messaggi. Il client XMS utilizza il valore della proprietà per impostare il contenuto del campo **DynamicQName** nel descrittore dell'oggetto utilizzato per creare la coda dinamica.

Il valore predefinito della proprietà è SYSTEM.JMS.ND.SUBSCRIBER.QUEUE, che significa che XMS utilizza l'approccio della coda condivisa per impostazione predefinita.

Questa proprietà è rilevante solo nel dominio di pubblicazione / sottoscrizione.

*VERSIONE_BROKER_WMQ_XMSC_*

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory e Destinazione

Nome utilizzato in un URI:
brokerVersion

Il tipo di broker utilizzato dall'applicazione per una connessione o per la destinazione. Solo una destinazione che è un argomento può avere questa proprietà.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
XMSC_WMQ_BROKER_V1	L'applicazione utilizza un broker di pubblicazione / sottoscrizione IBM MQ . L'applicazione può utilizzare questo valore anche se si esegue la migrazione dalla pubblicazione / sottoscrizione IBM MQ a WebSphere Message Broker ma non ha modificato l'applicazione.
XMSC_WMQ_BROKER_V2	L'applicazione utilizza un broker di IBM Integration Bus.
XMSC_WMQ_BROKER non specificato	Dopo la migrazione del broker, impostare questa proprietà in modo che le intestazioni RFH2 non vengano più utilizzate. Dopo la migrazione, questa proprietà non è più rilevante.

Il valore predefinito per una factory di connessione è XMSC_WMQ_BROKER_UNSPECIFIED ma, per impostazione predefinita, la proprietà non è impostata per una destinazione. L'impostazione della proprietà per una destinazione sovrascrive qualsiasi valore specificato dalla proprietà del factory di connessione.

URL CCDT WMQ_XMSC

Tipo di dati:
System.String

Proprietà di:
ConnectionFactory

Oggetti applicabili:
Nome esteso strumento di gestione JMS: CCDTURL

Nome breve dello strumento di amministrazione JMS: CCDT

Un URL (A Uniform Resource Locator) che identifica il nome e l'ubicazione del file che contiene la tabella di definizione del canale client e che specifica anche come è possibile accedere al file.

Per impostazione predefinita, questa proprietà non è impostata.

CCSID WMQ_XMSC

Tipo di dati:
System.Int32

Proprietà di:
Destinazione

Nome utilizzato in un URI:
CCSID

L'identificativo (CCSID) della serie di caratteri codificati, o codepage, in cui si trovano le stringhe di dati carattere nel corpo di un messaggio quando il client XMS inoltra il messaggio alla destinazione. Se impostata per un singolo messaggio, la proprietà JMS_IBM_CHARACTER_SET sovrascrive il CCSID specificato per la destinazione da questa proprietà.

Il valore predefinito della proprietà è 1208.

Questa proprietà è relativa solo ai messaggi inviati alla destinazione, non ai messaggi ricevuti dalla destinazione.

XMSC_WMQ_CHALLEGATO

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Oggetti applicabili:
Nome esteso dello strumento di amministrazione JMS: CHANNEL

Nome breve dello strumento di amministrazione JMS: CHAN

Il nome del canale da utilizzare per una connessione.

Per default, la proprietà non è impostata.

Questa proprietà è rilevante solo quando un'applicazione si connette a un gestore code in modalità client.

XMSC_WMQ_CLIENT_RECONNECT_OPTIONS

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Oggetti applicabili:
Nome esteso dello strumento di amministrazione JMS: CLIENTRECONNECTOPTIONS

Nome breve strumento di amministrazione JMS: CROPT

Questa proprietà specifica le opzioni di riconnessione client per le nuove connessioni create da questo factory. Si trova in XMSC ed è uno dei seguenti:

- WMQ_CLIENT_RECONNECT_AS_DEF (predefinito). Utilizzare il valore specificato nel file `mqclient.ini` . Impostare il valore utilizzando la proprietà **DefRecon** nella sezione Canali. Può essere impostato su uno dei seguenti:
  1. Sì. Si comporta come l''opzione WMQ_CLIENT_RECONNECT
  2. No. Valore predefinito. Non specifica alcuna opzione di riconnessione
  3. QMGR. Si comporta come l'opzione WMQ_CLIENT_RECONNECT_Q_MGR
  4. DISABILITATO. Si comporta come l'opzione WMQ_CLIENT_RECONNECT_DISABLED
- WMQ_CLIENT_RECONNECT. Riconnettersi a uno dei gestori code specificati nell'elenco dei nomi di connessione.
- WMQ_CLIENT_RECONNECT_Q_MGR. Si riconnette allo stesso gestore code a cui è originariamente connesso. Restituisce `MQRC_RECONNECT_QMID_MISMATCH` se il gestore code a cui tenta di connettersi (specificato nell'elenco dei nomi di connessione) ha un QMID differente per il gestore code a cui è originariamente connesso.
- WMQ_CLIENT_RECONNECT_DISABLED. La riconnessione è disabilitata.

XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Oggetti applicabili:
Nome esteso dello strumento di amministrazione JMS: CLIENTRECONNECTTIMEOUT

Nome breve dello strumento di amministrazione JMS: CRT

La proprietà XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT è valida solo per il client XMS .NET gestito.

Questa proprietà specifica il periodo di tempo, in secondi, in cui una connessione client tenta di riconnettersi.

Dopo aver tentato di riconnettersi per questo periodo di tempo, il client avrà esito negativo con MQRC_RECONNECT_FAILED. L'impostazione predefinita per questa proprietà è XMSC.WMQ_CLIENT_RECONNECT_TIMEOUT_DEFAULT.

Il valore predefinito di questa proprietà è 1800.

XMSC_WMQ_CONNECTION_MODE

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

La modalità in base alla quale un'applicazione si connette a un gestore code.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
XMSC_WMQ_CM_BINDINGS	Una connessione a un gestore code in modalità bind, per prestazioni ottimali. Questo è il valore predefinito per C/C + +.
XMSC_WMQ_CM_CLIENT	Una connessione a un gestore code in modalità client, per garantire uno stack completamente gestito. Questo valore è il valore predefinito per .NET.
XMSC_WMQ_CM_CLIENT_UNMANAGED (solo per .NET)	Una connessione a un gestore code che forza uno stack client non gestito.

*ELENCO_NOMI_CONNESSIONI_WMQ_XMSC_*

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Oggetti applicabili:
Nome esteso strumento di amministrazione JMS: CONNECTIONNAMELIST

Nome breve dello strumento di gestione JMS: CNLIST

Questa proprietà specifica gli host a cui il client tenta di riconnettersi dopo che la connessione è stata interrotta.

L'elenco dei nomi di connessione è un elenco separato da virgole di coppie host / porta ip. L'impostazione predefinita per questa proprietà è WMQ_CONNECTION_NAME_LIST_DEFAULT.

Ad esempio,`127.0.0.1 (1414),host2.example.com(1400)`

L'impostazione predefinita di questa proprietà è `localhost (1414)`.

XMSC_WMQ_DUR_SUBQ

Tipo di dati:
Stringa

Proprietà di:
Destinazione

il nome della coda sottoscrittore per un sottoscrittore durevole che sta ricevendo messaggi dalla destinazione. Solo una destinazione che è un argomento può avere questa proprietà.

Il nome della coda del sottoscrittore deve iniziare con i seguenti caratteri:

SYSTEM.JMS.D.

Se si desidera che tutti i sottoscrittori durevoli condividano una coda del sottoscrittore, specificare il nome completo della coda condivisa. Una coda con il nome specificato deve esistere prima che un'applicazione possa creare un sottoscrittore durevole.

Se si desidera che ogni sottoscrittore durevole richiami i messaggi dalla propria coda di sottoscrittori esclusiva, specificare un nome coda che termini con un asterisco (*). Quindi, quando un'applicazione crea un sottoscrittore duraturo, il client XMS crea una coda dinamica per l'uso esclusivo da parte del sottoscrittore durevole. Il client XMS utilizza il valore della proprietà per impostare il contenuto del campo **DynamicQName** nel descrittore dell'oggetto utilizzato per creare la coda dinamica.

Il valore predefinito della proprietà è SYSTEM.JMS.D.SUBSCRIBER.QUEUE, che significa che XMS utilizza l'approccio della coda condivisa per impostazione predefinita.

Questa proprietà è rilevante solo nel dominio di pubblicazione / sottoscrizione.

XMSC_WMQ_ENCODING

Tipo di dati:
System.Int32

Proprietà di:
Destinazione

Modalità di rappresentazione dei dati numerici nel corpo di un messaggio quando il client XMS inoltra il messaggio alla destinazione. Se è impostata per un singolo messaggio, la proprietà JMS_IBM_ENCODING sovrascrive la codifica specificata per la destinazione da questa proprietà. La proprietà specifica la rappresentazione di numeri interi binari, interi decimali compressi e numeri a virgola mobile.

I valori validi della proprietà sono uguali ai valori che possono essere specificati nel campo **Encoding** di un descrittore di messaggi.

Un'applicazione può utilizzare le seguenti costanti denominate per impostare la proprietà:

Costante con nome	Significato
MQENC_INTEGER_NORMAL	Codifica numero intero normale
MQENC_INTEGER_REVERSED	Codifica numero intero inversa
MQENC_DECIMAL_NORMAL	Codifica decimale compresso normale
MQENC_DECIMAL_REVERSED	Codifica decimale compresso inversa
MQENC_FLOAT_IEEE_NORMAL	Codifica a virgola mobile IEEE normale
MQENC_FLOAT_IEEE_REVERSED	Codifica a virgola mobile IEEE inversa
z/OS MQENC_FLOAT_S390	Codifica a virgola mobile dell'architettura z/OS
MQEN_NATIVE	Codifica macchina nativa

Per formare un valore per la proprietà, l'applicazione può aggiungere tre di queste costanti come segue:

- Una costante il cui nome inizia con MQENC_INTEGER, per specificare la rappresentazione di numeri interi binari
- Una costante il cui nome inizia con MQENC_DECIMAL, per specificare la rappresentazione di numeri interi decimali compressi
- Una costante il cui nome inizia con MQENC_FLOAT, per specificare la rappresentazione dei numeri a virgola mobile

In alternativa, l'applicazione può impostare la proprietà su MQENC_NATIVE, il cui valore dipende dall'ambiente.

Il valore predefinito della proprietà è MQENC_NATIVE.

Questa proprietà è relativa solo ai messaggi inviati alla destinazione, non ai messaggi ricevuti dalla destinazione.

XMSC_WMQ_FAIL_IF_QUIESCE

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory e Destinazione

Nome utilizzato in un URI:
FAILIFQUIESCE

Oggetti applicabili:
Nome completo dello strumento di gestione JMS: FAILIFQUIESCE

Nome breve strumento di amministrazione JMS: FIQ

Indicazione della mancata riuscita delle chiamate a specifici metodi se il gestore code a cui è connessa l'applicazione è in uno stato di non attivo.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
SÌ WMQ_XMSC	Le chiamate a determinati metodi hanno esito negativo se il gestore code si trova in uno stato di inattività. Quando l'applicazione rileva che il gestore code è in fase di sospensione, può completare l'attività immediata e chiudere la connessione, consentendo l'arresto del gestore code.
N. FIQ WMQ_XMSC	Nessuna chiamata al metodo non riesce perché il gestore code è in uno stato di inattività. Se si specifica questo valore, l'applicazione non è in grado di rilevare che il gestore code è in fase di sospensione. L'applicazione potrebbe continuare ad eseguire operazioni sul gestore code e quindi impedire l'arresto del gestore code.

Il valore predefinito per una factory di connessioni è XMSC_WMQ_FIQ_YES ma, per impostazione predefinita, la proprietà non è impostata per una destinazione. L'impostazione della proprietà per una destinazione sovrascrive qualsiasi valore specificato dalla proprietà del factory di connessione.

*CORPO messaggio wmq_xmsc_*

Tipo di dati:
System.Int32

Proprietà di:
Destinazione

Questa proprietà determina se un'applicazione XMS elabora MQRFH2 di un messaggio IBM MQ come parte del payload del messaggio (ovvero, come parte del contenuto del messaggio).

Nota: Quando si inviano messaggi a una destinazione, la proprietà XMSC_WMQ_MESSAGE_BODY sostituisce la proprietà di destinazione XMS esistente XMSC_WMQ_TARGET_CLIENT.

I valori validi per questa proprietà sono:

XMSC_WMQ_MESSAGE_BODY_JMS
Ricezione: il tipo e il corpo del messaggio XMS in entrata sono determinati dal contenuto di MQRFH2 (se presente) o di MQMD (se non è presente alcun MQRFH2) nel messaggio IBM MQ ricevuto.

Invia : il corpo del messaggio XMS in uscita contiene un'intestazione di MQRFH2 prestata e generata automaticamente in base alle proprietà e ai campi di intestazione del messaggio XMS .

MESSAGE_BODY_MQ XMSC_WMQ_MQ
Ricezione: il tipo di messaggio XMS in entrata è sempre ByteMessage, indipendentemente dal contenuto del messaggio IBM MQ ricevuto o dal campo del formato dell'MQMD ricevuto. Il corpo

del messaggio XMS è costituito dai dati del messaggio non modificati restituiti dalla chiamata API del fornitore di messaggistica sottostante. La serie di caratteri e la codifica dei dati nel corpo del messaggio sono determinati dai campi CodedCharSetId e Codifica di MQMD. Il formato dei dati nel corpo del messaggio è determinato dal campo Formato di MQMD.

Invia: il corpo del messaggio in uscita XMS contiene il payload dell'applicazione così com' è e nessuna intestazione IBM MQ generata automaticamente viene aggiunta al corpo.

XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED

Ricezione: Il client XMS determina un valore adatto per questa proprietà. Nel percorso di ricezione, questo valore è il valore della proprietà WMQ_MESSAGE_BODY_JMS.

Invia : il client XMS determina un valore adatto per questa proprietà. Nel percorso di invio, questo valore è il valore della proprietà XMSC_WMQ.

Per impostazione predefinita, questa proprietà è impostata su XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED.

XMSC_WMQ_MQMD_MESSAGE_CONTEXT

Tipo di dati:
System.Int32

Proprietà di:
Destinazione

Determina quale livello di contesto del messaggio deve essere impostato dall'applicazione XMS . L'applicazione deve essere in esecuzione con l'autorizzazione di contesto appropriata perché questa proprietà diventi effettiva.

I valori validi per questa proprietà sono:

XMSC_WMQ_MDCTX_DEFAULT
Per i messaggi in uscita, la chiamata API MQOPEN e la struttura MQPMO non specificano opzioni di contesto del messaggio esplicite.

XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT
La chiamata API MQOPEN specifica l'opzione di contesto del messaggio MQOO_SET_IDENTITY_CONTEXT e la struttura di MQPMO specifica MQPMO_SET_IDENTITY_CONTEXT.

XMSC_WMQ_MDCTX_SET_ALL_CONTEXT
La chiamata API MQOPEN specifica l'opzione di contesto del messaggio MQOO_SET_ALL_CONTEXT e la struttura MQPMO specifica MQPMO_SET_ALL_CONTEXT.

Per impostazione predefinita questa proprietà è impostata su XMSC_WMQ_MDCTX_DEFAULT.

Nota: Questa proprietà non è rilevante quando un'applicazione si connette a WebSphere Application Server service integration bus.

Le seguenti proprietà richiedono che la proprietà XMSC_WMQ_MQMD_MESSAGE_CONTEXT sia impostata sul valore della proprietà XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT o sul valore della proprietà XMSC_WMQ_MDCTX_SET_ALL_CONTEXT quando si invia un messaggio per ottenere l'effetto desiderato:

- JMS_IBM_MQMD_USERIDENTIFIER
- JMS_IBM_MQMD_ACCOUNTINGTOKEN
- JMS_IBM_MQMD_APPLIDENTITYDATA

Le seguenti proprietà richiedono che la proprietà XMSC_WMQ_MQMD_MESSAGE_CONTEXT sia impostata sul valore della proprietà XMSC_WMQ_MDCTX_SET_ALL_CONTEXT quando si invia un messaggio per ottenere l'effetto desiderato:

- PUTAPPLTYPE IBM_MQMD_JMS
- NomePUTAPPL_IBM_MQMD_JMS

- PUTDATE MQMD_IBM_JMS
- PUTORA_IBM_MQMD_JMS
- JMS_IBM_MQMD_APPLORIGINDATA

XMSC_WMQ_MQMD_READ_ENABLED

Tipo di dati:
System.Int32

Proprietà di:
Destinazione

Questa proprietà determina se un'applicazione XMS può estrarre o meno i valori dei campi MQMD.

I valori validi per questa proprietà sono:

XMSC_WMQ_READ_ENABLED_NO
Quando si inviano messaggi, le proprietà JMS_IBM_MQMD* di un messaggio inviato non vengono aggiornate per riflettere i valori dei campi aggiornati in MQMD.

Quando si ricevono i messaggi, nessuna delle proprietà JMS_IBM_MQMD* è disponibile su un messaggio ricevuto, anche se alcune o tutte sono impostate dal mittente.

XMSC_WMQ_READ_ENABLED_SI
Quando si inviano messaggi, tutte le proprietà JMS_IBM_MQMD* di un messaggio inviato vengono aggiornate in modo da riflettere i valori del campo aggiornati in MQMD, incluse quelle proprietà che il mittente non ha impostato esplicitamente.

Quando si ricevono messaggi, tutte le proprietà JMS_IBM_MQMD* sono disponibili su un messaggio ricevuto, incluse quelle che il mittente non ha impostato esplicitamente.

Per impostazione predefinita, questa proprietà è impostata su XMSC_WMQ_READ_ENABLED_NO.

XMSC_WMQ_MQMD_WRITE_ENABLED

Tipo di dati:
System.Int32

Proprietà di:
Destinazione

Questa proprietà determina se un'applicazione XMS può impostare o meno i valori dei campi MQMD.

I valori validi per questa proprietà sono:

XMSC_WMQ_WRITE_ENABLED_NO
Tutte le proprietà JMS_IBM_MQMD* vengono ignorate e i relativi valori non vengono copiati nella struttura MQMD sottostante.

XMSC_WMQ_WRITE_ENABLED_SÌ
Le proprietà JMS_IBM_MQMD* vengono elaborate. I loro valori vengono copiati nella struttura MQMD sottostante.

Per impostazione predefinita, questa proprietà è impostata su XMSC_WMQ_WRITE_ENABLED_NO.

XMSC_WMQ_PUT_ASYNC_ALLOWED

Tipo di dati:
System.Int32

Proprietà di:
Destinazione

Questa proprietà determina se ai mittenti del messaggio è consentito utilizzare i put asincroni per inviare messaggi a questa destinazione.

I valori validi per questa proprietà sono:

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_DEST
Determinare se gli inserimenti asincroni sono consentiti facendo riferimento alla definizione dell'argomento o della coda.

XMSC_WMQ _UT_ASYNC_ALLOWED_AS_Q_DEF
Determinare se gli inserimenti asincroni sono consentiti facendo riferimento alla definizione della coda.

_PUT_ASYNC_ALLOWED_AS_TOPIC_DEF XMSC_WMQ
Determinare se gli inserimenti asincroni sono consentiti facendo riferimento alla definizione dell'argomento.

XMSC_WMQ _PUT_ASYNC_ALLOWED_DISABLED
Le immissioni asincrone non sono consentite.

XMSC_WMQ _PUT_ASYNC_ALLOWED_ENABLED
Le immissioni asincrone sono consentite.

Per impostazione predefinita questa proprietà è impostata su XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_DEST.

Nota: Questa proprietà non è rilevante quando un'applicazione si connette a WebSphere Application Server service integration bus.

XMSC_WMQ_READ_AHEAD_ALLOWED

Tipo di dati:
System.Int32

Proprietà di:
Destinazione

Questa proprietà determina se ai destinatari dei messaggi e ai browser della coda è consentito utilizzare la lettura anticipata per ottenere i messaggi non persistenti e non transazionali da questa destinazione in un buffer interno prima di riceverli.

I valori validi per questa proprietà sono:

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF
Determinare se la lettura anticipata è consentita facendo riferimento alla definizione della coda.

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_ ARGOMENTO _DEF
Determinare se la lettura anticipata è consentita facendo riferimento alla definizione dell'argomento.

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_DEST
Determinare se la lettura anticipata è consentita facendo riferimento alla definizione dell'argomento o della coda.

XMSC_WMQ_READ_AHEAD_ALLOWED_DISABLED
La lettura anticipata non è consentita durante l'utilizzo o l'esplorazione dei messaggi.

XMSC_WMQ_READ_AHEAD_ALLOWED_ENABLED
La lettura anticipata è consentita.

Per impostazione predefinita, questa proprietà è impostata su XMSC_WMQ _READ_AHEAD_ALLOWED_AS_DEST.

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY

Tipo di dati:
System.Int32

Proprietà di:
Destinazione

Questa proprietà determina, per i messaggi di cui si sta eseguendo la consegna a un listener di messaggi asincroni, cosa succede ai messaggi nel buffer di lettura anticipata interno quando viene chiuso il destinatario del messaggio.

Questa proprietà è applicabile nella specifica delle opzioni della coda di chiusura quando si utilizzano i messaggi da una destinazione e non è applicabile quando si inviano i messaggi a una destinazione.

Questa proprietà viene ignorata per i browser delle code poiché durante l'esplorazione i messaggi sono ancora disponibili nelle code.

I valori validi per questa proprietà sono:

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT
Solo la chiamata del listener dei messaggi corrente viene completata prima della restituzione, lasciando potenzialmente i messaggi nel buffer di lettura anticipata interno, che vengono quindi eliminati.

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_ALL
Tutti i messaggi nel buffer di lettura anticipata interno vengono consegnati al listener dei messaggi dell'applicazione prima della restituzione.

Per impostazione predefinita, questa proprietà è impostata su XMSC_WMQ _READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT.

Nota:

Fine anomala dell'applicazione
Tutti i messaggi nel buffer di lettura anticipata vengono persi quando un'applicazione XMS termina bruscamente.

Implicazioni per le transazioni
La lettura anticipata è disabilitata quando le applicazioni utilizzano le transazioni. Quindi, l'applicazione non vede alcuna differenza nel comportamento quando utilizzano le sessioni transazionali.

Implicazioni delle modalità di riconoscimento della sessione
La lettura anticipata è abilitata su una sessione non sottoposta a transazione quando le modalità di riconoscimento sono XMSC_AUTO_ACKNOWLEDGEMENT o XMSC_DUPS_OK_ACKNOWLEDGEMENT. La lettura anticipata è disabilitata se la modalità di riconoscimento della sessione è XMSC_CLIENT_ACKNOWLEDGEMENT indipendentemente dalle sessioni transazionali o non transazionali.

Implicazioni per i browser delle code e i selettori dei browser delle code
I browser delle code e i selettori dei browser delle code, utilizzati in applicazioni XMS , ottengono il vantaggio delle prestazioni dalla lettura anticipata. La chiusura del browser della coda non degrada le prestazioni poiché il messaggio è ancora disponibile nella coda per ulteriori operazioni. Non ci sono altre implicazioni per i browser delle code e i selettori dei browser delle code a parte i vantaggi delle prestazioni di lettura anticipata.

Nome_HOST_WMQ_XMSC

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Oggetti applicabili:
Nome esteso strumento di gestione JMS: HOSTNAME

Nome breve dello strumento di amministrazione JMS: HOST

Il nome host o l'indirizzo IP del sistema su cui viene eseguito un gestore code.

Questa proprietà viene utilizzata solo quando un'applicazione si connette a un gestore code in modalità client. La proprietà viene utilizzata con la proprietà XMSC_WMQ_PORT per identificare il gestore code.

Il valore predefinito della proprietà è `localhost`.

XMSC_WMQ_LOCAL_ADDRESS

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Oggetti applicabili:
Nome esteso dello strumento di gestione JMS: LOCALADDRESS

Nome breve strumento di amministrazione JMS: LA

Per una connessione a un gestore code, questa proprietà specifica l'interfaccia di rete locale da utilizzare oppure la porta locale o l'intervallo di porte locali da utilizzare oppure entrambe le cose.

Il valore della proprietà è una stringa con il seguente formato:

[*nome_host*] [(*porta_inferiore*) [,*porta_alto*]]]

I significati delle variabili sono i seguenti:

nome_host
Il nome host o l'indirizzo IP dell'interfaccia di rete locale da utilizzare per la connessione.

Fornire queste informazioni è necessario solo se il sistema su cui è in esecuzione l'applicazione ha due o più interfacce di rete ed è necessario essere in grado di specificare quale interfaccia deve essere utilizzata per la connessione. Se il sistema dispone di una sola interfaccia di rete, è possibile utilizzare solo tale interfaccia. Se il sistema ha due o più interfacce di rete e non si specifica quale interfaccia deve essere utilizzata, l'interfaccia viene selezionata in modo casuale.

porta_inferiore
Il numero della porta locale da utilizzare per la connessione.

Se viene specificato anche *high_port* , *low_port* viene interpretato come il numero di porta più basso in un intervallo di numeri di porta.

porta_alto
Il numero di porta più alto in un intervallo di numeri di porta. Una delle porte nell'intervallo specificato deve essere utilizzata per la connessione.

La lunghezza massima della stringa è 48 caratteri.

Ecco alcuni esempi di valori validi della proprietà:

JUPITER
9.20.4.98
JUPITER (1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)

Per default, la proprietà non è impostata.

Questa proprietà è rilevante solo quando un'applicazione si connette a un gestore code in modalità client.

*SELEZIONE_MESSAGGIO_WMQ_XMSC_*

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Determina se la selezione del messaggio viene effettuata dal client XMS o dal broker.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
XMSC_WMQ_MSEL_CLIENT	La selezione del messaggio viene effettuata dal client XMS .
XMSC_WMQ_MSEL_BROKER	La selezione del messaggio viene effettuata dal broker.

Il valore predefinito è XMSC_WMQ_MSEL_CLIENT.

Questa proprietà è rilevante solo nel dominio di pubblicazione / sottoscrizione. La selezione dei messaggi da parte del Broker non è supportata se la proprietà XMSC_WMQ_BROKER_VERSION è impostata su XMSC_WMQ_BROKER_V1.

XMSC_WMQ_MSG_BATCH_SIZE

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Il numero massimo di messaggi da richiamare da una coda in un singolo batch quando si utilizza la consegna di messaggi asincrona.

Quando un'applicazione utilizza la consegna asincrona dei messaggi, in determinate condizioni, il client XMS richiama un batch di messaggi da una coda prima di inoltrare ogni messaggio singolarmente all'applicazione. Questa proprietà specifica il numero massimo di messaggi che possono trovarsi nel batch.

Il valore della proprietà è un numero intero positivo e il valore predefinito è 10. Impostare la proprietà su un valore diverso solo se si ha un problema di prestazioni specifico che è necessario risolvere.

Se un'applicazione è connessa a un gestore code su una rete, l'aumento del valore di questa proprietà può ridurre i sovraccarichi di rete e i tempi di risposta, ma aumentare la quantità di memoria richiesta per memorizzare i messaggi sul sistema client. Al contrario, l'abbassamento del valore di questa proprietà potrebbe aumentare i sovraccarichi di rete e i tempi di risposta, ma ridurre la quantità di memoria richiesta per memorizzare i messaggi.

*INTERVALLO_POLLING_WMQ_XMSC_*

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Se ciascun listener messaggi in una sessione non dispone di alcun messaggio adatto nella propria coda, questo valore è l'intervallo massimo, in millisecondi, che trascorre prima che ciascun listener messaggi provi di nuovo ad ottenere un messaggio dalla propria coda.

Se si verifica spesso che non è disponibile alcun messaggio adatto per nessuno dei listener di messaggi in una sessione, aumentare il valore di questa proprietà.

Il valore della proprietà è un numero intero positivo. Il valore predefinito è 5000.

PORTA_WMQ_XMSC

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Oggetti applicabili:
Nome esteso dello strumento di amministrazione JMS: PORT

Nome breve dello strumento di amministrazione JMS: PORT

Il numero della porta su cui un gestore code è in ascolto per le richieste in entrata.

Questa proprietà viene utilizzata solo quando un'applicazione si connette a un gestore code in modalità client. La proprietà viene utilizzata con la proprietà XMSC per identificare il gestore code.

Il valore predefinito della proprietà è XMSC_WMQ_DEFAULT_CLIENT_PORT o 1414.

XMSC_WMQ_PROVIDER_VERSION

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

La versione, la release, il livello di modifica e il fix pack del gestore code al quale l'applicazione intende connettersi. I valori validi per questa proprietà sono:

- Non specificato

Oppure una stringa in uno dei seguenti formati

- V.R.M.F
- V.R.M
- V.R
- V

dove V, R, M e F sono numeri interi maggiori o uguali a zero.

Per impostazione predefinita, questa proprietà è impostata su "non specificato".

La versione di IBM MQ Client ha anche un ruolo importante nel determinare se un'applicazione client XMS può utilizzare funzioni specifiche di IBM MQ .

Nota: Una proprietà di sistema XMSC_WMQ_OVERRIDEPROVIDERVERSION sovrascrive la proprietà XMSC_WMQ_PROVIDER_VERSION. Questa proprietà può essere utilizzata se non è possibile modificare l'impostazione del factory di connessione.

XMSC_WMQ_PUB_ACK_INTERVAL

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Il numero di messaggi pubblicati da un publisher prima che il client XMS richieda un riconoscimento dal broker.

Se si diminuisce il valore di questa proprietà, il client richiede i riconoscimenti più spesso e quindi le prestazioni del publisher diminuiscono. Se si aumenta il valore, il client attende un periodo più lungo prima di inviare un'eccezione in caso di errore del broker.

Il valore della proprietà è un numero intero positivo. Il valore predefinito è 25.

*CCSID QMGR XMSC_WMQ_*

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

L'identificativo (CCSID) della serie di caratteri codificati, o codepage, in cui i campi di dati carattere definiti in MQI (Message Queue Interface) vengono scambiati tra il client XMS e quello IBM MQ . Questa proprietà non si applica alle stringhe di dati carattere nei corpi dei messaggi.

Quando l'applicazione XMS si connette a un gestore code in modalità client, il client XMS si collega al client IBM MQ . Le informazioni scambiate tra i due client contengono campi di dati carattere definiti in MQI. In circostanze normali, il client IBM MQ presuppone che questi campi si trovino nella codepage del sistema su cui sono in esecuzione i client. Se il client XMS fornisce e prevede di ricevere questi campi in una codepage differente, è necessario impostare questa proprietà per informare il client IBM MQ .

Quando il client IBM MQ inoltra questi campi di dati di caratteri al gestore code, i dati in essi contenuti devono essere convertiti, se necessario, nella codepage utilizzata dal gestore code. Allo stesso modo, quando il client IBM MQ riceve questi campi dal gestore code, i dati in essi contenuti devono essere convertiti, se necessario, nella codepage in cui il client XMS prevede di ricevere i dati. Il client IBM MQ utilizza questa proprietà per eseguire queste conversione dati.

Per default, la proprietà non è impostata.

L'impostazione di questa proprietà è equivalente all'impostazione della variabile di ambiente MQCCSID per un client IBM MQ che supporta le applicazioni client IBM MQ native. Per ulteriori informazioni su questa variabile di ambiente, consultare MQCCSID.

XMSC_WMQ_QUEUE_MANAGER

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Oggetti applicabili:
Nome esteso dello strumento di amministrazione JMS: QMANAGER

Nome breve strumento di amministrazione JMS: QMGR

Il nome del gestore code a cui connettersi.

Per default, la proprietà non è impostata.

XMSC_WMQ_RECEIVE_CCSID

Proprietà di destinazione che imposta il CCSID di destinazione per la conversione del messaggio del gestore code. Il valore viene ignorato a meno che XMSC_WMQ_RECEIVE_CONVERSION non sia impostato su WMQ_RECEIVE_CONVERSION_QMGR.

Tipo di dati:
Numero intero

Valore:
Qualsiasi numero intero positivo.

Il valore predefinito è 1208.

La specifica di un valore `GMO_CONVERT` in un messaggio è facoltativa. Se viene specificato un valore `GMO_CONVERT` , la conversione viene eseguita in base al valore specificato.

XMSC_WMQ_RECEIVE_CONVERSIONE

Proprietà di destinazione che determina se la conversione dati verrà eseguita dal gestore code.

Tipo di dati:
Numero intero

Valori:
XMSC_WMQ_RECEIVE_CONVERSION_CLIENT_MSG (VALORE PREDEFINITO): eseguire la conversione dati solo sul client XMS . La conversione viene sempre eseguita utilizzando la codepage 1208.

XMSC: eseguire la conversione dei dati sul gestore code prima di inviare un messaggio al client XMS .

XMSC_WMQ_RECEIVE_EXIT

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Identifica un'uscita di ricezione del canale da eseguire.

Il valore della proprietà è una stringa che identifica un'uscita di ricezione del canale e ha il formato seguente:

libraryName(NomeentryPoint)

dove,

- **libraryName** è il percorso completo dell'uscita gestita `.dll`
- `entryPointName` è il nome della classe qualificato dallo spazio dei nomi

Ad esempio, `C:\MyReceiveExit.dll(MyReceiveExitNameSpace.MyReceiveExitClassName)`

Per default, la proprietà non è impostata.

Questa proprietà è rilevante solo quando un'applicazione si connette a un gestore code in modalità client gestito. Inoltre, sono supportate solo le uscite gestite.

XMSC_WMQ_RECEIVE_EXIT_INIT

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

I dati utente passati a un'uscita di ricezione del canale quando viene richiamata.

Il valore della proprietà è una stringa. Per default, la proprietà non è impostata.

Questa proprietà è rilevante solo quando un'applicazione si connette ad un gestore code in modalità client gestito e la proprietà “XMSC_WMQ_RECEIVE_EXIT” a pagina 2133 è impostata.

XMSC_WMQ_RESOLVED_QUEUE_MANAGER

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Questa proprietà viene utilizzata per ottenere il nome del gestore code a cui è connesso.

Quando viene utilizzato con CCDT (Client Channel Definition Table), questo nome potrebbe essere diverso dal nome del gestore code specificato nel factory di connessione.

*ID_MANAGER_XMSC_WMQ_RESOLVED_QUEUE_*

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Questa proprietà viene popolata con l'ID del gestore code dopo la connessione.

XMSC_WMQ_SECURITY_EXIT

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Identifica un'uscita di sicurezza del canale.

Il valore della proprietà è una stringa che identifica un'uscita di sicurezza del canale e ha il formato seguente:

libraryName(NomeentryPoint)

dove,

- **libraryName** è il percorso completo dell'uscita gestita .dll
- entryPointName è il nome della classe qualificato dallo spazio dei nomi

Ad esempio, C:\MySecurityExit.dll(MySecurityExitNameSpace.MySecurityExitClassName)

La lunghezza massima della stringa è 128 caratteri.

Per default, la proprietà non è impostata.

Questa proprietà è rilevante solo quando un'applicazione si connette a un gestore code in modalità client gestito. Inoltre, sono supportate solo le uscite gestite.

XMSC_WMQ_SECURITY_EXIT_INIT

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

I dati utente passati a un'uscita di sicurezza del canale quando viene richiamata.

La lunghezza massima della stringa di dati utente è 32 caratteri.

Per default, la proprietà non è impostata.

Questa proprietà è rilevante solo quando un'applicazione si connette ad un gestore code in modalità client gestito e la proprietà "XMSC_WMQ_SECURITY_EXIT" a pagina 2133 è impostata.

XMSC_WMQ_SEND_EXIT

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Identifica un'uscita di invio del canale.

Il valore della proprietà è una stringa. Un'uscita di invio del canale ha il formato seguente:

libraryName(NomeentryPoint)

dove,

- **libraryName** è il percorso completo dell'uscita gestita .dll
- entryPointName è il nome della classe qualificato dallo spazio dei nomi

Ad esempio: C:\MySendExit.dll(MySendExitNameSpace.MySendExitClassName)

Per default, la proprietà non è impostata.

Questa proprietà è rilevante solo quando un'applicazione si connette a un gestore code in modalità client gestito. Inoltre, sono supportate solo le uscite gestite.

XMSC_WMQ_SEND_EXIT_INIT

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

I dati utente passati alle uscite di invio del canale quando vengono richiamate.

Il valore della proprietà è una stringa di uno o più elementi di dati utente separati da virgole. Per default, la proprietà non è impostata.

Le regole per specificare i dati utente che vengono passati a una sequenza di uscite di invio del canale sono le stesse regole per specificare i dati utente che vengono passati a una sequenza di uscite di ricezione del canale. Per le regole, consultare "XMSC_WMQ_RECEIVE_EXIT_INIT" a pagina 2133.

Questa proprietà è rilevante solo quando un'applicazione si connette ad un gestore code in modalità client gestito e la proprietà "XMSC_WMQ_SEND_EXIT" a pagina 2134 è impostata.

WMQ_XMSC_SEND_CHECK_COUNT

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Il numero di chiamate di invio da consentire tra i controlli per rilevare eventuali errori di put asincrono all'interno di una singola sessione XML non sottoposta a transazione.

Per impostazione predefinita, questa proprietà è impostata su 0.

XMSC_WMQ_SHARE_CONV_ALLOWED

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Oggetti applicabili:
Nome lungo strumento di gestione JMS: SHARECONVALLOWED

Nome breve dello strumento di amministrazione JMS: SCALD

Se una connessione client può condividere il proprio socket con altre connessioni XMS di livello superiore dallo stesso processo allo stesso gestore code, se le definizioni di canale corrispondono. Questa proprietà viene fornita per consentire un isolamento completo delle connessioni in socket separati se necessario per motivi correlati allo sviluppo, alla manutenzione e alla gestione delle applicazioni. L'impostazione di questa proprietà indica semplicemente a XMS di rendere condiviso il socket sottostante. Non indica quante connessioni condividono un singolo socket. Il numero di connessioni che condividono un socket è determinato dal valore SHARECNV negoziato tra il client IBM MQ e il server IBM MQ .

Un'applicazione può impostare le seguenti costanti denominate per impostare la proprietà:

- XMSC_WMQ_SHARE_CONV_ALLOWED_FALSE - Le connessioni non condividono un socket.
- XMSC_WMQ_SHARE_CONV_ALLOWED_TRUE - Le connessioni condividono un socket.

Per impostazione predefinita, la proprietà è impostata su XMSC_WMQ_SHARE_CONV_ALLOWED_ENABLED.

Questa proprietà è rilevante solo quando un'applicazione si connette a un gestore code in modalità client.

XMSC_WMQ_SSL_CERT_STORES

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

L'ubicazione dei server che contengono i CRL (cettificate revocation list) da utilizzare su una connessione SSL a un gestore code.

Il valore della proprietà è un elenco di uno o più URL separati da virgole. Ciascun URL ha il formato seguente:

```
[user[/password]@]ldap://[serveraddress][:portnum][,...]
```

Questo formato è compatibile con, ma esteso da, il formato MQJMS di base.

È valido avere un `serveraddress`vuoto. In questo caso, XMS presume che il valore sia la stringa "localhost".

Un elenco di esempio è:

```
myuser/mypassword@ldap://server1.mycom.com:389
ldap://server1.mycom.com
ldap://
ldap://:389
```

Solo per .NET : le connessioni gestite a IBM MQ (WMQ_CM_CLIENT) e le connessioni non gestite a IBM MQ (WMQ_CM_CLIENT_UNMANAGED) supportano entrambe le connessioni TLS/SSL.

Per default, la proprietà non è impostata.

Concetti correlati
Supporto SSL e TLS per il client .NET non gestito
Supporto SSL e TLS per il client .NET gestito

XMSC_WMQ_SSL_CIPHER_SPEC

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il nome della CipherSpec da utilizzare su una connessione protetta a un gestore code.

Le specifiche di cifratura che puoi utilizzare con il supporto TLS IBM MQ sono riportate nella seguente tabella. Quando si richiede un certificato personale, si specifica una dimensione di chiave per la coppia di chiavi pubblica e privata. La dimensione della chiave utilizzata durante l'handshake SSL è la dimensione memorizzata nel certificato a meno che non sia determinata da CipherSpec, come indicato nella tabella. Per impostazione predefinita, questa proprietà non è impostata.

Nome CipherSpec	Protocollo utilizzato	Algoritmo hash	Algoritmo di codifica	Bit di codifica	FIPS[1]	Suite B a 128 bit	Suite B 192 bit
TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	SHA-1	AES	128	Sì	No	No
TLS_RSA_WITH_AES_256_CBC_SHA[2]	TLS 1.0	SHA-1	AES	256	Sì	No	No
TLS_RSA_WITH_DES_CBC_SHA	TLS 1.0	SHA-1	DES	56	No	No	No
TLS_RSA_WITH_3DES_EDE_CBC_SHA[4]	TLS 1.0	SHA-1	3DES	168	Sì	No	No
TLS_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Sì	No	No

Nome CipherSpec	Protocollo utilizzato	Algoritmo hash	Algoritmo di codifica	Bit di codifica	FIPS[1]	Suite B a 128 bit	Suite B 192 bit
TLS_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Sì	No	No
TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Sì	No	No
TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2	SHA-256	AES	256	Sì	No	No
ECDHE_ECDSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	No	No	No
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	Sì	No	No
ECDHE_RSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	No	No	No
ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	Sì	No	No
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Sì	No	No
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	AES	256	Sì	No	No
ECDHE_RSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Sì	No	No
ECDHE_RSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	AES	256	Sì	No	No
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Sì	Sì	No
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Sì	No	Sì
ECDHE_RSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Sì	No	No
ECDHE_RSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Sì	No	No
TLS_RSA_WITH_NULL_SHA256	TLS 1.2	SHA-256	Nessuna	0	No	No	No
ECDHE_RSA_NULL_SHA256	TLS 1.2	SHA-256	Nessuna	0	No	No	No
ECDHE_ECDSA_NULL_SHA256	TLS 1.2	SHA-256	Nessuna	0	No	No	No
TLS_RSA_WITH_NULL_NULL	TLS 1.2	Nessuna	Nessuna	0	No	No	No
TLS_RSA_WITH_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	No	No	No

Nome CipherSpec	Protocollo utilizzato	Algoritmo hash	Algoritmo di codifica	Bit di codifica	FIPS[1]	Suite B a 128 bit	Suite B 192 bit

Note:

1. Specifica se CipherSpec è conforme a FIPS (Federal Information Processing Standards) 140-2. Per una spiegazione di FIPS e informazioni su come configurare IBM MQ per operazioni compatibili con FIPS 140-2, consultare FIPS (Federal Information Processing Standards).
2. Questo CipherSpec non può essere utilizzato per proteggere una connessione da IBM MQ Explorer a un gestore code, a meno che i file delle politiche non limitati appropriati non vengano applicati al JRE utilizzato da IBM MQ Explorer.
3. Questa CipherSpec era certificata FIPS 140-2 prima del 19 maggio 2007.
4. Quando IBM MQ è configurato per il funzionamento conforme a FIPS 140-2, questo CipherSpec può essere utilizzato per trasferire fino a 32 GB di dati prima che la connessione venga terminata con l'errore AMQ9288. Per evitare questo errore, evitare di utilizzare il triplo DES (obsoleto) o abilitare la reimpostazione della chiave segreta quando si utilizza questa CipherSpec in una configurazione FIPS 140-2.

Concetti correlati

Integrità dei dati dei messaggi

Attività correlate

Protezione

Specifica di CipherSpecs

XMSC_WMQ_SSL_CIPHER_SUITE

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il nome della CipherSuite da utilizzare su una connessione TLS a un gestore code. Il protocollo utilizzato nella negoziazione della connessione protetta dipende dalla CipherSuite specificata.

Questa proprietà ha i seguenti valori canonici:

- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA
- SSL_RSA_EXPORT1024_WITH_RC4_56_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- SSL_RSA_WITH_AES_256_CBC_SHA
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA

Questo valore può essere fornito come un'alternativa a XMSC_WMQ_SSL_CIPHER_SPEC.

Se viene specificato un valore non vuoto per XMSC_WMQ_SSL_CIPHER_SPEC, questo valore sovrascrive l'impostazione per XMSC_WMQ_SSL_CIPHER_SUITE. Se XMSC_WMQ_SSL_CIPHER_SPEC non ha un valore, il valore di XMSC_WMQ_SSL_CIPHER_SUITE viene utilizzato come suite di cifratura da fornire a IBM Global Security Kit (GSKit). In tal caso, il valore viene associato al valore CipherSpec equivalente,

come descritto nelle associazioni di nome CipherSuite e CipherSpec per le connessioni XMS a un gestore code IBM MQ.

Se sia XMSC_WMQ_SSL_CIPHER_SPEC che XMSC_WMQ_SSL_CIPHER_SUITE sono vuoti, il campo pChDef ->SSLCipherSpec viene riempito con spazi.

Solo per .NET : le connessioni gestite a IBM MQ (WMQ_CM_CLIENT) e le connessioni non gestite a IBM MQ (WMQ_CM_CLIENT_UNMANAGED) supportano entrambe le connessioni TLS/SSL.

Per default, la proprietà non è impostata.

Concetti correlati

Supporto SSL e TLS per il client .NET non gestito

Supporto SSL e TLS per il client .NET gestito

XMSC_WMQ_SSL_CRYPTO_HW

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

I dettagli di configurazione per l'hardware di crittografia connesso al sistema client.

Questa proprietà ha i seguenti valori canonici:

- GSK_ACCELERATOR_RAINBOW_CS_OFF
- GSK_ACCELERATOR_RAINBOW_CS_ON
- GSK_ACCELERATOR_NCIPHER_NF_OFF
- GSK_ACCELERATOR_NCIPHER_NF_ON

Esiste un formato speciale per l'hardware crittografico PKCS11 (dove DriverPath, TokenLabele TokenPassword sono stringhe specificate dall'utente):

```
GSK_PKCS11=PKCS#11 DriverPath; PKCS#11 TokenLabel;PKCS#11 TokenPassword
```

XMS non interpreta o modifica il contenuto della stringa. Copia il valore fornito, fino a un limite di 256 caratteri a byte singolo, in MQSCO.CryptoHardware .

Solo per .NET : le connessioni gestite a IBM MQ (WMQ_CM_CLIENT) e le connessioni non gestite a IBM MQ (WMQ_CM_CLIENT_UNMANAGED) supportano entrambe le connessioni TLS/SSL.

Per default, la proprietà non è impostata.

Concetti correlati

Supporto SSL e TLS per il client .NET non gestito

Supporto SSL e TLS per il client .NET gestito

XMSC_WMQ_SSL_FIPS_REQUIRED

Tipo di dati:
Booleano

Proprietà di:
ConnectionFactory

Il valore della proprietà determina se un'applicazione può o meno utilizzare delle suite di crittografia non conformi a FIPS. Se questa proprietà è impostata su true, per la connessione client-server vengono utilizzati solo gli algoritmi FIPS.

Questa proprietà può avere i seguenti valori, che vengono convertiti in due valori canonici per MQSCO MQSCO.FipsRequired:

Tabella 880. Tabella dei valori per MQSCO.FlipsRequired FlipsRequired		
Valore	**Descrizione**	**Valore corrispondente di MQSCO.FipsRequired**
No	È possibile utilizzare qualsiasi CipherSpec .	MQSSL_FIPS_NO (impostazione predefinita)
vero, true	Solo gli algoritmi di crittografia certificati FIPS possono essere utilizzati in CipherSpec che si applica a questa connessione client.	SÌ MQSSL_FIPS

XMS copia il relativo valore in MQSCO.FipsRequired prima di richiamare MQCONNX.

Solo per .NET : le connessioni gestite a IBM MQ (WMQ_CM_CLIENT) e le connessioni non gestite a IBM MQ (WMQ_CM_CLIENT_UNMANAGED) supportano entrambe le connessioni TLS/SSL.

Concetti correlati

Supporto SSL e TLS per il client .NET non gestito

Supporto SSL e TLS per il client .NET gestito

XMSC_WMQ_SSL_KEY_REPOSITORY

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

L'ubicazione del file del database delle chiavi in cui sono memorizzati chiavi e certificati.

XMS copia la stringa, fino a un limite di 256 caratteri a byte singolo, in MQSCO.KeyRepository . IBM MQ interpreta questa stringa come un nome file, incluso il percorso completo.

Solo per .NET : le connessioni gestite a IBM MQ (WMQ_CM_CLIENT) e le connessioni non gestite a IBM MQ (WMQ_CM_CLIENT_UNMANAGED) supportano entrambe le connessioni TLS/SSL.

Per default, la proprietà non è impostata.

Concetti correlati

Supporto SSL e TLS per il client .NET non gestito

Supporto SSL e TLS per il client .NET gestito

XMSC_WMQ_SSL_KEY_RESETCOUNT

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Il KeyResetCount rappresenta il numero totale di byte non crittografati inviati e ricevuti in una conversazione SSL prima che venga rinegoziata la chiave segreta. Il numero di byte include le informazioni di controllo inviate dall'MCA.

XMS copia il valore fornito per questa proprietà in MQSCO.KeyResetCount prima di richiamare MQCONNX.

Il parametro MQSCO.KeyRestCount è disponibile solo da IBM MQ versione 6. Se IBM MQ versione 5.3, se questa proprietà è impostata, XMS non tenta di stabilire la connessione con il gestore code e genera invece un'eccezione appropriata.

Solo per .NET : le connessioni gestite a IBM MQ (WMQ_CM_CLIENT) e le connessioni non gestite a IBM MQ (WMQ_CM_CLIENT_UNMANAGED) supportano entrambe le connessioni TLS/SSL.

Il valore predefinito di questa proprietà è zero, il che significa che le chiavi segrete non vengono mai rinegoziate.

Concetti correlati

Supporto SSL e TLS per il client .NET non gestito

Supporto SSL e TLS per il client .NET gestito

XMSC_WMQ_SSL_PEER_NAME

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il nome peer da utilizzare su una connessione SSL a un gestore code.

Non esiste alcun elenco di valori canonici per questa proprietà. Invece, è necessario creare questa stringa in base alle regole per SSLPEER.

Un esempio di nome peer è:

```
"CN=John Smith, O=IBM ,OU=Test , C=GB"
```

XMS copia la stringa nella codepage a byte singolo corretta e inserisce i valori corretti in MQCD.SSLPeerNamePtr e MQCD.SSLPeerNameLength prima di chiamare MQCONNX.

Questa proprietà è rilevante solo se l'applicazione si connette a un gestore code in modalità client.

Solo per .NET : le connessioni gestite a IBM MQ (WMQ_CM_CLIENT) e le connessioni non gestite a IBM MQ (WMQ_CM_CLIENT_UNMANAGED) supportano entrambe le connessioni TLS/SSL.

Per default, la proprietà non è impostata.

Concetti correlati

Supporto SSL e TLS per il client .NET non gestito

Supporto SSL e TLS per il client .NET gestito

Riferimenti correlati

SSLPEERNAME

XMSC_WMQ_SYNCPOINT_ALL_GETS

Tipo di dati:
System.Boolean

Proprietà di:
ConnectionFactory

Indica se tutti i messaggi devono essere richiamati dalle code entro il controllo del punto di sincronizzazione.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
No	Quando le circostanze sono appropriate, il client XMS può richiamare i messaggi dalle code al di fuori del controllo del punto di sincronizzazione.
vero, true	Il client XMS deve recuperare tutti i messaggi dalle code all'interno del controllo del punto di sincronizzazione.

Il valore predefinito è false.

XMSC_WMQ_TARGET_CLIENT

Tipo di dati:
System.Int32

Proprietà di:
Destinazione

Nome utilizzato in un URI:
targetClient

Indica se i messaggi inviati alla destinazione contengono un'intestazione MQRFH2.

Se un'applicazione invia un messaggio contenente un'intestazione MQRFH2 , l'applicazione ricevente deve essere in grado di gestire l'intestazione.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
XMSC_WMQ_TARGET_DEST_JMS	I messaggi inviati alla destinazione contengono un'intestazione MQRFH2 . Specificare questo valore se l'applicazione sta inviando i messaggi a un'altra applicazione XMS , un'applicazione IBM MQ classes for JMS o un'applicazione IBM MQ nativa progettata per gestire un'intestazione MQRFH2 .
XMSC_WMQ_TARGET_DEST_MQ	I messaggi inviati alla destinazione non contengono un'intestazione MQRFH2 . Specificare questo valore se l'applicazione sta inviando i messaggi a una applicazione IBM MQ nativa che non è progettata per gestire un'intestazione MQRFH2 .

Il valore predefinito è XMSC_WMQ_TARGET_DEST_JMS.

XMSC_WMQ_TEMP_Q_PREFIX

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il prefisso utilizzato per formare il nome della coda dinamica IBM MQ creata quando l'applicazione crea una coda temporanea XMS .

Le regole per formare il prefisso sono le stesse regole per formare il contenuto del campo **DynamicQName** in un descrittore oggetto, ma l'ultimo carattere non vuoto deve essere un asterisco (*). Per default, la proprietà non è impostata. Se non è impostato, viene utilizzato il seguente valore:

- Multi `AMQ.*` su Multiplatforms
- z/OS `CSQ.*` su z/OS

Questa proprietà è rilevante solo nel dominio point-to-point.

XMSC_WMQ_TEMP_TOPIC_PREFIX

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory, Destinazione

Quando si creano argomenti temporanei, XMS genera una stringa di argomenti nel formato "TEMP/TEMPTOPICPREFIX/unique_id" oppure, se questa proprietà contiene il valore predefinito, viene generata questa stringa, "TEMP/unique_id". La specifica di un valore non vuoto consente la definizione

di specifiche code modello per la creazione delle code gestite per i sottoscrittori di argomenti temporanei creati in questa connessione.

Qualsiasi stringa non null costituita solo da caratteri validi per una stringa di argomenti IBM MQ è un valore valido per questa proprietà.

Per impostazione predefinita questa proprietà è impostata su "" (stringa vuota).

Nota: Questa proprietà è rilevante solo nel dominio di pubblicazione / sottoscrizione.

XMSC_WMQ_TEMPORA_MODEL

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il nome della coda modello IBM MQ da cui viene creata una coda dinamica quando l'applicazione crea una coda temporanea XMS .

Il valore predefinito della proprietà è SYSTEM.DEFAULT.MODEL.QUEUE.

Questa proprietà è rilevante solo nel dominio point-to-point.

XMSC_WMQ_WILDCARD_FORMATO

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory, Destinazione

Questa proprietà determina quale versione della sintassi dei caratteri jolly deve essere utilizzata.

Quando si utilizza la pubblicazione / sottoscrizione con IBM MQ '*' e '?' sono considerati caratteri jolly. Mentre '#' e '+' vengono considerati come caratteri jolly quando si utilizza la sottoscrizione di pubblicazione con IBM Integration Bus. Questa proprietà sostituisce la proprietà XMSC_WMQ_BROKER_VERSION.

I valori validi per questa proprietà sono:

XMSC_WMQ_WILDCARD_SOLO argomento
Riconosce solo i caratteri jolly a livello di argomento, ad es. '#' e '+' sono trattati come caratteri jolly. Questo valore è uguale a XMSC_WMQ_BROKER_V2.

XMSC_WMQ_WILDCARD_CHAR_ONLY
Riconosce solo i caratteri jolly, ad esempio '*' e '?' sono considerati caratteri jolly. Questo valore è uguale a XMSC_WMQ_BROKER_V1.

Per impostazione predefinita, questa proprietà è impostata su XMSC_WMQ_WILDCARD_TOPIC_ONLY.

BUS_XMSC_WPM_NOME

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory e Destinazione

Nome utilizzato in un URI:
busName

Per una factory di connessione, il nome del bus di integrazione del servizio a cui si connette l'applicazione oppure, per una destinazione, il nome del bus di integrazione del servizio in cui esiste la destinazione.

Per una destinazione che è un argomento, questa proprietà è il nome del SIB (service integration bus) in cui esiste lo spazio argomento associato. Questo spazio argomento viene specificato dalla proprietà XMSC_WPM_TOPIC_SPACE .

Se la proprietà non è impostata per una destinazione, si presuppone che la coda o lo spazio argomento associato esista nel SIB (Service Integration Bus) a cui si connette l'applicazione.

Per default, la proprietà non è impostata.

*PROTOCOLLO_COLLEGAMENTO_WPM_XMSC_*

Tipo di dati:
System.Int32

Proprietà di:
Connessione

Il protocollo di comunicazione utilizzato per la connessione al motore di messaggistica. Questa proprietà è di sola lettura.

I valori possibili della proprietà sono i seguenti:

Valore	Significato
HTTP CP WPM XMSC	La connessione utilizza HTTP su TCP/IP.
CP_XMSC_WPM_TCP	La connessione utilizza TCP/IP.

XMSC_WPM_CONNECTION_PROSSIMITÀ

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

L'impostazione di prossimità della connessione per la connessione. Questa proprietà determina la vicinanza del motore di messaggistica a cui si connette l'applicazione al server di avvio.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Impostazione di prossimità della connessione
XMSC_WPM_CONNECTION_PROXIMITY_BUS	Autobus
XMSC_WPM_CONNECTION_PROXIMITY_CLUSTER	Cluster
HOST_PROSSIMITÀ_WPM_CONNECTION_XMSC_	Host
SERVER_PROSSIMITÀ_WPM_XMSC_CONNESSIONE	Server

Il valore predefinito è XMSC_WPM_CONNECTION_PROXIMITY_BUS.

XMSC_WPM_DUR_SUB_HOME

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Nome utilizzato in un URI:
Home di durableSubscription

Il nome del motore di messaggistica dove vengono gestite tutte le sottoscrizioni durevoli per una connessione o una destinazione. I messaggi da consegnare ai sottoscrittori durevoli vengono archiviati nel punto di pubblicazione dello stesso motore di messaggistica.

È necessario specificare una home di sottoscrizione durevole per una connessione prima che un'applicazione possa creare un sottoscrittore durevole che utilizza la connessione. Qualsiasi valore specificato per una destinazione sovrascrive il valore specificato per la connessione.

Per default, la proprietà non è impostata.

Questa proprietà è rilevante solo nel dominio di pubblicazione / sottoscrizione.

Nome_HOST_WPM_XMSC

Tipo di dati:
Stringa

Proprietà di:
Connessione

Il nome host o l'indirizzo IP del sistema che contiene il motore di messaggistica a cui è connessa l'applicazione. Questa proprietà è di sola lettura.

XMSC_WPM_LOCAL_ADDRESS

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Per una connessione a un bus di integrazione del servizio, questa proprietà specifica l'interfaccia di rete locale da utilizzare oppure la porta locale o l'intervallo di porte locali da utilizzare, oppure entrambe le cose.

Il valore della proprietà è una stringa con il seguente formato:

[*nome_host*] [(*porta_inferiore*) [,*porta_alto*]]]

I significati delle variabili sono i seguenti:

nome_host
Il nome host o l'indirizzo IP dell'interfaccia di rete locale da utilizzare per la connessione.

Fornire queste informazioni è necessario solo se il sistema su cui è in esecuzione l'applicazione ha due o più interfacce di rete ed è necessario essere in grado di specificare quale interfaccia deve essere utilizzata per la connessione. Se il sistema dispone di una sola interfaccia di rete, è possibile utilizzare solo tale interfaccia. Se il sistema ha due o più interfacce di rete e non si specifica quale interfaccia deve essere utilizzata, l'interfaccia viene selezionata in modo casuale.

porta_inferiore
Il numero della porta locale da utilizzare per la connessione.

Se viene specificato anche *high_port* , *low_port* viene interpretato come il numero di porta più basso in un intervallo di numeri di porta.

porta_alto
Il numero di porta più alto in un intervallo di numeri di porta. Una delle porte nell'intervallo specificato deve essere utilizzata per la connessione.

Ecco alcuni esempi di valori validi della proprietà:

JUPITER
9.20.4.98
JUPITER (1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)

Per default, la proprietà non è impostata.

Nome_ME_WPM_XMSC

Tipo di dati:
Stringa

Proprietà di:
Connessione

Il nome del motore di messaggistica a cui è connessa l'applicazione. Questa proprietà è di sola lettura.

MAP_PERSISTENT_WPM_NON_XMSC

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Il livello di affidabilità dei messaggi non persistenti inviati utilizzando la connessione.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Livello di affidabilità
XMSC_WPM_MAPPING_AS_DESTINATION	Determinato dal livello di affidabilità predefinito specificato per la coda o lo spazio argomento nel SIB (Service Integration Bus)
XMSC_WPM_MAPPING_BEST_FACILE non_ Persistente	Massimo sforzo non persistente
XMSC_WPM_MAPPING_EXPRESS_NON_ Persistente	Espresso non persistente
XMSC_WPM_MAPPING_RELIABLE_NON_ Persistente	Affidabile non persistente
XMSC_WPM_MAPPING_RELIABLE_PERSISTENT	Affidabile persistente
XMSC_WPM_MAPPING_ASSURED_PERSISTENT	Assicurato persistente

Il valore predefinito è XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT.

MAP _WPM_PERSISTENT_XMSC

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Il livello di affidabilità dei messaggi persistenti inviati utilizzando la connessione.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Livello di affidabilità
XMSC_WPM_MAPPING_AS_DESTINATION	Determinato dal livello di affidabilità predefinito specificato per la coda o lo spazio argomento nel SIB (Service Integration Bus)

Valore valido	Livello di affidabilità
XMSC_WPM_MAPPING_BEST_FACILE non_ Persistente	Massimo sforzo non persistente
XMSC_WPM_MAPPING_EXPRESS_NON_ Persistente	Espresso non persistente
XMSC_WPM_MAPPING_RELIABLE_NON_ Persistente	Affidabile non persistente
XMSC_WPM_MAPPING_RELIABLE_PERSISTENT	Affidabile persistente
XMSC_WPM_MAPPING_ASSURED_PERSISTENT	Assicurato persistente

Il valore predefinito è XMSC_WPM_MAPPING_RELIABLE_PERSISTENT.

PORTA_WPM_XMSC

Tipo di dati:
System.Int32

Proprietà di:
Connessione

Il numero della porta di ascolto da parte del motore di messaggistica a cui è connessa l'applicazione. Questa proprietà è di sola lettura.

XMSC_WPM_PROVIDER_ENDPOINTS

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Una sequenza di uno o più indirizzi endpoint di server di avvio. Gli indirizzi endpoint sono separati da virgole.

Un server bootstrap è un server delle applicazioni responsabile della selezione del motore di messaggistica a cui si connette l'applicazione. L'indirizzo endpoint di un server di avvio ha il seguente formato:

nome_host:*numero_porta*:*nome_catena*

I significati dei componenti di un indirizzo endpoint sono i seguenti:

nome_host
Il nome host o l'indirizzo IP del sistema su cui risiede il server di avvio. Se non viene specificato alcun nome host o indirizzo IP, il valore predefinito è `localhost`.

port_number
Il numero della porta su cui il server bootstrap è in attesa delle richieste in entrata. Se non viene specificato alcun numero di porta, il valore predefinito è 7276.

nome_catena
Il nome di una catena di trasporto di avvio utilizzata dal server di avvio. I valori validi sono i seguenti:

Valore valido	Nome della catena di trasporto bootstrap
XMSC_WPM_BOOTSTRAP_HTTP	Messaggistica BootstrapTunneled
XMSC_WPM_BOOTSTRAP_HTTPS	BootstrapTunneledSecureMessaging
XMSC_WPM_BOOTSTRAP_SSL	Messaggistica BootstrapSecure

Valore valido	Nome della catena di trasporto bootstrap
XMSC_WPM_BOOTSTRAP_TCP	Messaggistica BootstrapBasic

Se non viene specificato alcun nome, il valore predefinito è XMSC_WPM_BOOTSTRAP_TCP.

Se non viene specificato alcun indirizzo endpoint, il valore predefinito è localhost:7276:BootstrapBasicMessaging.

XMSC_WPM_SSL_CIPHER_SUITE

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il nome della CipherSuite da utilizzare su una connessione TLS a un motore di messaggistica WebSphere Application Server service integration bus . Il protocollo utilizzato nella negoziazione della connessione protetta dipende dalla CipherSuite specificata.

Tabella 881. Opzioni CipherSuite per la connessione a un motore di messaggistica WebSphere Application Server service integration bus

Suite di cifratura	Protocollo utilizzato
TLS_RSA_WITH_DES_CBC_SHA	TLSv1
TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_128_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_256_CBC_SHA	TLSv1

Note:

1. Windows TLS_RSA_WITH_AES_128_CBC_SHA e TLS_RSA_WITH_AES_256_CBC_SHA CipherSuites sono supportati solo su Windows . (Ciò è dettato da GSKit.)
2. Deprecated TLS_RSA_WITH_3DES_EDE_CBC_SHA è obsoleto. Tuttavia, può essere ancora utilizzato per trasferire fino a 32 GB di dati prima che la connessione venga terminata con l'errore AMQ9288. Per evitare questo errore, è necessario evitare di utilizzare il triplo DES o abilitare la reimpostazione della chiave segreta quando si utilizza questo CipherSpec.

Non esiste alcun valore predefinito per questa proprietà. Se si desidera utilizzare SSL o TLS, è necessario specificare un valore per questa proprietà, altrimenti l'applicazione non sarà in grado di connettersi correttamente al server.

XMSC_WPM_SSL_FIPS_REQUIRED

Nota: Su AIX, Linux, and Windows, IBM MQ fornisce la conformità FIPS 140-2 tramite il modulo crittografico IBM Crypto for C (ICC) . Il certificato per questo modulo è stato spostato nello stato cronologico. I clienti devono visualizzare il certificatoIBM Crypto for C (ICC) ed essere a conoscenza di eventuali consigli forniti da NIST. Un modulo FIPS 140-3 di sostituzione è attualmente in corso e il relativo stato può essere visualizzato ricercandolo in NIST CMVP modules in process list.

IBM MQ Operator 3.2.0 e l'immagine del contenitore del gestore code 9.4.0.0 sono basati su UBI 9. La conformità FIPS 140-3 è attualmente in sospeso e il suo stato può essere visualizzato ricercando "Red Hat Enterprise Linux 9 - OpenSSL FIPS Provider" in NIST CMVP modules in process list.

Tipo di dati:
Booleano

Proprietà di:
ConnectionFactory

Il valore della proprietà determina se un'applicazione può o meno utilizzare delle suite di crittografia non conformi a FIPS. Se questa proprietà è impostata su true, solo gli algoritmi FIPS vengono utilizzati per la connessione client - server.L'impostazione del valore di questa proprietà su TRUE impedisce all'applicazione di utilizzare suite di cifratura non conformi a FIPS.

Per impostazione predefinita, la proprietà è impostata su FALSE (ossia, modalità FIPS disattivata).

XMSC_WPM_SSL_KEY_REPOSITORY

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Un percorso al file che è il file keyring contenente le chiavi pubbliche o private da utilizzare nella connessione sicura.

L'impostazione della proprietà del file keyring sul valore speciale di XMSC_WPM_SSL_MS_CERTIFICATE_STORE specifica l'utilizzo del database delle chiavi Microsoft Windows . L'utilizzo del database di chiavi Microsoft Windows , disponibile in **Pannello di controllo** > **Opzioni Internet** > **Contenuto** > **Certificati**, elimina la necessità di un database di file di chiavi separato. L'utilizzo di questa costante su Windows x64 e altre piattaforme non è consentito.

Per default, la proprietà non è impostata.

XMSC_WPM_SSL_KEYRING_LABEL

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il certificato da utilizzare quando si esegue l'autenticazione presso il server. Se non viene specificato alcun valore, viene utilizzato il certificato predefinito.

Per default, la proprietà non è impostata.

*PW ARCHIVIAZIONE CHIAVI XMSC_WPM_SSL_KEYRING_*

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

La password per il file keyring.

Questa proprietà può essere utilizzata come un'alternativa all'utilizzo di XMSC_WPM_SSL_KEYRING_STASH_FILE per configurare la password per il file keyring.

Per default, la proprietà non è impostata.

XMSC_WPM_SSL_KEYRING_STASH_FILE

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il nome di un file binario contenente la password del file del repository delle chiavi.

Questa proprietà può essere utilizzata come alternativa all'uso di XMSC_WPM_SSL_KEYRING_PW per configurare la password per il file keyring.

Per default, la proprietà non è impostata.

GRUPPO_SISTEMA_WPM_XMSC

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il nome del gruppo di destinazione dei motori di messaggistica. La natura del gruppo di destinazione è determinata dalla proprietà XMSC_WPM .

Impostare questa proprietà se si desidera limitare la ricerca di un motore di messaggistica a un sottogruppo dei motori di messaggistica nel SIB (service integration bus). Se si desidera che l'applicazione sia in grado di collegarsi a qualsiasi motore di messaggistica nel SIB (Service Integration Bus), non impostare questa proprietà.

Per default, la proprietà non è impostata.

*SIGNIFICATIVA_DESTINAZIONE_WPM_XMSC_*

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

La significatività del gruppo di destinazione dei motori di messaggistica.

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
XMSC_WPM_TARGET_SIGNIFICANCE_ Preferito	Viene selezionato un motore di messaggistica nel gruppo di destinazione, se disponibile. In caso contrario, viene selezionato un motore di messaggistica esterno al gruppo di destinazione, purché si trovi nello stesso SIB (service integration bus).
XMSC_WPM_TARGET_SIGNIFICANCE_ OBBLIGATORIO	Il motore di messaggistica selezionato deve essere nel gruppo di destinazione. Se un motore di messaggistica nel gruppo di destinazione non è disponibile, il processo di connessione non riesce.

Il valore predefinito della proprietà è XMSC_WPM_TARGET_SIGNIFICANCE_PREFERRED.

XMSC_WPM_TARGET_TRANSPORT_CHAIN

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il nome della catena di trasporto in entrata che l'applicazione deve utilizzare per connettersi a un motore di messaggistica.

Il valore della proprietà può essere il nome di qualsiasi catena di trasporto in entrata disponibile nel server delle applicazioni che ospita il motore di messaggistica. La seguente costante denominata viene fornita per una delle catene di trasporto in entrata predefinite:

Costante con nome	Nome della catena di trasporto
XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC	Messaggistica InboundBasic

Il valore predefinito della proprietà è XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC.

TIPO_SISTEMA_DI_DESTINAZIONE XMSC

Tipo di dati:
System.Int32

Proprietà di:
ConnectionFactory

Il tipo di gruppo di destinazione dei motori di messaggistica. Questa proprietà determina la natura del gruppo di destinazione identificato dalla proprietà XMSC_WPM_TARGET_GROUP .

Di seguito sono riportati i valori validi della proprietà:

Valore valido	Significato
XMSC_WPM_TARGET_TYPE_BUSMEMBER	Il nome del gruppo di destinazione è il nome di un membro bus. Il gruppo di destinazione è tutti i motori di messaggistica nel membro bus.
XMSC_WPM_TARGET_TYPE_CUSTOM	Il nome del gruppo di destinazioni è il nome di un gruppo definito dall'utente di motori di messaggistica. Il gruppo di destinazione è costituito da tutti i motori di messaggistica registrati con il gruppo definito dall'utente.
XMSC_WPM_TARGET_TYPE_ME	Il nome del gruppo di destinazione è il nome di un motore di messaggistica. Il gruppo di destinazioni è il motore di messaggistica specificato.

Per default, la proprietà non è impostata.

XMSC_WPM_TEMP_Q_PREFIX

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il prefisso utilizzato per formare il nome della coda temporanea creata nel SIB (service integration bus) quando l'applicazione crea una coda temporanea XMS . Il prefisso può contenere fino a 12 caratteri.

Il nome di una coda temporanea inizia con i caratteri "_Q" seguiti dal prefisso. Il resto del nome è costituito da caratteri generati dal sistema.

Per impostazione predefinita, la proprietà non è impostata, il che significa che il nome di una coda temporanea non ha un prefisso.

Questa proprietà è rilevante solo nel dominio point-to-point.

XMSC_WPM_TEMP_TOPIC_PREFIX

Tipo di dati:
Stringa

Proprietà di:
ConnectionFactory

Il prefisso utilizzato per formare il nome di un argomento temporaneo creato dall'applicazione. Il prefisso può contenere fino a 12 caratteri.

Il nome di un argomento temporaneo inizia con i caratteri "_T" seguiti dal prefisso. Il resto del nome è costituito da caratteri generati dal sistema.

Per impostazione predefinita, la proprietà non è impostata, il che significa che il nome di un argomento temporaneo non ha un prefisso.

Questa proprietà è rilevante solo nel dominio di pubblicazione / sottoscrizione.

SPACE XMSC_WPM_TOPIC_XX_ENCODE_CASE_ONE spazio

Tipo di dati:
Stringa

Proprietà di:
Destinazione

Nome utilizzato in un URI:
topicSpace

Il nome dello spazio argomento che contiene l'argomento. Solo una destinazione che è un argomento può avere questa proprietà.

Per impostazione predefinita, la proprietà non viene impostata, il che significa che viene utilizzato lo spazio argomento predefinito.

Questa proprietà è rilevante solo nel dominio di pubblicazione / sottoscrizione.

Managed File Transfer riferimento per lo sviluppo di applicazioni

Informazioni di riferimento che consentono di sviluppare applicazioni per Managed File Transfer.

Esempi di utilizzo del trasferimento fteCreateper avviare i programmi

È possibile utilizzare il comando **fteCreateTransfer** per specificare i programmi da eseguire prima o dopo un trasferimento.

Oltre a utilizzare **fteCreateTransfer**, esistono altri modi per richiamare un programma prima o dopo un trasferimento. Per ulteriori informazioni, consultare Specifica dei programmi da eseguire con MFT.

Tutti questi esempi utilizzano la seguente sintassi per specificare un programma:

```
[type:]commandspec[,[retrycount][,[retrywait][,successrc]]]
```

Per ulteriori informazioni su questa sintassi, consultare **fteCreateTransfer**: avvio di un nuovo trasferimento file.

Esecuzione di un programma eseguibile

Il seguente esempio specifica un programma eseguibile denominato `mycommand` e passa due argomenti, a e b, al programma.

```
mycommand(a,b)
```

Per eseguire questo programma sull'agente origine AGENT1 prima dell'avvio del trasferimento, utilizzare il seguente comando:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -presrc mycommand(a,b)
destinationSpecification sourceSpecification
```

Esecuzione e nuovo tentativo di un programma eseguibile

Il seguente esempio specifica un programma eseguibile denominato `simple`, che non utilizza alcun argomento. Per `retrycount` è specificato il valore 1 e per `retrywait`è specificato il valore 5. Questi valori indicano che il programma verrà ritentato una volta se non restituisce un codice di ritorno di esito

positivo, dopo un'attesa di cinque secondi. Non viene specificato alcun valore per `successrc`, quindi l'unico codice di ritorno corretto è il valore predefinito 0.

```
executable:simple,1,5
```

Per eseguire questo programma sull'agente origine AGENT1 una volta completato il trasferimento, utilizzare il seguente comando:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -postsrc executable:simple,1,5
destinationSpecification sourceSpecification
```

Esecuzione di uno script Ant e specifica di codici di ritorno corretti

Il seguente esempio specifica uno script Ant denominato `myscript` e trasmette due proprietà allo script. Lo script viene eseguito utilizzando il comando **fteAnt** . Il valore per `successrc` è specificato come `>2&<7&!5|0|14`, che indica che i codici di ritorno 0, 3, 4, 6 e 14 indicano l'esito positivo.

```
antscript:myscript(prop1=fred,prop2=bob),,,>2&<7&!5|0|14
```

Per eseguire questo programma sull'agente di destinazione AGENT2 prima dell'avvio del trasferimento, utilizzare il seguente comando:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -predst
"antscript:myscript(prop1=fred,prop2=bob),,,>2&<7&!5|0|14"destinationSpecification sourceSpecification
```

Esecuzione di uno script Ant e specifica delle destinazioni da chiamare

Il seguente esempio specifica uno script Ant denominato `script2` e due destinazioni, `target1` e `target2`, da chiamare. Viene inoltrata anche la proprietà `prop1` , con un valore `recmfm(F,B)`. Le parentesi, le virgole (,) e le barre retroverse (\) sono caratteri speciali nei comandi MFT e devono essere preceduti da un carattere barra retroversa (\). Windows I percorsi dei file su Windows possono essere specificati utilizzando doppie barre rovesciate (\\) come separatore o utilizzando singole barre (/). Nel seguente esempio, la virgola (,) e le parentesi sono precedute da un carattere di escape barra rovesciata (\).

```
antscript:script2(target1,target2,prop1=recmfm\(F\,B\)),,,>2&<7&!5|0|14
```

Per eseguire questo programma sull'agente di destinazione AGENT2 una volta completato il trasferimento, utilizzare il seguente comando:

```
fteCreateTransfer -sa AGENT1 -da AGENT2
-postdst "antscript:script2(target1,target2,prop1=recmfm\(F\,B\)),,,>2&<7&!5|0|14"
destinationSpecification sourceSpecification
```

Utilizzo dei metadati in uno script Ant

È possibile specificare un'attività Ant come una delle seguenti chiamate per un trasferimento:

- Pre-origine
- Post-origine
- predestinazione
- destinazione post

Quando si esegue l'attività Ant , i metadati utente del trasferimento vengono resi disponibili utilizzando variabili di ambiente. È possibile accedere a questi dati utilizzando, ad esempio, il codice seguente:

```
<property environment="environment" />
<echo>${environment.mymetadata}</echo>
```

dove `mymetadata` è il nome di alcuni metadati inseriti nel trasferimento.

Esecuzione di uno script JCL

Il seguente esempio specifica uno script JCL denominato ZOSBATCH. È stato specificato un valore di 3 per `retrycount`, un valore di 30 per `retrywait` e un valore di 0 per `successrc`. Questi valori indicano che lo script viene ritentato tre volte se non restituisce un codice di ritorno corretto pari a 0, con un'attesa di trenta secondi tra un tentativo e l'altro.

```
jcl:ZOSBATCH,3,30,0
```

dove ZOSBATCH è un membro di un PDS denominato MYSYS.JCLe il file `agent.properties` contiene la riga `commandPath=....://'MYSYS.JCL':...`

Per eseguire questo programma sull'agente origine AGENT1 una volta completato il trasferimento, utilizzare il seguente comando:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -postsrc jcl:ZOSBATCH,3,30,0
destinationSpecification sourceSpecification
```

Attività correlate

Specifica dei programmi da eseguire con MFT

Riferimenti correlati

`fteCreateTransfer`: avviare un nuovo trasferimento file

`fteAnt`: eseguire attività Ant in MFT

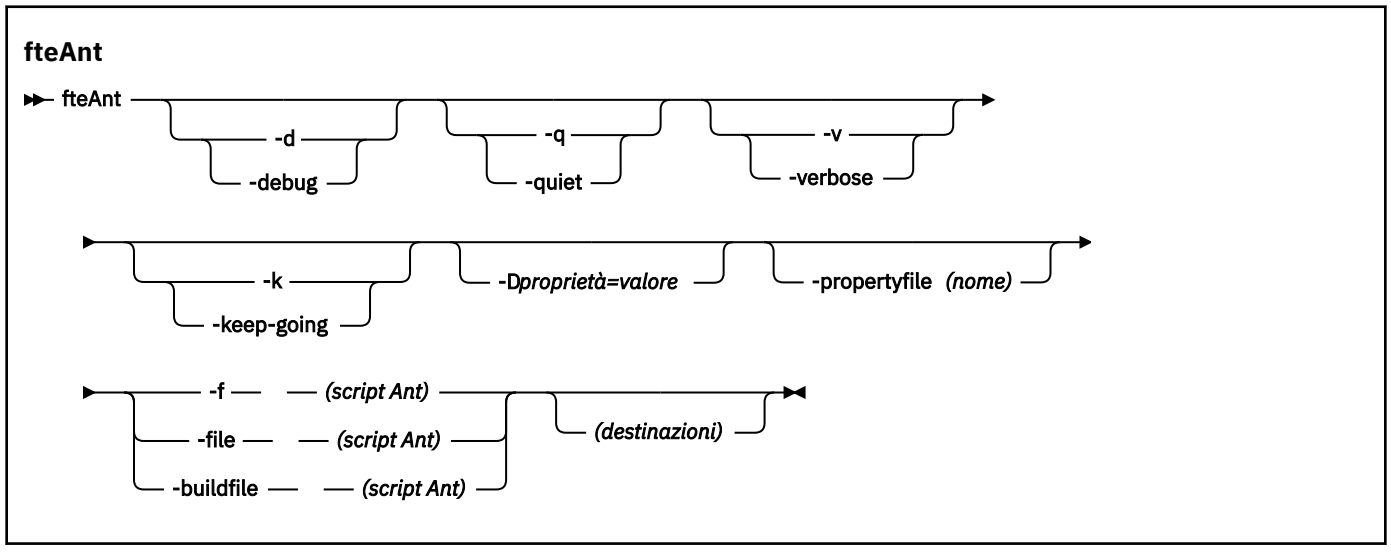
Il comando **`fteAnt`** esegue gli script Ant in un ambiente che ha attività Managed File Transfer Ant disponibili. A differenza del comando standard **ant** , **`fteAnt`** richiede la definizione di un file script.

Attività MFT Ant e parametri nidificati

Managed File Transfer fornisce una serie di Ant attività che è possibile utilizzare per accedere alle funzionalità di trasferimento file. Esiste anche una serie di parametri nidificati disponibili; questi parametri descrivono serie nidificate di elementi comuni a diverse attività Ant fornite.

La sintassi del comando **`fteAnt`** , i parametri, l'esempio di utilizzo e i codici di ritorno sono descritti nel resto di questo argomento. Per i dettagli delle attività Ant e dei parametri nidificati forniti da MFT, consultare gli argomenti secondari.

Sintassi di fteAnt



Parametri

-debug o -d
Facoltativo. Genera output di debug.

-quiet o -q
Facoltativo. Genera output minimo.

-verbose o -v
Facoltativo. Genera output dettagliato.

-keep - andare o -k
Facoltativo. Eseguire tutte le destinazioni che non dipendono da destinazioni non riuscite.

-D *proprietà=valore*
Facoltativo. Utilizzare *valore* per una determinata *proprietà*. Le proprietà impostati con **-D** hanno la precedenza rispetto a quelle impostate in un file delle proprietà.

Utilizzare la proprietà **com.ibm.wmqfte.propertyset** per specificare la serie di opzioni di configurazione utilizzate per attività Ant . Utilizzare il nome di un gestore code di coordinamento non predefinito come valore per questa proprietà. Le attività Ant utilizzano quindi la serie di opzioni di configurazione associate a questo gestore code di coordinamento non predefinito. Se non si specifica questa proprietà, viene utilizzata la serie predefinita di opzioni di configurazione basate sul gestore code di coordinamento predefinito. Se si specifica l'attributo **cmdqm** per un'attivit ... Ant , questo attributo ha la precedenza sulla serie di opzioni di configurazione specificate per il comando **fteAnt** . Questo comportamento si applica indipendentemente dal fatto che si stia utilizzando la serie predefinita di opzioni di configurazione o specificando una serie con la proprietà **com.ibm.wmqfte.propertyset** .

-propertyfile (*nome*)
Facoltativo. Caricare tutte le proprietà da un file con le proprietà **-D** che hanno la precedenza.

-f (*Script Ant*), -file (*Script Ant*) o -buildfile (*Script Ant*)
Obbligatorio. Specifica il nome dello script Ant da eseguire.

Destinazione
Facoltativo. Il nome di una o più destinazioni da eseguire dallo script Ant. Se non si specifica un valore per questo parametro, viene eseguita la destinazione predefinita per lo script.

-version
Facoltativo. Visualizza il comando Managed File Transfer e le versioni Ant .

-? o -h
Facoltativo. Visualizza la sintassi del comando.

Esempio

In questo esempio, viene eseguita la destinazione **copy** nello script Ant `fte_script.xml` e il comando scrive l'output di debug nell'output standard.

```
fteAnt -d -f fte_script.xml copy
```

Codici di ritorno

0
Comando completato correttamente.

1
Comando terminato con esito negativo.

Altri codici di ritorno di stato possono essere specificati anche da script Ant, ad esempio utilizzando l'attività di errore Ant .

Per ulteriori informazioni, consultare Errore .

Concetti correlati
Introduzione all'utilizzo degli script Ant con MFT

Attività correlate

Utilizzo di Apache Ant con MFT

Riferimenti correlati

Attività Ant di esempio per MFT

fte: attività Ant awaitoutcome

Attende il completamento di un'operazione **fte:filecopy**, **fte:filemove**o **fte:call** .

Attributi

ID

Obbligatorio. Identifica il trasferimento da cui attendere un risultato. Generalmente, si tratta di una proprietà impostata dall'attributo idProperty delle attività fte: filecopy, fte: filemoveo fte: call .

proprietà rc

Obbligatorio. Indica una proprietà in cui memorizzare il codice di ritorno dell'attività **fte:awaitoutcome** .

tempo massimo

Facoltativo. La quantità massima di tempo, in secondi, di attesa per completare l'operazione. Il timeout minimo è di un secondo. Se non si specifica un valore di timeout, l'attività **fte:awaitoutcome** attende per sempre che venga determinato il risultato dell'operazione.

Esempio

In questo esempio viene avviata una copia del file e il suo identificativo viene memorizzato nella proprietà `copy.id` . Mentre la copia è in corso, possono essere effettuate altre elaborazioni. L'istruzione **fte:awaitoutcome** viene utilizzata per attendere il completamento dell'operazione di copia. L'istruzione **fte:awaitoutcome** identifica quale operazione attendere per utilizzare l'identificativo memorizzato nella proprietà `copy.id` . **fte:awaitoutcome** memorizza un codice di ritorno che indica il esito dell'operazione di copia in una proprietà denominata `copy.result`.

```
<-- issue a file copy request -->
<fte:filecopy
src="AGENT1@QM1"
dst="AGENT2@QM2"
idproperty="copy.id"
outcome="defer">

<fte:filespec
  srcfilespec="/home/fteuser1/file.bin"
  dstdir="/home/fteuser2"/>

</fte:filecopy>

<fte:awaitoutcome id="${copy.id}" rcProperty="copy.rc"/>

<echo>Copy id=${copy.id} rc=${copy.rc}</echo>
```

Attività correlate

Utilizzo di Apache Ant con MFT

attività fte: call Ant

È possibile utilizzare l'attività **fte:call** per richiamare in remoto script e programmi.

Questa attività consente di inviare una richiesta **fte:call** a un agente. L'agent elabora questa richiesta eseguendo uno script o un programma e restituendo il risultato. I comandi da richiamare devono essere accessibili all'agent. Assicurarsi che il valore della proprietà commandPath nel file `agent.properties` includa l'ubicazione dei comandi da richiamare. Qualsiasi informazione sul percorso specificata dall'elemento nidificato del comando deve essere relativa alle ubicazioni specificate dalla proprietà commandPath . Per impostazione predefinita, commandPath è vuoto in modo che l'agente non

possa richiamare alcun comando. Per ulteriori informazioni su questa proprietà, consultare commandPath MFT property.

Per ulteriori informazioni relative al file `agent.properties` , consultare Il file MFT `agent.properties`.

Attributi

agente
Obbligatorio. Specifica l'agent a cui inoltrare la richiesta **`fte:call`** . Specificare le informazioni sull'agent nel formato: *agentname*@*qmgrname* dove *agentname* è il nome dell'agent e *qmgrname* è il nome del gestore code a cui questo agent è direttamente connesso.

cmdqm
Facoltativo. Il gestore code comandi a cui inoltrare la richiesta. Specificare queste informazioni nel formato *qmgrname*@*host*@*port*@*channel*, dove:

- *qmgrname* è il nome del gestore code
- *host* è il nome host facoltativo del sistema su cui è in esecuzione il gestore code
- *port* è il numero di porta facoltativo su cui è in ascolto il gestore code
- *channel* è il canale SVRCONN facoltativo da utilizzare

Se si omettono le informazioni *host*, *porto* *channel* per il gestore code comandi, vengono utilizzate le informazioni di connessione specificate nel file `command.properties` .

Attenzione: Se non viene specificato alcun valore per:

- Variabile *host* , viene utilizzata la modalità di bind
- variabile *port* , viene utilizzato il valore 1414
- *channel* , la variabile SYSTEM.DEF.SVRCONN .

Per ulteriori informazioni, consultare Il file MFT command.properties .

Tuttavia, non è possibile ignorare gli attributi nel mezzo, ad esempio `qmgrname@host@@channel`. È possibile avere, ad esempio, `qmgrname@host`, `qmgrname@host@porto` `qmgrname@hostport@@channel`.

MFT suddivide l'attributo fornito utilizzando il delimitatore `@` . A seconda del numero di token trovati, utilizza il primo token come *qmgrname*, il secondo come *host*, il terzo come *port* e infine *channel*.

Per ulteriori informazioni, vedere File MFT command.properties.

È possibile utilizzare la proprietà **`com.ibm.wmqfte.propertySet`** per specificare quale file `command.properties` utilizzare. Per ulteriori informazioni, consultare com.ibm.wmqfte.propertySet.

Se non si utilizza l'attributo cmdqm , per impostazione predefinita l'attività utilizza la proprietà `com.ibm.wmqfte.ant.commandQueueManager` , se è impostata. Se la proprietà `com.ibm.wmqfte.ant.commandQueueManager` non è impostata, viene tentata una connessione al gestore code predefinito, definito nel file `command.properties` . Il formato della proprietà `com.ibm.wmqfte.ant.commandQueueManager` è lo stesso dell'attributo cmdqm , ovvero `qmgrname@host@port@channel`.

idproperty
Facoltativo a meno che non sia stato specificato un `outcome` di `defer`. Specifica il nome di una proprietà a cui assegnare l'ID trasferimento. Gli identificativi di trasferimento vengono generati nel punto in cui viene inoltrata una richiesta di trasferimento ed è possibile utilizzare gli identificativi di trasferimento per tracciare l'avanzamento di un trasferimento, diagnosticare i problemi con un trasferimento e annullare un trasferimento.

Non è possibile specificare questa proprietà se è stata specificata anche una proprietà `outcome` di `ignore`. Tuttavia, è necessario specificare `idproperty` se è stata specificata anche una proprietà `outcome` di `defer`.

jobName
Facoltativo. Assegna un nome lavoro alla richiesta **fte:call** . È possibile utilizzare i nomi lavoro per creare gruppi logici di trasferimenti. Utilizzare l'attività “fte: uuid Ant attività” a pagina 2169 per generare nomi lavoro pseudo - univoci. Se non si utilizza l'attributo jobname , per impostazione predefinita l'attività utilizza il valore della proprietà `com.ibm.wmqfte.ant.jobName` , se questa proprietà è impostata. Se non si imposta questa proprietà, nessun nome lavoro è associato alla richiesta **fte:call** .

origuser
Facoltativo. Specifica l'identificativo utente di origine da associare alla richiesta **fte:call** . Se non si utilizza l'attributo origuser , per impostazione predefinita l'attività utilizza l'ID utente utilizzato per eseguire lo script Ant.

risultato
Facoltativo. Determina se l'attività attende il completamento dell'operazione **fte:call** prima di restituire il controllo allo script Ant . Specifica una delle seguenti opzioni:

attendere
L'attività attende il completamento dell'operazione **fte:call** prima di ritornare. Quando viene specificato un `outcome` di `await` , l'attributo `idproperty` è facoltativo.

differire
L'attività viene restituita non appena la richiesta **fte:call** è stata inoltrata e presuppone che il risultato dell'operazione di chiamata venga trattato in un secondo momento utilizzando le attività awaitoutcome o ignoreoutcome . Quando viene specificato un `outcome` di `defer` , l'attributo `idproperty` è obbligatorio.

ignorare
Se il risultato dell'operazione **fte:call** non è importante, è possibile specificare un valore di `ignore`. L'attività viene quindi restituita non appena la richiesta **fte:call** viene inoltrata, senza allocare alcuna risorsa per tenere traccia del risultato del comando. Quando viene specificata una `outcome` di `ignore` , non è possibile specificare l'attributo `idproperty` .

Se non si specifica l'attributo outcome , per impostazione predefinita l'attività utilizza il valore `await`.

proprietà rc
Facoltativo. Specifica il nome di una proprietà a cui assegnare il codice risultato della richiesta **fte:call** . Il codice risultato riflette il risultato generale della richiesta **fte:call** .

Non è possibile specificare questa proprietà se è stata specificata anche una proprietà `outcome` di `ignore` o `defer`. Tuttavia, è necessario specificare `rcproperty` se è stato specificato un risultato di `await`.

Parametri specificati come elementi nidificati

fte: comando
Specifica il comando che deve essere richiamato dall'agent. È possibile associare un solo elemento `fte:command` a una determinata operazione**fte:call** . Il comando da richiamare deve trovarsi nel percorso specificato dalla proprietà commandPath del file `agent.properties` dell'agente.

fte: metadati
È possibile specificare metadati da associare all'operazione di chiamata. Questi metadati vengono registrati nei messaggi di log generati dall'operazione di chiamata. È possibile associare solo un singolo blocco di metadati a un determinato elemento di trasferimento; tuttavia, questo blocco può contenere molte parti di metadati.

Esempio

Questo esempio mostra come richiamare un comando in AGENT1 in esecuzione sul gestore code QM1. Il comando da richiamare è lo script `command.sh` e lo script viene richiamato con un singolo argomento

di xyz. Il comando command.sh si trova nel percorso specificato dalla proprietà commandPath nel file agent.properties dell'agent.

```
<fte:call cmdqm="QM0@localhost@1414@SYSTEM.DEF.SVRCONN"
            agent="AGENT1@QM1"
            rcproperty="call.rc"
            origuser="bob"
            jobname="${job.id}">

    <fte:command command="command.sh" successrc="1" retrycount="5" retrywait="30">
        <fte:arg value="xyz"/>
    </fte:command>

    <fte:metadata>
        <fte:entry name="org.foo.accountName" value="BDG3R"/>
    </fte:metadata>

</fte:call>
```

Attività correlate

Utilizzo di Apache Ant con MFT

attività fte: cancel Ant

Annulla un trasferimento gestito o una chiamata gestita Managed File Transfer . Un trasferimento gestito potrebbe essere stato creato utilizzando le attività **fte:filecopy** o **fte:filemove** . Una chiamata gestita potrebbe essere stata creata utilizzando l'attività **fte:call** .

Attributi

agente

Obbligatorio. Specifica l'agent a cui inoltrare la richiesta **fte:cancel** . Il valore è nel seguente formato: *agentname*@*qmgrname* dove *agentname* è il nome dell'agent e *qmgrname* è il nome del gestore code a cui questo agent è direttamente connesso.

cmdqm

Facoltativo. Il gestore code comandi a cui inoltrare la richiesta. Specificare queste informazioni nel formato *qmgrname*@*host*@*port*@*channel*, dove:

- *qmgrname* è il nome del gestore code
- *host* è il nome host facoltativo del sistema su cui è in esecuzione il gestore code
- *port* è il numero di porta facoltativo su cui è in ascolto il gestore code
- *channel* è il canale SVRCONN facoltativo da utilizzare

Se si omettono le informazioni *host*, *port*o *channel* per il gestore code comandi, vengono utilizzate le informazioni di connessione specificate nel file command.properties .

Attenzione: Se non viene specificato alcun valore per:

- Variabile *host* , viene utilizzata la modalità di bind
- variabile *port* , viene utilizzato il valore 1414
- *channel* , la variabile SYSTEM.DEF.SVRCONN .

Per ulteriori informazioni, consultare Il file MFT command.properties .

Tuttavia, non è possibile ignorare gli attributi nel mezzo, ad esempio qmgrname@host@@channel. È possibile avere, ad esempio, qmgrname@host, qmgrname@host@porto qmgrname@hostport@@channel.

MFT suddivide l'attributo fornito utilizzando il delimitatore @ . A seconda del numero di token trovati, utilizza il primo token come *qmgrname*, il secondo come *host*, il terzo come *port* e infine *channel*.

Per ulteriori informazioni, vedere File MFT command.properties.

È possibile utilizzare la proprietà **com.ibm.wmqfte.propertySet** per specificare quale file `command.properties` utilizzare. Per ulteriori informazioni, consultare com.ibm.wmqfte.propertySet.

Se non si utilizza l'attributo cmdqm , per impostazione predefinita l'attività utilizza la proprietà `com.ibm.wmqfte.ant.commandQueueManager` , se è impostata. Se la proprietà `com.ibm.wmqfte.ant.commandQueueManager` non è impostata, viene tentata una connessione al gestore code predefinito, definito nel file `command.properties` . Il formato della proprietà `com.ibm.wmqfte.ant.commandQueueManager` è lo stesso dell'attributo cmdqm , ovvero `qmgrname@host@port@channel`.

ID

Obbligatorio. Specifica l'identificativo del trasferimento da annullare. Gli identificativi di trasferimento vengono generati nel punto in cui una richiesta di trasferimento viene inoltrata sia dalle attività fte: filecopy che fte: filemove .

origuser

Facoltativo. Specifica l'identificativo utente di origine da associare alla richiesta **cancel** . Se l'attributo origuser non viene utilizzato, per impostazione predefinita l'attività utilizza l'ID utente utilizzato per eseguire lo script Ant .

Esempio

L'esempio invia una richiesta **fte:cancel** al gestore code comandi `qm0`. La richiesta **fte:cancel** è indirizzata a `agent1` sul gestore code `qm1` per l'identificativo di trasferimento popolato dalla variabile `transfer.id` . La richiesta viene eseguita utilizzando l'ID utente "bob" .

```
<fte:cancel cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
            agent="agent1@qm1"
            id="${transfer.id}"
            origuser="bob"/>
```

Attività correlate

Utilizzo di Apache Ant con MFT

attività fte: filecopy Ant

L'attività **fte:filecopy** copia i file tra agent Managed File Transfer . Il file non viene eliminato dall'agent di origine.

Attributi

cmdqm

Facoltativo. Il gestore code comandi a cui inoltrare la richiesta. Specificare queste informazioni nel formato *qmgrname*@*host*@*port*@*channel*, dove:

- *qmgrname* è il nome del gestore code
- *host* è il nome host facoltativo del sistema su cui è in esecuzione il gestore code
- *port* è il numero di porta facoltativo su cui è in ascolto il gestore code
- *channel* è il canale SVRCONN facoltativo da utilizzare

Se si omettono le informazioni *host*, *porto* *channel* per il gestore code comandi, vengono utilizzate le informazioni di connessione specificate nel file `command.properties` .

Attenzione: Se non viene specificato alcun valore per:

- Variabile *host* , viene utilizzata la modalità di bind
- variabile *port* , viene utilizzato il valore 1414
- *channel* , la variabile SYSTEM.DEF.SVRCONN .

Per ulteriori informazioni, consultare Il file MFT command.properties .

Tuttavia, non è possibile ignorare gli attributi nel mezzo, ad esempio `qmgrname@host@@channel`. È possibile avere, ad esempio, `qmgrname@host`, `qmgrname@host@porto` `qmgrname@hostport@@channel`.

MFT suddivide l'attributo fornito utilizzando il delimitatore `@` . A seconda del numero di token trovati, utilizza il primo token come *qmgrname*, il secondo come *host*, il terzo come *port* e infine *channel*.

Per ulteriori informazioni, vedere File MFT command.properties.

È possibile utilizzare la proprietà **`com.ibm.wmqfte.propertySet`** per specificare quale file `command.properties` utilizzare. Per ulteriori informazioni, consultare com.ibm.wmqfte.propertySet.

Se non si utilizza l'attributo cmdqm , per impostazione predefinita l'attività utilizza la proprietà `com.ibm.wmqfte.ant.commandQueueManager` , se è impostata. Se la proprietà `com.ibm.wmqfte.ant.commandQueueManager` non è impostata, viene tentata una connessione al gestore code predefinito, definito nel file `command.properties` . Il formato della proprietà `com.ibm.wmqfte.ant.commandQueueManager` è lo stesso dell'attributo cmdqm , ovvero `qmgrname@host@port@channel`.

DST
Obbligatorio. Specifica l'agent di destinazione per l'operazione di copia. Specificare queste informazioni nel modulo: `agentname@qmgrname` dove `agentname` è il nome dell'agent di destinazione e `qmgrname` è il nome del gestore code a cui questo agent è direttamente connesso.

idproperty
Facoltativo a meno che non sia stato specificato un `outcome` di `defer`. Specifica il nome di una proprietà a cui assegnare l'ID trasferimento. Gli identificativi di trasferimento vengono generati nel punto in cui viene inoltrata una richiesta di trasferimento ed è possibile utilizzare gli identificativi di trasferimento per tracciare l'avanzamento di un trasferimento, diagnosticare i problemi con un trasferimento e annullare un trasferimento.

Non è possibile specificare questa proprietà se è stata specificata anche una proprietà `outcome` di `ignore`. Tuttavia, è necessario specificare `idproperty` se è stata specificata anche una proprietà `outcome` di `defer`.

jobName
Facoltativo. Assegna un nome lavoro alla richiesta di copia. È possibile utilizzare i nomi lavoro per creare gruppi logici di trasferimenti. Utilizzare l'attività “fte: uuid Ant attività” a pagina 2169 per generare nomi lavoro pseudo - univoci. Se non si utilizza l'attributo jobname , per impostazione predefinita l'attività utilizza il valore della proprietà `com.ibm.wmqfte.ant.jobName` , se questa proprietà è impostata. Se non si imposta questa proprietà, nessun nome lavoro viene associato alla richiesta di copia.

origuser
Facoltativo. Specifica l'identificativo utente di origine da associare alla richiesta di copia. Se non si utilizza l'attributo origuser , per impostazione predefinita l'attività utilizza l'ID utente utilizzato per eseguire lo script Ant .

risultato
Facoltativo. Determina se l'attività attende il completamento dell'operazione di copia prima di restituire il controllo allo script Ant . Specifica una delle seguenti opzioni:

attendere
L'attività attende il completamento dell'operazione di copia prima di ritornare. Quando viene specificato un `outcome` di `await` , l'attributo `idproperty` è facoltativo.

differire
L'attività ritorna non appena la richiesta di copia è stata inoltrata e presuppone che il risultato dell'operazione di copia venga gestito in un secondo momento utilizzando le attività “fte: attività Ant awaitoutcome” a pagina 2156 o “fte: attività ignoreoutcome Ant” a pagina 2168 . Quando viene specificato un `outcome` di `defer` , l'attributo `idproperty` è obbligatorio.

ignorare
Se il risultato dell'operazione di copia non è importante, è possibile specificare un valore di `ignore`. L'attività viene quindi restituita non appena la richiesta di copia è stata inoltrata, senza

allocare alcuna risorsa per tenere traccia del risultato del trasferimento. Quando viene specificata una `outcome` di `ignore` , non è possibile specificare l'attributo `idproperty` .

Se non si specifica l'attributo outcome , per impostazione predefinita l'attività utilizza il valore `await`.

priorità
Facoltativo. Specifica la priorità da associare alla richiesta di copia. In generale, le richieste di trasferimento con priorità più alta hanno la precedenza sulle richieste con priorità più bassa. Il valore della priorità deve essere compreso tra 0 e 9 (inclusi). Un valore di priorità pari a 0 è la priorità più bassa e un valore pari a 9 è la priorità più alta. Se non si specifica l'attributo `priority` , il trasferimento assume come valore predefinito una priorità 0.

proprietà rc
Facoltativo. Specifica il nome di una proprietà a cui assegnare il codice risultato della richiesta di copia. Il codice risultato riflette il risultato generale della richiesta di copia.

Non è possibile specificare questa proprietà se è stata specificata anche una proprietà `outcome` di `ignore` o `defer`. Tuttavia, è necessario specificare `rcproperty` se si specifica un risultato di `await`.

Timeout transferRecovery
Facoltativo. Imposta la quantità di tempo, in secondi, durante la quale l'agente di origine tenta di ripristinare un trasferimento file bloccato. Specifica una delle seguenti opzioni:

-1
L'agent continua a tentare di recuperare il trasferimento in stallo fino al completamento del trasferimento. L'uso di questa opzione equivale al comportamento predefinito dell'agente quando la proprietà non è impostata.

0
L'agent arresta il trasferimento file non appena avvia il ripristino.

>0
L'agent continua a tentare di recuperare il trasferimento in stallo per il periodo di tempo in secondi come impostato dal valore intero positivo specificato. Ad esempio:

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm1" dst="agent2@qm2"
              rcproperty="copy.result" transferRecoveryTimeout="21600">

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/
file.bin"/>

</fte:filecopy>
```

indica che l'agent continua a tentare di recuperare il trasferimento per 6 ore da quando entra nel ripristino. Il valore massimo per questo attributo è 999999999.

La specifica del valore di timeout di ripristino del trasferimento in questo modo lo imposta in base al trasferimento. Per impostare un valore globale per tutti i trasferimenti in una rete Managed File Transfer , è possibile aggiungere una proprietà alle Proprietà di timeout di recupero trasferimento. Per ulteriori informazioni, consultare Opzione di timeout per i trasferimenti in ripristino.

src
Obbligatorio. Specifica l'agente di origine per l'operazione di copia. Specificare queste informazioni nel modulo: *agentname*@*qmgrname* dove *agentname* è il nome dell'agent di origine e *qmgrname* è il nome del gestore code a cui questo agent è direttamente connesso.

Parametri specificati come elementi nidificati

fte: filespec
Obbligatorio. È necessaria almeno una specifica file che identifichi i file da copiare. Se necessario, è possibile specificare più di una specifica file. Per ulteriori informazioni, consultare “fte: filespec Ant elemento nidificato” a pagina 2170.

fte: metadati
È possibile specificare i metadati da associare all'operazione di copia. Questi metadati vengono trasferiti e registrati nei messaggi di log generati dal trasferimento. È possibile associare solo un

singolo blocco di metadati a un determinato elemento di trasferimento; tuttavia, questo blocco può contenere molte parti di metadati. Per ulteriori informazioni, consultare l'argomento fte: metadata .

fte: presrc
Specifica un richiamo di programma da eseguire sull'agente di origine prima dell'avvio del trasferimento. È possibile solo associare un singolo elemento `fte:presrc` ad un determinato trasferimento. Per ulteriori informazioni, consultare l'argomento Richiamo del programma .

fte: predst
Specifica un richiamo di programma da eseguire sull'agente di destinazione prima dell'avvio del trasferimento. È possibile solo associare un singolo elemento `fte:predst` ad un determinato trasferimento. Per ulteriori informazioni, consultare l'argomento Richiamo del programma .

fte: postsrc
Specifica un richiamo di programma che deve essere eseguito sull'agente di origine una volta completato il trasferimento. È possibile solo associare un singolo elemento `fte:postsrc` ad un determinato trasferimento. Per ulteriori informazioni, consultare l'argomento Richiamo del programma .

fte: postdst
Specifica un richiamo di programma che deve essere eseguito sull'agente di destinazione dopo il completamento del trasferimento. È possibile solo associare un singolo elemento `fte:postdst` ad un determinato trasferimento. Per ulteriori informazioni, consultare l'argomento Richiamo del programma .

Se fte:pre - rc, fte:predst, fte:postsrc, fte:postdst e exit non restituiscono uno stato di esito positivo, le regole sono le seguenti nell'ordine specificato:

1. Eseguire le uscite di avvio origine. Se le uscite di avvio dell'origine hanno esito negativo, il trasferimento ha esito negativo e non viene eseguito altro.
2. Eseguire la chiamata pre - origine (quando presente). Se la chiamata di pre - origine ha esito negativo, il trasferimento ha esito negativo e non viene eseguito altro.
3. Eseguire le uscite iniziali di destinazione. Se le uscite di avvio della destinazione hanno esito negativo, il trasferimento ha esito negativo e non viene eseguito nulla di più.
4. Eseguire la chiamata pre - destinazione (quando presente). Se la chiamata di pre - destinazione ha esito negativo, il trasferimento ha esito negativo e non viene eseguito altro.
5. Eseguire i trasferimenti file.
6. Eseguire le uscite di fine destinazione. Non esiste uno stato di errore per queste uscite.
7. Se il trasferimento ha esito positivo (se il trasferimento di alcuni file ha esito positivo, è considerato riuscito), eseguire la chiamata post - destinazione (se presente). Se la chiamata post - destinazione ha esito negativo, il trasferimento ha esito negativo.
8. Eseguire le uscite di fine origine. Non esiste uno stato di errore per queste uscite.
9. Se il trasferimento ha esito positivo, eseguire la chiamata post - origine (se presente). Se la chiamata di post - origine ha esito negativo, il trasferimento ha esito negativo.

Esempi

Questo esempio mostra un trasferimento file di base tra `agent1` e `agent2`. Il comando per avviare il trasferimento file viene inviato a un gestore code denominato `qm0`, utilizzando una connessione in modalità di trasporto client. Il risultato dell'operazione di trasferimento file viene assegnato alla proprietà denominata `copy.result`.

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm1" dst="agent2@qm2"
              rcproperty="copy.result">

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

</fte:filecopy>
```

Questo esempio mostra lo stesso trasferimento di file, ma con l'aggiunta di metadati e l'avvio di un programma nell'agent di origine dopo il completamento del trasferimento.

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm"1 dst="agent2@qm2"
              rcproperty="copy.result">


    <fte:metadata>
            <fte:entry name="org.example.departId" value="ACCOUNTS"/>
            <fte:entry name="org.example.batchGroup" value="A1"/>
    </fte:metadata>

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

    <fte:postsrc command="/home/fteuser2/scripts/post.sh" successrc="1" >
            <fte:arg value="/home/fteuser2/file.bin"/>
  </fte:postsrc>
</fte:filecopy>
```

Concetti correlati

Opzione di timeout per i trasferimenti file in recupero

Attività correlate

Utilizzo di Apache Ant con MFT

fte: filemove Ant attività

L'attività **fte:filemove** sposta i file tra agent Managed File Transfer . Quando un file è stato trasferito correttamente dall'agent di origine all'agent di destinazione, il file viene eliminato dall'agent di origine.

Attributi

cmdqm

Facoltativo. Il gestore code comandi a cui inoltrare la richiesta. Specificare queste informazioni nel formato *qmgrname*@*host*@*port*@*channel*, dove:

- *qmgrname* è il nome del gestore code
- *host* è il nome host facoltativo del sistema su cui è in esecuzione il gestore code
- *port* è il numero di porta facoltativo su cui è in ascolto il gestore code
- *channel* è il canale SVRCONN facoltativo da utilizzare

Se si omettono le informazioni *host*, *porto* *channel* per il gestore code comandi, vengono utilizzate le informazioni di connessione specificate nel file command.properties .

Attenzione: Se non viene specificato alcun valore per:

- Variabile *host* , viene utilizzata la modalità di bind
- variabile *port* , viene utilizzato il valore 1414
- *channel* , la variabile SYSTEM.DEF.SVRCONN .

Per ulteriori informazioni, consultare Il file MFT command.properties .

Tuttavia, non è possibile ignorare gli attributi nel mezzo, ad esempio qmgrname@host@@channel. È possibile avere, ad esempio, qmgrname@host, qmgrname@host@porto qmgrname@hostport@@channel.

MFT suddivide l'attributo fornito utilizzando il delimitatore @ . A seconda del numero di token trovati, utilizza il primo token come *qmgrname*, il secondo come *host*, il terzo come *port* e infine *channel*.

Per ulteriori informazioni, vedere File MFT command.properties.

È possibile utilizzare la proprietà **com.ibm.wmqfte.propertySet** per specificare quale file command.properties utilizzare. Per ulteriori informazioni, consultare com.ibm.wmqfte.propertySet.

Se non si utilizza l'attributo cmdqm , per impostazione predefinita l'attività utilizza la proprietà `com.ibm.wmqfte.ant.commandQueueManager` , se è impostata. Se la proprietà `com.ibm.wmqfte.ant.commandQueueManager` non è impostata, viene tentata una connessione al gestore code predefinito, definito nel file `command.properties` . Il formato della proprietà `com.ibm.wmqfte.ant.commandQueueManager` è lo stesso dell'attributo cmdqm , ovvero `qmgrname@host@port@channel`.

DST
Obbligatorio. Specifica l'agent di destinazione per l'operazione di copia. Specificare queste informazioni nel modulo: *agentname@qmgrname* dove *agentname* è il nome dell'agent di destinazione e *qmgrname* è il nome del gestore code a cui questo agent è direttamente connesso.

idproperty
Facoltativo a meno che non sia stato specificato un `outcome` di `defer`. Specifica il nome di una proprietà a cui assegnare l'ID trasferimento. Gli identificativi di trasferimento vengono generati nel punto in cui viene inoltrata una richiesta di trasferimento ed è possibile utilizzare gli identificativi di trasferimento per tracciare l'avanzamento di un trasferimento, diagnosticare i problemi con un trasferimento e annullare un trasferimento.

Non è possibile specificare questa proprietà se è stata specificata anche una proprietà `outcome` di `ignore`. Tuttavia, è necessario specificare `idproperty` se è stata specificata anche una proprietà `outcome` di `defer`.

jobName
Facoltativo. Assegna un nome lavoro alla richiesta di spostamento. È possibile utilizzare i nomi lavoro per creare gruppi logici di trasferimenti. Utilizzare l'attività fte: uuid per creare nomi lavoro pseudo - univoci. Se non si utilizza l'attributo jobname , per impostazione predefinita l'attività utilizza il valore della proprietà `com.ibm.wmqfte.ant.jobName` , se questa proprietà è impostata. Se non si imposta questa proprietà, nessun nome lavoro viene associato alla richiesta di spostamento.

origuser
Facoltativo. Specifica l'identificativo utente di origine da associare alla richiesta di spostamento. Se non si utilizza l'attributo origuser , per impostazione predefinita l'attività utilizza l'ID utente utilizzato per eseguire lo script Ant .

risultato
Facoltativo. Determina se l'attività attende il completamento dell'operazione di spostamento prima di restituire il controllo allo script Ant . Specifica una delle seguenti opzioni:

attendere
L'attività attende il completamento dell'operazione di spostamento prima di ritornare. Quando viene specificato un `outcome` di `await` , l'attributo `idproperty` è facoltativo.

differire
L'attività ritorna non appena la richiesta di spostamento è stata inoltrata e presuppone che il risultato dell'operazione di spostamento venga gestito successivamente utilizzando l'attività “fte: attività Ant awaitoutcome” a pagina 2156 o “fte: attività ignoreoutcome Ant” a pagina 2168 . Quando viene specificato un `outcome` di `defer` , l'attributo `idproperty` è obbligatorio.

ignorare
Se il risultato dell'operazione di spostamento non è importante, è possibile specificare un valore di `ignore`. L'attività ritorna quindi non appena la richiesta di spostamento è stata inoltrata, senza assegnare alcuna risorsa per tenere traccia del risultato del trasferimento. Quando viene specificata una `outcome` di `ignore` , non è possibile specificare l'attributo `idproperty` .

Se non si specifica l'attributo outcome , per impostazione predefinita l'attività utilizza il valore `await`.

priorità
Facoltativo. Specifica la priorità da associare alla richiesta di spostamento. In generale, le richieste di trasferimento con priorità più alta hanno la precedenza sulle richieste con priorità più bassa. Il valore della priorità deve essere compreso tra 0 e 9 (inclusi). Un valore di priorità pari a 0 è la priorità più bassa e un valore pari a 9 è la priorità più alta. Se non si specifica l'attributo `priority` , il trasferimento assume come valore predefinito una priorità 0.

proprietà rc
Facoltativo. Specifica il nome di una proprietà a cui assegnare il codice risultato della richiesta di spostamento. Il codice risultato riflette il risultato generale della richiesta di spostamento.

Non è possibile specificare questa proprietà se è stata specificata anche una proprietà `outcome` di `ignore` o `defer`. Tuttavia, è necessario specificare `rcproperty` se è stato specificato un risultato di `await`.

Timeout transferRecovery
Facoltativo. Imposta la quantità di tempo, in secondi, durante la quale l'agente di origine tenta di ripristinare un trasferimento file bloccato. Specifica una delle seguenti opzioni:

`-1`
L'agent continua a tentare di recuperare il trasferimento in stallo fino al completamento del trasferimento. L'uso di questa opzione equivale al comportamento predefinito dell'agente quando la proprietà non è impostata.

`0`
L'agent arresta il trasferimento file non appena avvia il ripristino.

`>0`
L'agent continua a tentare di recuperare il trasferimento in stallo per il periodo di tempo in secondi come impostato dal valore intero positivo specificato. Ad esempio:

```
<fte:filemove cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src=agent1@qm1 dst="agent2@qm2"
              rcproperty="move.result" transferRecoveryTimeout="21600">

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/
file.bin"/>

</fte:filemove
```

indica che l'agent continua a tentare di recuperare il trasferimento per 6 ore da quando entra nel ripristino. Il valore massimo per questo attributo è 999999999.

La specifica del valore di timeout di ripristino del trasferimento in questo modo lo imposta in base al trasferimento. Per impostare un valore globale per tutti i trasferimenti in una rete Managed File Transfer , è possibile aggiungere una proprietà alle Proprietà di timeout di recupero trasferimento. Per ulteriori informazioni, consultare Opzione di timeout per i trasferimenti in ripristino.

src
Obbligatorio. Specifica l'agent di origine per l'operazione di spostamento. Specificare queste informazioni nel seguente formato: *agentname*`@`*qmgrname* dove *agentname* è il nome dell'agent di origine e *qmgrname* è il nome del gestore code a cui questo agent è direttamente connesso.

Parametri specificati come elementi nidificati

fte: filespec
Obbligatorio. È necessario specificare almeno una specifica file che identifichi i file da spostare. Se necessario, è possibile specificare più di una specifica file. Per ulteriori informazioni, consultare “fte: filespec Ant elemento nidificato” a pagina 2170.

fte: metadati
Facoltativo. È possibile specificare i metadati da associare all'operazione di spostamento file. Questi metadati vengono trasferiti e registrati nei messaggi di log generati dal trasferimento. È possibile associare solo un singolo blocco di metadati a un determinato elemento di trasferimento; tuttavia, questo blocco può contenere molte parti di metadati. Per ulteriori informazioni, consultare l'argomento fte: metadata .

fte: presrc
Facoltativo. Specifica un richiamo di programma da eseguire sull'agente di origine prima dell'avvio del trasferimento. È possibile solo associare un singolo elemento `fte:presrc` ad un determinato trasferimento. Per ulteriori informazioni, consultare l'argomento Richiamo del programma .

fte: predst
Facoltativo. Specifica un richiamo di programma da eseguire sull'agente di destinazione prima dell'avvio del trasferimento. È possibile solo associare un singolo elemento `fte:predst` ad un determinato trasferimento. Per ulteriori informazioni, consultare l'argomento Richiamo del programma .

fte: postsrc
Facoltativo. Specifica un richiamo di programma che deve essere eseguito sull'agente di origine una volta completato il trasferimento. È possibile solo associare un singolo elemento `fte:postsrc` ad un determinato trasferimento. Per ulteriori informazioni, consultare l'argomento Richiamo del programma .

fte: postdst
Facoltativo. Specifica un richiamo di programma che deve essere eseguito sull'agente di destinazione dopo il completamento del trasferimento. È possibile solo associare un singolo elemento `fte:postdst` ad un determinato trasferimento. Per ulteriori informazioni, consultare l'argomento Richiamo del programma .

Se fte:pre - rc, fte:predst, fte:postsrc, fte:postdst e exit non restituiscono uno stato di esito positivo, le regole sono le seguenti nell'ordine specificato:

1. Eseguire le uscite di avvio origine. Se le uscite di avvio dell'origine hanno esito negativo, il trasferimento ha esito negativo e non viene eseguito altro.
2. Eseguire la chiamata pre - origine (quando presente). Se la chiamata di pre - origine ha esito negativo, il trasferimento ha esito negativo e non viene eseguito altro.
3. Eseguire le uscite iniziali di destinazione. Se le uscite di avvio della destinazione hanno esito negativo, il trasferimento ha esito negativo e non viene eseguito nulla di più.
4. Eseguire la chiamata pre - destinazione (quando presente). Se la chiamata di pre - destinazione ha esito negativo, il trasferimento ha esito negativo e non viene eseguito altro.
5. Eseguire i trasferimenti file.
6. Eseguire le uscite di fine destinazione. Non esiste uno stato di errore per queste uscite.
7. Se il trasferimento ha esito positivo (se alcuni file vengono trasferiti correttamente, il trasferimento viene considerato riuscito), eseguire la chiamata di post - destinazione (se presente). Se la chiamata post - destinazione ha esito negativo, il trasferimento ha esito negativo.
8. Eseguire le uscite di fine origine. Non esiste uno stato di errore per queste uscite.
9. Se il trasferimento ha esito positivo, eseguire la chiamata post - origine (se presente). Se la chiamata di post - origine ha esito negativo, il trasferimento ha esito negativo.

Esempi

Questo esempio mostra uno spostamento di file di base tra `agent1` e `agent2`. Il comando per avviare lo spostamento del file viene inviato a un gestore code denominato `qm0`, utilizzando una connessione in modo trasporto client. Il risultato dell'operazione di trasferimento file viene assegnato alla proprietà denominata `move.result`.

```
<fte:filemove cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm1" dst="agent2@qm2"
              rcproperty="move.result">

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

</fte:filemove>
```

Concetti correlati

Opzione di timeout per i trasferimenti file in recupero

Attività correlate

Utilizzo di Apache Ant con MFT

fte: attività ignoreoutcome Ant

Ignorare il risultato di un comando **fte:filecopy**, **fte:filemove**o **fte:call** . Quando si specifica un'attività **fte:filecopy**, **fte:filemove**o **fte:call** per avere un risultato di `defer`, l'attività Ant assegna le risorse per tenere traccia di questo risultato. Se non si è più interessati al risultato, è possibile utilizzare l'attività **fte:ignoreoutcome** per liberare tali risorse.

Attributi

ID

Obbligatorio. Identifica il risultato che non è più di interesse. Generalmente, questo identificativo viene specificato utilizzando una proprietà impostata utilizzando l'attributo `idproperty` dell'attività “attività fte: filecopy Ant” a pagina 2160, “fte: filemove Ant attività” a pagina 2164o “attività fte: call Ant” a pagina 2156 .

Esempio

Questo esempio mostra come utilizzare l'attività fte: ignoreoutcome per liberare le risorse allocate per tenere traccia del risultato dell'attività “attività fte: filecopy Ant” a pagina 2160 precedente.

```
<!-- issue a file copy request -->
<fte:filecopy cmdqm="qm1@localhost@1414@SYSTEM.DEF.SVRCONN"
                  src="agent1@qm1" dst="agent1@qm1"
                  idproperty="copy.id"
                  outcome="defer"/>

<!-- do some other things -->

<!-- decide that the result of the copy is not interesting -->
<fte:ignoreoutcome id="${copy.id}"/>
```

Attività correlate

Utilizzo di Apache Ant con MFT

attività fte: ping Ant

Questa attività IBM MQ Managed File Transfer Ant esegue il ping di un agent per ottenere una risposta e determina se l'agent è in grado di elaborare i trasferimenti.

Nota: IBM WebSphere MQ File Transfer Edition (FTE) non è più un prodotto supportato. Per eseguire la migrazione da FTE al componente MFT (Managed File Transfer) in IBM MQ, consultare Migrazione di MFT (Managed File Transfer).

Attributi

agente

Obbligatorio. Specifica l'agent a cui inoltrare la richiesta **fte:ping** . Il valore è nel seguente formato: *agentname*@*qmgrname* dove *agentname* è il nome dell'agent e *qmgrname* è il nome del gestore code a cui questo agent è direttamente connesso.

cmdqm

Facoltativo. Il gestore code comandi a cui inoltrare la richiesta. Specificare queste informazioni nel formato *qmgrname*@*host*@*port*@*channel*, dove:

- *qmgrname* è il nome del gestore code
- *host* è il nome host facoltativo del sistema su cui è in esecuzione il gestore code
- *port* è il numero di porta facoltativo su cui è in ascolto il gestore code
- *channel* è il canale SVRCONN facoltativo da utilizzare

Se si omettono le informazioni *host*, *port*o *channel* per il gestore code comandi, vengono utilizzate le informazioni di connessione specificate nel file `command.properties` .

Attenzione: Se non viene specificato alcun valore per:

- Variabile *host* , viene utilizzata la modalità di bind
- variabile *port* , viene utilizzato il valore 1414
- *channel* , la variabile SYSTEM.DEF.SVRCONN .

Per ulteriori informazioni, consultare Il file MFT command.properties .

Tuttavia, non è possibile ignorare gli attributi nel mezzo, ad esempio `qmgrname@host@@channel`. È possibile avere, ad esempio, `qmgrname@host`, `qmgrname@host@porto` `qmgrname@hostport@@channel`.

MFT suddivide l'attributo fornito utilizzando il delimitatore `@` . A seconda del numero di token trovati, utilizza il primo token come *qmgrname*, il secondo come *host*, il terzo come *port* e infine *channel*.

Per ulteriori informazioni, vedere File MFT command.properties.

È possibile utilizzare la proprietà **`com.ibm.wmqfte.propertySet`** per specificare quale file `command.properties` utilizzare. Per ulteriori informazioni, consultare com.ibm.wmqfte.propertySet.

Se non si utilizza l'attributo cmdqm , per impostazione predefinita l'attività utilizza la proprietà `com.ibm.wmqfte.ant.commandQueueManager` , se è impostata. Se la proprietà `com.ibm.wmqfte.ant.commandQueueManager` non è impostata, viene tentata una connessione al gestore code predefinito, definito nel file `command.properties` . Il formato della proprietà `com.ibm.wmqfte.ant.commandQueueManager` è lo stesso dell'attributo cmdqm , ovvero `qmgrname@host@port@channel`.

proprietà rc
Obbligatorio. Indica una proprietà in cui memorizzare il codice di ritorno dell'operazione **`ping`** .

tempo massimo
Facoltativo. La quantità massima di tempo, in secondi, per cui l'attività attende la risposta dell'agent. Il timeout minimo è zero secondi, tuttavia è possibile specificare anche un timeout meno uno in modo che il comando attenda per sempre la risposta dell'agente. Se non viene specificato alcun valore per `timeout` , il valore predefinito è di attendere fino a 5 secondi per la risposta dell'agent.

Esempio

Questo esempio invia una richiesta **`fte:ping`** a `agent1` ospitata da `qm1`. La richiesta **`fte:ping`** attende 15 secondi che l'agent risponda. Il risultato della richiesta **`fte:ping`** viene memorizzato in una proprietà denominata `ping.rc`.

```
<fte:ping agent="agent1@qm1" rcproperty="ping.rc" timeout="15"/>
```

Codici di ritorno

0
Comando completato correttamente.

2
Comando scaduto.

Attività correlate
Utilizzo di Apache Ant con MFT

fte: uuid Ant attività

Genera un identificativo univoco pseudo - casuale e lo assegna a una determinata proprietà. Ad esempio, è possibile utilizzare questo identificativo per generare nomi lavoro per altre operazioni di trasferimento file.

Attributi

lunghezza
Obbligatorio. La lunghezza numerica dell'UUID da creare. Questo valore di lunghezza non include la lunghezza di alcun prefisso, specificato dal parametro **prefix** .

Proprietà
Obbligatorio. Il nome della proprietà a cui assegnare l'UUID generato.

prefisso
Facoltativo. Un prefisso da aggiungere all'UUID generato. Questo prefisso non viene conteggiato come parte della lunghezza dell'UUID, come specificato dal parametro **length** .

Esempio

Questo esempio definisce un UUID che inizia con le lettere ABC seguite da 16 caratteri esadecimali pseudo - casuali. L'UUID viene assegnato ad una proprietà denominata `uuid.property`.

```
<fte:uuid length="16" property="uuid.property" prefix="ABC"/>
```

Attività correlate

Utilizzo di Apache Ant con MFT

fte: filespec Ant elemento nidificato

Il parametro **fte:filespec** viene utilizzato come elemento nidificato in altre attività. Utilizzare **fte:filespec** per descrivere una mappatura tra uno o più file di origine, directory z/OS o datasete una destinazione. Generalmente, questo elemento viene utilizzato quando si esprime una serie di file o directory z/OS o serie di dati da spostare o copiare.

Nidificato da:

- L'attività fte: filecopy
- Attività fte: filemove

Attributi della specifica di origine

È necessario specificare srcfilespec o srcqueue.

specfilesrc
Specifica l'origine dell'operazione file. Il valore di questo attributo può includere un carattere jolly.

coda origine
Specifica che l'origine del trasferimento è una coda. Il trasferimento sposta i dati dai messaggi memorizzati nella coda specificata da questo attributo. Non è possibile specificare questo attributo se l'attività **fte:filespec** è nidificata all'interno dell'attività **fte:filecopy** .

L'attributo srcqueue non è supportato quando l'agent di origine è un agent bridge di protocollo.

Attributi della specifica di destinazione

È necessario specificare uno tra dstdir, dstds, dstfilespace, dstfile, dstqueue o dstpds.

dstdir
Specifica una directory come destinazione per un'operazione file.

z/OS **dstd**
Specifica un dataset come destinazione per un'operazione file.

Questo attributo è supportato solo quando l'agente di destinazione è in esecuzione sulla piattaforma z/OS .

filedd
Specifica un file come destinazione per un'operazione file.

spaziofilestd
Specifica uno spazio file come destinazione per un'operazione file.

Questo attributo si applica solo se l'agente di destinazione è un agente Web IBM MQ 8.0 che ha accesso allo spazio file del gateway Web.

z/OS **dstpds**
Specifica un dataset partizionato come destinazione per un'operazione file.

Questo attributo è supportato solo quando l'agente di destinazione è in esecuzione sulla piattaforma z/OS .

dstqueue

Specifica una coda come destinazione per un'operazione file - messaggio. Facoltativamente, è possibile includere un nome gestore code in questa specifica, utilizzando il formato QUEUE@QUEUEMANAGER. Se non si specifica un nome gestore code, viene utilizzato il gestore code dell'agent di destinazione se non è stata impostata la proprietà dell'agent di output enableClusterQueueInputsu true. Se la proprietà di output enableClusterQueueInputè impostata su true, l'agent di destinazione utilizza le procedure standard IBM MQ per stabilire dove si trova la coda. È necessario specificare un nome coda valido che esista sul gestore code.

Se si specifica l'attributo dstqueue , non è possibile specificare gli attributi srcqueue perché questi attributi si escludono a vicenda.

L'attributo dstqueue non è supportato quando l'agente di destinazione è un agente bridge di protocollo.

Attributi opzione di origine

codifica srl
Facoltativo. La codifica della serie di caratteri utilizzata dal file da trasferire.

È possibile specificare questo attributo solo quando l'attributo conversion è impostato sul valore `text`.

Se non si specifica l'attributo srcencoding , la serie di caratteri del sistema di origine viene utilizzata per i trasferimenti di testo.

srceol
Facoltativo. Il delimitatore di fine riga utilizzato dal file che si sta trasferendo. I valori validi sono i seguenti:

- CRLF - Utilizzare un carattere di ritorno a capo seguito da un carattere di avanzamento riga come delimitatore di riga. Questa convenzione è tipica per i sistemi Windows .
- LF - Utilizzare un carattere di avanzamento riga come delimitatore di fine riga. Questa convenzione è tipica per i sistemi UNIX .

È possibile specificare questo attributo solo quando l'attributo conversion è impostato sul valore `text`. Se non si specifica l'attributo srceol , i trasferimenti di testo determinano automaticamente il valore corretto in base al sistema operativo dell'agent di origine.

z/OS **spazi di riferimento**
Facoltativo. Determina se gli spazi finali vengono conservati sui record di origine letti da un dataset in formato a lunghezza fissa come parte di un trasferimento in modalità testo. I valori validi sono i seguenti:

- `true` - vengono mantenuti gli spazi finali.
- `false` - gli spazi finali vengono eliminati.

Se non si specifica l'attributo srckeeptrailingspaces , viene specificato il valore predefinito `false` .

È possibile specificare questo attributo solo se si specifica anche l'attributo srcfilespec e si imposta l'attributo conversion su un valore `text`.

srcmsgdelimbyte
Facoltativo. Specifica uno o più valori di byte da inserire come delimitatore quando si aggiungono più messaggi a un file binario. Ogni valore deve essere specificato come due cifre esadecimali nell'intervallo 00-FF, con prefisso x. Più byte devono essere separati da virgola. Ad esempio, `srcmsgdelimbytes="x08,xA4"`. È possibile specificare l'attributo srcmsgdelimbytes solo se è stato specificato anche l'attributo srcqueue . Non è possibile specificare l'attributo srcmsgdelimbytes se è stato anche specificato il valore `text` per l'attributo conversion .

srcmsgdelimtext
Facoltativo. Specifica una sequenza di testo da inserire come delimitatore quando si aggiungono più messaggi a un file di testo. È possibile includere sequenze di escape Java per i letterali stringa nel delimitatore. Ad esempio, `srcmsgdelimtext="\u007d\n"`. Il delimitatore di testo viene inserito dopo ogni messaggio dall'agente di origine. Il delimitatore di testo viene codificato in formato binario utilizzando la codifica di origine del trasferimento. Ogni messaggio viene letto in formato binario, il delimitatore codificato viene accodato in formato binario al messaggio e il risultato viene trasferito in formato binario all'agente di destinazione. Se la codepage dell'agent di origine include gli stati shift - in e shift - out, l'agent presuppone che ogni messaggio si trovi nello stato shift - out alla fine del messaggio. Nell'agente di destinazione i dati binari vengono convertiti nello stesso modo di un trasferimento di testo da file a file. È possibile specificare solo l'attributo srcmsgdelimtext se è stato specificato anche l'attributo srcqueue e un valore `text` per l'attributo conversion .

srcmsgdelimposition
Facoltativo. Specifica la posizione in cui viene inserito il delimitatore di testo o binario. I valori validi sono i seguenti:

- `prefix` - i delimitatori vengono inseriti nel file di destinazione prima dei dati di ciascun messaggio.
- `postfix` - i delimitatori vengono inseriti nel file di destinazione dopo i dati di ciascun messaggio.

È possibile specificare l'attributo srcmsgdelimposition solo se è stato specificato anche uno degli attributi srcmsgdelimbytes o srcmsgdelimtext .

srcmsggroups
Facoltativo. Specifica che i messaggi sono raggruppati per ID gruppo IBM MQ . Il primo gruppo completo viene scritto nel file di destinazione. Se questo attributo non viene specificato, tutti i messaggi sulla coda di origine vengono scritti nel file di destinazione. È possibile specificare l'attributo srcmsggroups solo se è stato specificato anche l'attributo srcqueue .

srcqueuetimeout
Facoltativo. Specifica il tempo, in secondi, di attesa per una delle seguenti condizioni da soddisfare:

- Per un nuovo messaggio da scrivere nella coda.
- Se è stato specificato l'attributo srcmsggroups , per un gruppo completo da scrivere sulla coda.

Se nessuna di queste condizioni è soddisfatta entro il tempo specificato dal valore di srcqueuetimeout, l'agent di origine arresta la lettura dalla coda e completa il trasferimento. Se l'attributo srcqueuetimeout non è specificato, l'agent di origine arresta immediatamente la lettura dalla coda di origine se la coda di origine è vuota o, nel caso in cui sia stato specificato l'attributo srcmsggroups , se non è presente alcun gruppo completo sulla coda. È possibile specificare l'attributo srcqueuetimeout solo se è stato specificato anche l'attributo srcqueue .

Per informazioni relative all'impostazione del valore *srcqueuetimeout* , consultare Guida per specificare un tempo di attesa su un trasferimento da messaggio a file.

z/OS **srcrecdelimbytes**
Facoltativo. Specifica uno o più valori di byte da inserire come delimitatore quando si aggiungono più record da un file di origine orientato ai record a un file binario. È necessario specificare ogni valore come due cifre esadecimali comprese nell'intervallo 00-FF, con prefisso x. Più byte devono essere separati da virgola. Ad esempio:

```
srcrecdelimbytes="x08,xA4"
```

È possibile specificare l'attributo srcrecdelimbytes solo se il file di origine del trasferimento è orientato ai record, ad esempio un dataset z/OS e il file di destinazione è un file normale, non orientato ai record. Non è possibile specificare l'attributo srcrecdelimbytes se è stato anche specificato il valore `text` per l'attributo conversion .

srcrecdelimpos

Facoltativo. Specifica la posizione in cui viene inserito il delimitatore binario. I valori validi sono i seguenti:

- prefisso - i delimitatori vengono inseriti nel file di destinazione prima dei dati da ciascun record del file orientato ai record di origine.
- postfix - i delimitatori vengono inseriti nel file di destinazione dopo i dati da ogni record del file orientato al record di origine.

È possibile specificare l'attributo srcrecdelimpos solo se è stato specificato anche l'attributo srcrecdelimbytes .

Attributi opzione di destinazione

codifica dati

Facoltativo. La codifica della serie di caratteri da utilizzare per il file trasferito.

È possibile specificare questo attributo solo quando l'attributo conversion è impostato sul valore `text.`

Se l'attributo dstencoding non viene specificato, la serie di caratteri del sistema di destinazione viene utilizzata per i trasferimenti di testo.

dsteolo

Facoltativo. Il delimitatore di fine riga da utilizzare per il file trasferito. I valori validi sono i seguenti:

- CRLF - Utilizzare un carattere di ritorno a capo seguito da un carattere di avanzamento riga come delimitatore di riga. Questa convenzione è tipica per i sistemi Windows .
- LF - Utilizzare un carattere di avanzamento riga come delimitatore di fine riga. Questa convenzione è tipica per i sistemi UNIX .

È possibile specificare questo attributo solo quando l'attributo conversion è impostato sul valore `text.`

Se non si specifica l'attributo dsteol , i trasferimenti di testo determinano automaticamente il valore corretto in base al sistema operativo dell'agent di destinazione.

dstmsgdelimbytes

Facoltativo. Specifica il delimitatore esadecimale da utilizzare quando si suddivide un file binario in più messaggi. Tutti i messaggi hanno lo stesso ID gruppo IBM MQ ; l'ultimo messaggio nel gruppo ha l'indicatore IBM MQ LAST_MSG_IN_GROUP impostato. Il formato per specificare un byte esadecimale come delimitatore è `xNN`, dove N è un carattere compreso nell'intervallo 0 - 9 o a - f. È possibile specificare una sequenza di byte esadecimali come delimitatori specificando un elenco separato da virgole di byte esadecimali, ad esempio `x3e,x20,x20,xbf`.

È possibile specificare l'attributo dstmsgdelimbytes solo se è stato specificato anche l'attributo dstqueue e il trasferimento è in modalità binaria. È possibile specificare solo uno tra gli attributi dstmsgsize, dstmsgdelimbytese dstmsgdelimpattern .

dstmsgdelimpattern

Facoltativo. Specifica l'espressione regolare Java da utilizzare quando si suddivide un file di testo in più messaggi. Tutti i messaggi hanno lo stesso ID gruppo IBM MQ ; l'ultimo messaggio nel gruppo ha l'indicatore IBM MQ LAST_MSG_IN_GROUP impostato. Il formato per la specifica di un'espressione regolare come delimitatore è un'espressione regolare racchiusa tra parentesi, `(`*`regular_expression`*`)` o racchiusa tra virgolette, "*`regular_expression`*". Per ulteriori informazioni, consultare Espressioni regolari utilizzate da MFT.

Per impostazione predefinita, la lunghezza della stringa a cui l'espressione regolare può corrispondere è limitata dall'agente di destinazione a cinque caratteri. È possibile modificare questo comportamento utilizzando la proprietà agent **maxDelimiterMatchLength** . Per ulteriori informazioni, consultare MFT advanced agent properties.

È possibile specificare l'attributo dstmsgdelimpattern solo se è stato specificato anche l'attributo dstqueue e il trasferimento è in modalità testo. È possibile specificare solo uno tra gli attributi dstmsgsize, dstmsgdelimbytese dstmsgdelimpattern .

posizione dstmsgdelimposition
Facoltativo. Specifica la posizione in cui è previsto che si trovi il delimitatore di testo o binario. I valori validi sono i seguenti:

- `prefix` - I delimitatori sono previsti all'inizio di ciascuna riga.
- `postfix` - I delimitatori sono previsti alla fine di ogni riga.

È possibile specificare l'attributo dstmsgdelimposition solo se è stato specificato anche l'attributo dstmsgdelimpattern .

dstmsgincludedelim
Facoltativo. Specifica se includere il delimitatore utilizzato per suddividere il file in più messaggi nei messaggi. Se viene specificato l'attributo dstmsgincludedelim , il delimitatore viene incluso alla fine del messaggio che contiene i dati del file che precedono il delimitatore. Per impostazione predefinita il delimitatore non è incluso nei messaggi. È possibile specificare l'attributo dstmsgincludedelim solo se è stato specificato anche uno degli attributi dstmsgdelimpattern e dstmsgdelimbytes .

dstmsgpersist
Facoltativo. Specifica se i messaggi scritti nella coda di destinazione sono persistenti. I valori validi sono i seguenti:

- `true` - Scrivi messaggi persistenti nella coda di destinazione. Questo è il valore predefinito.
- `false` - Scrivi messaggi non persistenti nella coda di destinazione.
- `qdef` - Il valore di persistenza viene preso dall'attributo DefPersistence della coda di destinazione.

È possibile specificare questo attributo solo quando viene specificato anche l'attributo dstqueue .

dstmsgprops
Facoltativo. Specifica se per il primo messaggio scritto nella coda di destinazione dal trasferimento sono impostate le proprietà del messaggio IBM MQ . I possibili valori sono:

- `true` - Impostare le proprietà del messaggio sul primo messaggio creato dal trasferimento.
- `false` - Non impostare le proprietà del messaggio sul primo messaggio creato dal trasferimento. Questo è il valore predefinito.

Per ulteriori informazioni, vedere Proprietà dei messaggi diMQ impostate da MFT sui messaggi scritti nelle code di destinazione.

È possibile specificare questo attributo solo quando viene specificato anche l'attributo dstqueue .

dimensione dstmsgsize
Facoltativo. Specifica se suddividere il file in più messaggi a lunghezza fissa. Tutti i messaggi hanno lo stesso IBM MQ ID gruppo; l'ultimo messaggio nel gruppo ha l'indicatore IBM MQ LAST_MSG_IN_GROUP impostato. La dimensione dei messaggi è specificata dal valore dstmsgsize. Il formato di dstmsgsize è *lengthunits*, dove *length* è un valore intero positivo e *units* è uno dei seguenti valori:

- B - Byte. Il valore minimo consentito è due volte il valore massimo di byte per carattere della codepage dei messaggi di destinazione.
- K - Kibibyte. Ciò equivale a 1024 byte.
- M - Mebibyte. Equivale a 1024 kibibyte.

Se il file viene trasferito in modalità testo e si trova in una serie di caratteri a doppio byte o in una serie di caratteri a più byte, il file viene suddiviso in messaggi sul limite di caratteri più vicino alla dimensione del messaggio specificata.

È possibile specificare l'attributo dstmsgsize solo se è stato specificato anche l'attributo dstqueue . È possibile specificare solo uno tra gli attributi dstmsgsize, dstmsgdelimbytese dstmsgdelimpattern .

dstunsupportedcodepage
Facoltativo. Specifica l'azione da intraprendere se il gestore code di destinazione, come specificato dall'attributo dstqueue , non supporta la codepage utilizzata durante il trasferimento dei dati file in una coda come trasferimento di testo. I valori validi per questo attributo sono i seguenti:

- `binary` - continuare il trasferimento ma non applicare la conversione della codepage ai dati in fase di trasferimento. Specificare questo valore equivale a non impostare l'attributo di conversione su `text`.
- `fail` - non continuare con l'operazione di trasferimento. Il trasferimento del file non è riuscito. Questa è l'opzione predefinita.

È possibile specificare l'attributo dstunsupportedcodepage solo se è stato specificato anche l'attributo dstqueue e un valore `text` per l'attributo conversion .

z/OS **dsttruncaterecords**
Facoltativo. Specifica che i record di destinazione più lunghi dell'attributo dataset LRECL vengono troncati. Se impostato su true, i record vengono troncati. Se impostato su false, i record vengono riportati a capo. L'impostazione predefinita è false. Questo parametro è valido solo per trasferimenti in modalità testo in cui la destinazione è un dataset.

Altri attributi

checksum
Facoltativo. Determina l'algoritmo utilizzato per il checksum dei file trasferiti.

- `MD5` - utilizzare l'algoritmo di hash MD5 .
- `NONE` - non utilizzare un algoritmo checksum.

Se non si specifica l'attributo checksum , viene utilizzato il valore predefinito `MD5` .

trasformazione
Facoltativo. Specifica il tipo di conversione da applicare al file mentre viene trasferito. I possibili valori sono:

- `binary` - non applicare alcuna conversione.
- `text` - applicare la conversione della codepage tra i sistemi di origine e di destinazione. Applicare anche la conversione dei delimitatori di riga. Gli attributi `srcencoding`, `dstencoding`, `srceol` e `dsteol` influenzano la conversione applicata.

Se non si specifica l'attributo conversion , viene specificato il valore predefinito `binary` .

sovrascrivere
Facoltativo. Determina se un file di destinazione esistente z/OS o un dataset può essere sovrascritto dall'operazione. Quando si specifica un valore di `true`, qualsiasi file di destinazione esistente z/OS o dataset viene sovrascritto. Quando si specifica un valore di `false`, l'esistenza di un file duplicato z/OS o di un dataset sulla destinazione determina l'esito negativo dell'operazione. Se l'attributo overwrite non viene specificato, viene specificato un valore predefinito di `false` .

eseguire in forma ricorsiva
Facoltativo. Determina se il trasferimento file viene ripetuto nelle sottodirectory. Quando si specifica un valore di `true`, il trasferimento ricorde nelle sottodirectory. Quando si specifica un valore di `false`, il trasferimento non ricorre nelle sottodirectory. Se l'attributo recurse non viene specificato, viene specificato un valore predefinito di `false` .

Esempio

Questo esempio specifica fte: filespec con un file di origine `file1.bin` e un file di destinazione `file2.bin`.

```
<fte:filespec srcfilespec="/home/fteuser/file1.bin" dstfile="/home/fteuser/file2.bin"/>
```

Attività correlate

Utilizzo di Apache Ant con MFT

fte: metadati Ant elementi nidificati

I metadati vengono utilizzati per trasportare ulteriori informazioni definite dall'utente con un'operazione di trasferimento file.

Consultare “Metadati per uscite utente MFT” a pagina 2180 per ulteriori informazioni su come Managed File Transfer utilizza i metadati.

Nidificato da:

- L'attività fte: filecopy
- Attività fte: filemove
- Attività fte: call

Parametri specificati come elementi nidificati

fte: voce
È necessario specificare almeno una voce all'interno dell'elemento nidificato fte:metadata . È possibile scegliere di specificare più di una voce. Le voci associano un nome chiave ad un valore. Le chiavi devono essere univoche in un blocco di fte:metadata

Attributi voce

Nome
Obbligatorio. Il nome della chiave appartenente a questa voce. Questo nome deve essere univoco tra tutti **`entry`** parametri nidificati all'interno di un elemento fte: metadata .

Valore
Obbligatorio. Il valore da assegnare a questa voce.

Esempio

Questo esempio mostra una definizione fte:metadata che contiene due voci.

```
<fte:metadata>
    <fte:entry name="org.foo.partColor" value="red"/>
    <fte:entry name="org.foo.partSize" value="medium"/>
</fte:metadata>
```

Attività correlate

Utilizzo di Apache Ant con MFT

Elementi nidificati richiamo programma

I programma possono essere avviati utilizzando uno dei cinque elementi nidificati: `fte:presrc`, `fte:predst`, `fte:postdst`, `fte:postsrce` `fte:command`. Questi elementi nidificati indicano a un agente di richiamare un programma esterno come parte della sua elaborazione. Prima di avviare un programma, è necessario assicurarsi che il comando si trovi in un'ubicazione specificata dalla proprietà commandPath nel file `agent.properties` dell'agent che esegue il comando.

Anche se ogni elemento di richiamo del programma ha un altro nome, condividono la stessa serie di attributi e la stessa serie di elementi nidificati. I programmi possono essere avviati dalle attività Ant **`fte:filecopy`**, **`fte:filemove`**e **`fte:command`** .

Non è possibile richiamare programmi da un agent bridge Connect:Direct .

Attività Ant che possono richiamare programmi:

- l'attività fte: filecopy nidifica i parametri di chiamata del programma utilizzando gli elementi nidificati `fte:predst`, `fte:postdst`, `fte:presrce` `fte:postsrc` .
- L'attività fte: filemove nidifica i parametri di richiamo del programma utilizzando elementi nidificati `fte:predst`, `fte:postdst`, `fte:presrce` `fte:postsrc` .
- L'attività fte: call nidifica i parametri di chiamata del programma utilizzando l'elemento nidificato `fte:command` .

Attributi

comando
Obbligatorio. Denomina il programma da richiamare. Per consentire all'agente di eseguire un comando, il comando deve trovarsi in un'ubicazione specificata dalla proprietà commandPath nel file `agent.properties` dell'agente. Per ulteriori informazioni, consultare commandPath MFT property. Qualsiasi informazione sul percorso specificata nell'attributo `command` viene considerata relativa a una posizione specificata dalla proprietà commandPath . Quando `type` è `executable`, è previsto un programma eseguibile, altrimenti è previsto uno script appropriato per il tipo di chiamata.

Num. nuovi tentativi
Facoltativo. Il numero di volte in cui ritentare la chiamata al programma se il programma non restituisce un codice di ritorno di esito positivo. Il programma indicato dall'attributo `command` viene richiamato fino a questo numero di volte. Il valore assegnato a questo attributo deve essere non negativo. Se non si specifica l'attributo `retrycount` , viene utilizzato il valore predefinito zero.

attesa nuovo tentativo
Facoltativo. Il tempo di attesa, in secondi, prima di ritentare il richiamo del programma. Se il programma indicato dall'attributo `command` non restituisce un codice di ritorno di esito positivo e l'attributo `retrycount` specifica un valore diverso da zero, questo parametro determina il periodo di attesa tra i tentativi. Il valore assegnato a questo attributo deve essere non negativo. Se non si specifica l'attributo `retrywait` , viene utilizzato il valore predefinito zero.

riuscito
Facoltativo. Il valore di questo attributo viene utilizzato per stabilire quando il richiamo del programma viene eseguito correttamente. Il codice di ritorno del processo per il comando viene valutato utilizzando questa espressione. Il valore può essere composto da una o più espressioni combinate con un carattere barra verticale (|) per indicare il valore booleano ORo una e commerciale (&) per indicare il valore booleano AND. Ogni espressione può essere uno dei seguenti tipi di espressione:

- Un numero che indica un test di uguaglianza tra il codice di ritorno del processo e il numero.
- Un numero preceduto da un carattere ">" per indicare un test maggiore tra il numero e il codice di ritorno del processo.
- Un numero preceduto da un carattere "<" per indicare un test minore di tra il numero e il codice di ritorno del processo.
- Un numero con prefisso "!" per indicare un test non uguale tra il numero e il codice di ritorno del processo.

Ad esempio: `>2&<7&!5|0|14` viene interpretato come i seguenti codici di ritorno: 0, 3, 4, 6, 14. Tutti gli altri codici di ritorno vengono interpretati come non riusciti. Se non si specifica l'attributo `successrc` , viene utilizzato il valore predefinito zero. Ciò significa che si ritiene che il comando sia stato eseguito correttamente se, e solo se, restituisce un codice di zero.

il tipo
Facoltativo. Il valore di questo attributo specifica quale tipo di programma viene richiamato. Specifica una delle seguenti opzioni:

eseguibile
L'attività richiama un programma eseguibile. È possibile specificare ulteriori argomenti utilizzando l'elemento nidificato arg . È previsto che il programma sia accessibile su commandPath e, laddove applicabile, che disponga dell'autorizzazione di esecuzione impostata. Gli script UNIX possono essere richiamati purché specifichino un programma shell (ad esempio, la prima riga del file script shell è: `#!/bin/sh`). L'output del comando scritto in stderr o stdout viene inviato al log Managed File Transfer per la chiamata. Tuttavia, la quantità di dati emessi è limitata dalla configurazione dell'agente. Il valore predefinito è 10K byte di dati, ma è possibile sovrascrivere questo valore predefinito utilizzando la proprietà dell'agent: maxCommandOutput.

antscript
L'attività esegue lo script Ant specificato utilizzando il comando **`fteAnt`** . Le proprietà possono essere specificate utilizzando l'elemento nidificato property . Le destinazioni Ant possono essere specificate utilizzando l'elemento nidificato target . Lo script Ant dovrebbe essere accessibile su commandPath. L'output di Ant scritto in stderr o stdout viene inviato al log Managed File Transfer per la chiamata. Tuttavia, la quantità di dati emessi è limitata dalla configurazione dell'agente. Il valore predefinito è 10K byte di dati ma è possibile sovrascrivere questo valore predefinito utilizzando la proprietà dell'agent: maxCommandOutput.

z/OS **JCL**
Il valore `jcl` è supportato solo su z/OS ed esegue lo script z/OS JCL specificato. Il JCL viene inoltrato come lavoro e richiede la presenza della scheda di lavoro. Quando il lavoro viene inoltrato correttamente, l'output del comando JCL, scritto nel log Managed File Transfer , contiene il seguente testo: JOB *nome_lavoro*(*id_lavoro*) dove:

- *nome_lavoro* è il nome del lavoro identificato dalla scheda lavoro nel JCL.
- *id_lavoro* è l'ID lavoro generato dal sistema z/OS .

Se il lavoro non può essere inoltrato correttamente, il comando di script JCL ha esito negativo e scrive un messaggio nel log che indica il motivo dell'errore (ad esempio, non è presente alcuna scheda di lavoro). Per comprendere se il lavoro è stato eseguito o completato correttamente, utilizzare un servizio di sistema come SDSF. Managed File Transfer non fornisce le informazioni perché inoltra solo il lavoro; il sistema determina quando eseguire il lavoro e come viene presentato l'output del lavoro. Poiché uno script JCL viene inoltrato come un lavoro batch, non è consigliabile specificare `jcl` per un elemento nidificato `presrc` o `predst` in quanto si sa solo che il lavoro è stato inoltrato correttamente e non se è stato eseguito correttamente prima dell'inizio del trasferimento. Non esistono elementi nidificati validi con un tipo di `jcl`.

Il seguente esempio mostra un lavoro JCL:

```
//MYJOB JOB
//*
//MYJOB EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=H
//SYSUT1 DD DSN=FRED.DEMO.TXT,DISP=SHR
//SYSUT2 DD DSN=BOB.DEMO.TXT,DISP=(NEW,CATLG),
// RECFM=VB,LRECL=133,BLKSIZE=2048,
// SPACE=(TRK,(30,5),RLSE)
//SYSIN DD DUMMY
```

Parametri specificati come elementi nidificati

fte: arg
Valido solo quando il valore dell'attributo `type` è eseguibile. Utilizzare elementi `fte:arg` nidificati per specificare argomenti per il programma che viene richiamato come parte del richiamo del programma. Gli argomenti del programma vengono creati dai valori specificati dagli elementi `fte:arg` nell'ordine in cui vengono rilevati gli elementi `fte:arg` . È possibile scegliere di specificare zero o più elementi `fte:arg` come elementi nidificati di un richiamo del programma.

fte: proprietà
Valido solo dove il valore dell'attributo `type` è antscript. Utilizzare gli attributi `name` e `value` degli elementi `fte:property` nidificati per passare coppie nome - valore allo script Ant . È possibile scegliere di specificare zero o più elementi `property` come elementi nidificati di un richiamo del programma.

fte: destinazione
Valido solo dove il valore dell'attributo `type` è antscript. Specificare una destinazione nello script Ant da chiamare. È possibile scegliere di specificare zero o più elementi `fte:target` come elementi nidificati di un richiamo del programma.

Attributi arg

valore
Obbligatorio. Il valore dell'argomento da passare al programma richiamato.

Attributi di Property

nome
Obbligatorio. Il nome di una proprietà da passare allo script Ant .

valore
Obbligatorio. Il valore da associare al nome della proprietà inoltrato allo script Ant .

Esempi

Questo esempio mostra un richiamo del programma `fte:postsrc` specificato come parte di un'attività fte: filecopy . Il richiamo del programma è per un programma denominato `post.sh` e viene fornito un singolo argomento di `/home/fteuser2/file.bin.`

```
<fte:filecopy
cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
 src="agent1@qm1" dst="agent2@qm2"
 rcproperty="copy.result">
<fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>
<fte:postsrc command="post.sh" successrc="1" >
    <fte:arg value="/home/fteuser2/file.bin"/>
</fte:postsrc>
</fte:filecopy>
```

Questo esempio mostra un richiamo del programma `fte:command` specificato come parte di una attività fte: call . Il richiamo del programma è per un eseguibile denominato `command.sh`, a cui non viene passato alcun argomento della riga comandi. Se `command.sh` non restituisce un codice di ritorno di esito positivo pari a 1, il comando viene ritentato dopo 30 secondi.

```
<fte:call
cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
 agent="agent1@qm1"
 rcproperty="call.rc"
 origuser="bob"
 jobname="${job.id}">
<fte:command command="command.sh" successrc="1" retrycount="5" retrywait="30"/>
</fte:call>
```

Questo esempio mostra un richiamo del programma `fte:command` specificato come parte di una attività fte: call . Il richiamo del programma è per le destinazioni di copia e compressione in un script Ant denominato `script.xml`, a cui vengono passate due proprietà.

```
<fte:call
cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
 agent="agent1@qm1"
 rcproperty="call.rc"
 origuser="bob"
 jobname="${job.id}">
<fte:command command="script.xml" type="antscript">
```

```
   <property name="src" value="AGENT5@QM5"/>
   <property name="dst" value="AGENT3@QM3"/>
   <target name="copy"/>
   <target name="compress"/>
 </fte:command>
 </fte:call>
```

Attività correlate

Specifica dei programmi da eseguire con MFT

Utilizzo di Apache Ant con MFT

Uscite utente MFT per riferimento personalizzazione

Informazioni di riferimento per la configurazione delle uscite utente per Managed File Transfer.

Concetti correlati

Uscite utente di origine e destinazione MFT

Metadati per uscite utente MFT

Ci sono tre diversi tipi di metadati che possono essere forniti alle routine di uscita utente per Managed File Transfer: ambiente, trasferimento e metadati file. Questi metadati vengono presentati come associazioni di coppie chiave - valore Java .

Metadati di ambiente

I metadati dell'ambiente vengono passati a tutte le routine di uscita utente e descrivono l'ambiente di runtime agent da cui viene richiamata la routine di uscita utente. Questi metadati sono di sola lettura e non possono essere aggiornati da alcuna routine di uscita utente.

Tabella 882. Metadati di ambiente

Chiave	Descrizione
AGENT_CONFIGURATION_DIRECTORY_KEY	Il nome della directory che contiene le informazioni di configurazione dell'agent.
CHIAVE_DIRECTORY_PRODOTTO_AGENT	Il nome della directory in cui è stato installato il codice agent.
CHIAVE_VERSO_AGENT	Numero di versione per il runtime dell'agent che richiama la routine di uscita.

I nomi chiave e i nomi valore forniti nella Tabella 1 sono costanti definite nell'interfacciaDataConstants EnvironmentMeta.

Metadati di trasferimento

I metadati di trasferimento vengono passati a tutte le routine di uscita utente. I metadati sono costituiti da valori forniti dal sistema e valori forniti dall'utente. Se si modificano i valori forniti dal sistema, queste modifiche vengono ignorate. I valori iniziali forniti dall'utente per l'uscita utente iniziale del trasferimento di origine si basano sui valori forniti quando si definisce il trasferimento. L'agent di origine può modificare i valori forniti dall'utente come parte dell'elaborazione dell'uscita utente di inizio trasferimento di origine. Questa uscita utente viene richiamata prima dell'avvio dell'intero trasferimento file. Queste modifiche vengono utilizzate nelle chiamate successive ad altre routine di uscita correlate a tale trasferimento. I metadati del trasferimento vengono applicati a un intero trasferimento.

Sebbene tutte le uscite utente possano leggere i valori dai metadati del trasferimento, solo l'uscita utente di avvio del trasferimento di origine può modificare i metadati del trasferimento

Non è possibile utilizzare i metadati di trasferimento per propagare le informazioni tra trasferimenti file differenti.

I metadati di trasferimento forniti dal sistema sono descritti in dettaglio nella Tabella 2:

Tabella 883. Metadati di trasferimento	
Chiave	**Descrizione**
CHIAVE_AGENT_DI_DESTINAZIONE	Il nome dell'agent che è la destinazione per il trasferimento.
CHIAVE_NOME_LAVORO	Il nome lavoro associato alla richiesta di trasferimento
CHIAVE_UTENTE_MQMD	Il campo utente MQMD dal messaggio utilizzato per inoltrare la richiesta di trasferimento
CHIAVE_HOST_DI origine	Il nome host specificato come nome host di origine nella richiesta di trasferimento
CHIAVE_UTENTE_DI origine	Il nome utente specificato come ID utente di origine nella richiesta di trasferimento
CHIAVE_ORIGINE_ORIGINE	Il nome dell'agent che è l'origine del trasferimento
CHIAVE_ID_TRASFERIMENTO	L'identificatore del trasferimento

I nomi chiave e i nomi valore forniti nella Tabella 2 sono costanti definite nell'interfacciaDataConstants TransferMeta.

Metadati file

I metadati del file vengono passati all'uscita di avvio del trasferimento di origine come parte della specifica file. Esistono metadati di file separati per i file di origine e di destinazione.

Non è possibile utilizzare metadati di file per propagare le informazioni tra diversi trasferimenti di file.

Tabella 884. Metadati file		
Chiave	**Valori consentiti**	**Descrizione**
CONVERT_LINE_SEPARATORS		Valore chiave utilizzato per i trasferimenti di testo per indicare se le sequenze di separatori di riga CRLF (carriage return - line feed) o LF (line feed) nei dati di origine vengono convertite nella sequenza di separatori di riga nella destinazione.
CHIAVE_DELIMITATORE		Valore chiave utilizzato per definire un delimitatore per separare i dati record quando si trasferiscono i dati orientati ai record in file normali. Utilizzato anche per i trasferimenti da messaggio a file e da file a messaggio.
CHIAVE_POSIZIONE_DELIMITATORE	DELIMITER_POSITION_PREFIX_VALUE VALORE_POSTFISSO_DELIMITATORE	Utilizzare con DELIMITER_KEY per definire la posizione del delimitatore; prefisso o suffisso.
CHIAVE_TIPO_DELIMITATORE	VALORE_BINARIO_TIPO_DELIMITATORE VALORE_TESTO_TIPO_DELIMITATORE VALORE_DIMENSIONE_TIPO_DELIMITATORE	Utilizzare con DELIMITER_KEY per definire il tipo di delimitatore.
CHIAVE_ESISTA_DESTINAZIONE	VALORE_ERRORE_CHIAVE_ESISTENZA_DI_DESTINAZIONE_VALORE_ESISTENZA_CHIAVE_SOVRASCRITTURA	Determina il comportamento del trasferimento file se il file di destinazione esiste.
CHIAVE_ALIAS_FILE		Valore chiave utilizzato per definire un alias per il file che si sta trasferendo.

Tabella 884. Metadati file (Continua)

Chiave	Valori consentiti	Descrizione
CHIAVE_METODO_CONTROLLO_FILE	FILE_CHECKSUM_METHOD_NONE_VALUE FILE_CHECKSUM_METHOD_MD5_VALUE	Determina il metodo di checksum da utilizzare durante il trasferimento del file.
CHIAVE_CONVERSIONE_FILE	VALORE_TESTO_CONVERSIONE_FILE VALORE_BINARIO_CONVERSIONE	Determina il tipo di conversione applicato al contenuto del file.
FILE_XX_ENCODE_CASE_ONE chiave_di_inserimento		Determina la codifica utilizzata per un file di testo.
FILE_END_OF_LINE_KEY	FILE_END_OF_LINE_LF_VALUE FILE_END_OF_LINE_CRLF_VALUE	Determina la sequenza di caratteri che indica la fine di una riga: < LF> o < CR> < LF>.
FILE_SPACE_ALIAS		Determina l'alias di un file nello spazio file. **Nota:** Questi metadati possono essere utilizzati solo se FILE_TYPE_KEY è FILE_TYPE_FILE_SPACE_VALUE
NOME_SPAZIO_FILE		Determina il nome dello spazio file. **Nota:** Questi metadati possono essere utilizzati solo se FILE_TYPE_KEY è FILE_TYPE_FILE_SPACE_VALUE
CHIAVE_TIPO_FILE	FILE_TYPE_FILE_VALUE FILE_TYPE_DIRECTORY_VALUE FILE_TYPE_DATASET_VALUE FILE_TYPE_PDS_VALUE FILE_TYPE_QUEUE_VALUE FILE_TYPE_FILE_SPACE_VALUE	Determina il file di destinazione, la coda o la specifica dello spazio file.
CHIAVE_ID_GRUPPO		Il valore chiave utilizzato per i trasferimenti da messaggio a file per determinare il gruppo di messaggi da leggere dalla coda origine. Questo attributo è valido solo quando il valore di USE_GROUPS_KEY è USE_GROUPS_TRUE_VALUE.
INCLUDE_DELIMITATORE_IN_CHIAVE_MESSAGGIO	INCLUDE_DELIMITER_IN_MESSAGE_TRUE_VALUE INCLUDE_DELIMITER_IN_MESSAGE_FALSE_VALUE	Il valore chiave utilizzato per i trasferimenti da file a messaggio per determinare se includere i delimitatori utilizzati per suddividere il file in più messaggi alla fine dei messaggi. Questo attributo è valido solo quando il valore di DELIMITER_TYPE_KEY è DELIMITER_TYPE_BINARY_VALUE DELIMITER_TYPE_TEXT_VALUE.
INSERT_RECORD_LINE_SEPARATOR_KEY		Valore chiave utilizzato per i trasferimenti di testo da file orientati ai record per specificare se i separatori di riga vengono inseriti nei dati dopo ogni record.
KEEP_TRAILING_SPACES_KEY	KEEP_TRAILING_SPACES_TRUE_VALUE KEEP_TRAILING_SPACES_FALSE_VALUE	Valore chiave utilizzato per determinare se gli spazi finali vengono rimossi dai record letti dai dataset in formato a lunghezza fissa.

Tabella 884. Metadati file (Continua)		
Chiave	**Valori consentiti**	**Descrizione**
NUOVA_RECORD_ON_LINE_SEPARATOR_KEY		Il valore chiave utilizzato per i trasferimenti di testo ai file orientati ai record per specificare se i separatori di riga nei dati vengono inclusi con i dati del record o causano un nuovo record (e non vengono scritti).
CHIAVE_PERSIST_PRINCIPALE	VALORE TRUE_PERSISTENT_ VALORE FALSE_PERSISTENT_ PERSISTENT_QDEF_VALUE	Il valore chiave utilizzato per i trasferimenti file - a - messaggio per determinare se i messaggi sono persistenti.
SET_MQ_PROPS_KEY	SET_MQ_PROPS_VALORE VALORE FALSE MQ_SET	Valore chiave utilizzato per i trasferimenti da file a messaggio per determinare se le proprietà del messaggio IBM MQ sono impostate sul primo messaggio in un file e i messaggi scritti nella coda quando si verifica un errore.
UNRECOGNISED_CODE_PAGE_KEY	VALORE_PAGINA_CODE_NON_RICONOSCIUTO_NON_VALORE_BINARIO_CODE_NON_RICONOSCIUTO_XX_ENCODE_CASE_CAPS_LOCK_OFF	Valore chiave utilizzato per i trasferimenti da file a messaggio per stabilire se un trasferimento in modalità testo non riesce o se viene eseguita la conversione, se la codepage dei dati non è riconosciuta dal gestore code di destinazione.
CHIAVE_GRUPPI_UTILIZZO	USE_GROUPS_TRUE_VALUE USE_GROUPS_FALSE_VALUE	Valore chiave utilizzato per i trasferimenti da messaggio a file per determinare se trasferire solo un gruppo completo di messaggi dalla coda di origine.
CHIAVE_TEMPO_DI_ATTESA		Valore chiave utilizzato per i trasferimenti da messaggio a file per determinare il tempo, in secondi, di attesa dell'agent di origine per uno dei seguenti casi: • Un messaggio da visualizzare nella coda di origine, se la coda è vuota o è diventata vuota, se il valore di USE_GROUPS_KEY è FALSE. • Un gruppo completo da visualizzare nella coda di origine, se il valore di USE_GROUPS_KEY è TRUE.

I nomi chiave e i nomi valore forniti nella Tabella 3 sono costanti definite nell'interfacciaDataConstants FileMeta.

Concetti correlati

“Interfacce Java per uscite utente MFT” a pagina 2190
Utilizzare gli argomenti in questa sezione per informazioni di riferimento sulle interfacce Java per le routine di uscita utente.

Attività correlate

Personalizzare MFT con uscite utente

Riferimenti correlati

“Uscite utente del monitoraggio risorse MFT” a pagina 2184

Le uscite utente del monitoraggio risorse consentono di configurare codice personalizzato da eseguire quando viene soddisfatta una condizione di trigger del monitoraggio, prima che venga avviata l'attività associata.

“MFT Proprietà dell'agent per uscite utente” a pagina 2188
Oltre alle proprietà standard nel file `agent.properties` sono disponibili diverse proprietà avanzate specifiche per le routine di uscita utente. Queste proprietà non sono incluse per impostazione predefinita, quindi se si desidera utilizzarle, è necessario modificare manualmente il file `agent.properties` . Se si apporta una modifica al file `agent.properties` mentre l'agent è in esecuzione, arrestare e riavviare l'agent per rendere effettive le modifiche.

Uscite utente del monitoraggio risorse MFT

Le uscite utente del monitoraggio risorse consentono di configurare codice personalizzato da eseguire quando viene soddisfatta una condizione di trigger del monitoraggio, prima che venga avviata l'attività associata.

Si consiglia di non richiamare nuovi trasferimenti direttamente dal codice di uscita utente. In alcune circostanze, ciò fa sì che i file vengano trasferiti più volte poiché le uscite utente non sono resilienti ai riavvii dell'agente.

Le uscite utente del monitoraggio risorse utilizzano l'infrastruttura esistente per le uscite utente. Le uscite utente di monitoraggio vengono richiamate dopo l'attivazione di un monitoraggio, ma prima che l'attività corrispondente sia stata eseguita dall'attività di monitoraggio. Ciò consente all'uscita utente di modificare l'attività da eseguire e decidere se un'attività deve procedere o meno. È possibile modificare l'attività di controllo aggiornandone i metadati, che vengono quindi utilizzati per la sostituzione delle variabili nel documento dell'attività creato dalla creazione del monitor originale. In alternativa, l'uscita di monitoraggio può sostituire o aggiornare la stringa XML di definizione attività inoltrata come parametro. L'uscita di monitoraggio può restituire un codice di risultato di 'procedi 'o' annulla ' per l'attività. Se viene restituito l'annullamento, l'attività non verrà avviata e il monitoraggio non verrà avviato di nuovo fino a quando la risorsa monitorata non corrisponde alle condizioni di trigger. Se la risorsa non è stata modificata, il trigger non verrà avviato. Come per le altre uscite utente, è possibile concatenare le uscite di monitoraggio. Se una delle uscite restituisce un codice di risultato di annullamento, il risultato complessivo viene annullato e l'attività non viene avviata.

- Una associazione di metadati di ambiente (uguale ad altre uscite utente)
- Una mappa di metadati di monitoraggio che include metadati di sistema immutabili e metadati utente modificabili. I metadati di sistema immutabili sono i seguenti:
  - FILENAME - nome del file che ha soddisfatto la condizione di trigger
  - FILEPATH - percorso del file che soddisfa la condizione di trigger
  - FILESIZE (in byte - questi metadati potrebbero non essere presenti) - dimensione del file che soddisfa la condizione di trigger
  - LASTMODIFIEDDATE (Locale) - data in cui il file che ha soddisfatto la condizione trigger è stato modificato l'ultima volta. Tale data viene espressa come data locale nel fuso orario di esecuzione dell'agent e ha il formato data ISO 8601.
  - LASTMODIFIEDTIME (Locale) - ora in formato locale in cui il file che ha soddisfatto la condizione di trigger è stato modificato l'ultima volta. Tale ora viene espressa come ora locale nel fuso orario di esecuzione dell'agent e ha il formato orario ISO 8601.
  - LASTMODIFIEDDATEUTC - data in formato universale in cui il file che ha soddisfatto la condizione di trigger è stato modificato l'ultima volta. Tale data viene espressa come data locale convertita nel fuso orario UTC e ha il formato data ISO 8601.
  - LASTMODIFIEDTIMEUTC - ora in formato universale in cui il file che soddisfa la condizione di trigger è stato modificato l'ultima volta. Tale ora viene espressa come ora locale convertita nel fuso orario UTC e ha il formato orario ISO 8601.

  - AGENTNAME - il nome dell'agent di monitoraggio
- Una stringa XML che rappresenta l'attività da eseguire come risultato del trigger di controllo.

Le uscite di monitoraggio restituiscono i seguenti dati:

- Un indicatore che specifica se procedere ulteriormente (procedere o annullare)
- Una stringa da inserire nel messaggio di log soddisfatto dal trigger

Come risultato dell'esecuzione del codice di uscita del monitor, è possibile che siano stati aggiornati anche i metadati del monitor e la stringa XML di definizione dell'attività che erano stati originariamente passati come parametri.

Il valore della proprietà dell'agent monitorExitClasses (nel file `agent.properties`) specifica quali classi di uscita di monitoraggio caricare, con ciascuna classe di uscita separata da una virgola. Ad esempio:

```
monitorExitClasses=testExits.TestExit1,testExits.testExit2
```

L'interfaccia per l'uscita utente del monitor è:

```
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately prior to starting a task as the result of a monitor trigger
 */
public interface MonitorExit {

    /**
     * Invoked immediately prior to starting a task as the result of a monitor
     * trigger.
     *
     * @param environmentMetaData
     *            meta data about the environment in which the implementation
     *            of this method is running.  This information can only be read,
     *            it cannot be updated by the implementation.  The constant
     *            defined in <code>EnvironmentMetaDataConstants</code> class can
     *            be used to access the data held by this map.
     *
     * @param monitorMetaData
     *            meta data to associate with the monitor. The meta data passed
     *            to this method can be altered, and the changes will be
     *            reflected in subsequent exit routine invocations. This map
     *            also contains keys with IBM reserved names.  These entries are
     *            defined in the <code>MonitorMetaDataConstants</code> class and
     *            have special semantics. The the values of the IBM reserved names
     *            cannot be modified by the exit
     *
     * @param taskDetails
     *            An XML String representing the task to be executed as a result of
     *            the monitor triggering. This XML string may be modified by the
     *            exit
     *
     * @return    a monitor exit result object which is used to determine if the
     *            task should proceed, or be cancelled.
     */
    MonitorExitResult onMonitor(Map<String, String> environmentMetaData,
                                Map<String, String> monitorMetaData,
                                Reference<String> taskDetails);
}
```

Le costanti per i valori riservati IBMnei metadati del monitoraggio sono le seguenti:

```
package com.ibm.wmqfte.exitroutine.api;

/**
* Constants for IBM reserved values placed into the monitor meta data
* maps used by the monitor exit routines.
*/
public interface MonitorMetaDataConstants {

/**
* The value associated with this key is the name of the trigger
* file associated with the monitor. Any modification performed
* to this property by user exit routines will be ignored.
*/
final String FILE_NAME_KEY = "FILENAME";

/**
* The value associated with this key is the path to the trigger
* file associated with the monitor. Any modification performed
* to this property by user exit routines will be ignored.
*/
final String FILE_PATH_KEY = "FILEPATH";

/**
* The value associated with this key is the size of the trigger
* file associated with the monitor. This will not be present in
* the cases where the size cannot be determined. Any modification
* performed to this property by user exit routines will be ignored.
*/
final String FILE_SIZE_KEY = "FILESIZE";

/**
* The value associated with this key is the local date on which
* the trigger file associated with the monitor was last modified.
* Any modification performed to this property by user exit routines
* will be ignored.
*/
final String LAST_MODIFIED_DATE_KEY = "LASTMODIFIEDDATE";

/**
* The value associated with this key is the local time at which
* the trigger file associated with the monitor was last modified.
* Any modification performed to this property by user exit routines
* will be ignored.
*/
final String LAST_MODIFIED_TIME_KEY = "LASTMODIFIEDTIME";

/**
* The value associated with this key is the UTC date on which
* the trigger file associated with the monitor was last modified.
* Any modification performed to this property by user exit routines
* will be ignored.
*/
final String LAST_MODIFIED_DATE_KEY_UTC = "LASTMODIFIEDDATEUTC";

/**
* The value associated with this key is the UTC time at which
* the trigger file associated with the monitor was last modified.
* Any modification performed to this property by user exit routines
* will be ignored.
*/
final String LAST_MODIFIED_TIME_KEY_UTC = "LASTMODIFIEDTIMEUTC";

/**
* The value associated with this key is the name of the agent on which
* the monitor is running. Any modification performed to this property by
* user exit routines will be ignored.
*/
final String MONITOR_AGENT_KEY = "AGENTNAME";

}
```

Esempio di user exit di controllo

Questa classe di esempio implementa l'interfaccia MonitorExit . Questo esempio aggiunge una variabile di sostituzione personalizzata nei metadati di monitoraggio denominati *REDIRECTEDAGENT* che verranno

popolati con un valore di LONDON se l'ora del giorno è dispari e con un valore di PARIS per ore pari. Il codice di risultato dell'uscita del controllo è impostato per restituire sempre `proceed`.

```
package com.ibm.wmqfte.monitor;

import java.util.Calendar;
import java.util.Map;

import com.ibm.wmqfte.exitroutine.api.MonitorExit;
import com.ibm.wmqfte.exitroutine.api.MonitorExitResult;
import com.ibm.wmqfte.exitroutine.api.Reference;

/**
 * Example resource monitor user exit that changes the monitor mutable
 * metadata value between 'LONDON' and 'PARIS' depending on the hour of the day.
 *
 */
public class TestMonitorExit implements MonitorExit {

    // custom variable that will substitute destination agent
    final static String REDIRECTED_AGENT = "REDIRECTEDAGENT";

    public MonitorExitResult onMonitor(
Map<String, String> environmentMetaData,
            Map<String, String> monitorMetaData,
Reference<String> taskDetails) {

        // always succeed
        final MonitorExitResult result = MonitorExitResult.PROCEED_RESULT;

        final int hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);

        if (hour%2 == 1) {
            monitorMetaData.put(REDIRECTED_AGENT, "LONDON");
        } else {
            monitorMetaData.put(REDIRECTED_AGENT, "PARIS");
        }

        return result;
    }
}
```

L'attività corrispondente per un monitoraggio che utilizza la variabile di sostituzione *REDIRECTEDAGENT* potrebbe essere simile alla seguente:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
        <hostName>reportserver.com</hostName>
        <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT1"
                 QMgr="QM1"/>
    <destinationAgent agent="${REDIRECTEDAGENT}"
                      QMgr="QM2"/>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="delete">
          <file>c:\sourcefiles\reports.doc</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>c:\destinationfiles\reports.doc</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

Prima dell'avvio di questo trasferimento, il valore dell'attributo `agent` dell'elemento <destinationAgent> viene sostituito con LONDON o PARIS.

È necessario specificare la variabile di sostituzione nella classe di uscita del monitoraggio e l'XML di definizione attività in maiuscolo.

Concetti correlati

Personalizzare MFT con uscite utente

“Metadati per uscite utente MFT” a pagina 2180
Ci sono tre diversi tipi di metadati che possono essere forniti alle routine di uscita utente per Managed File Transfer: ambiente, trasferimento e metadati file. Questi metadati vengono presentati come associazioni di coppie chiave - valore Java .

“Interfacce Java per uscite utente MFT” a pagina 2190
Utilizzare gli argomenti in questa sezione per informazioni di riferimento sulle interfacce Java per le routine di uscita utente.

Uscite utente di origine e destinazione MFT

Riferimenti correlati

“MFT Proprietà dell'agent per uscite utente” a pagina 2188
Oltre alle proprietà standard nel file `agent.properties` sono disponibili diverse proprietà avanzate specifiche per le routine di uscita utente. Queste proprietà non sono incluse per impostazione predefinita, quindi se si desidera utilizzarle, è necessario modificare manualmente il file `agent.properties` . Se si apporta una modifica al file `agent.properties` mentre l'agent è in esecuzione, arrestare e riavviare l'agent per rendere effettive le modifiche.

MFT Proprietà dell'agent per uscite utente

Oltre alle proprietà standard nel file `agent.properties` sono disponibili diverse proprietà avanzate specifiche per le routine di uscita utente. Queste proprietà non sono incluse per impostazione predefinita, quindi se si desidera utilizzarle, è necessario modificare manualmente il file `agent.properties` . Se si apporta una modifica al file `agent.properties` mentre l'agent è in esecuzione, arrestare e riavviare l'agent per rendere effettive le modifiche.

Le variabili di ambiente possono essere utilizzate in alcune proprietà Managed File Transfer che rappresentano le ubicazioni di file o directory. Ciò consente alle ubicazioni dei file o delle directory utilizzate durante l'esecuzione di parti del prodotto, di variare in base alle modifiche dell'ambiente, ad esempio l'utente che sta eseguendo il processo. Per ulteriori informazioni, vedi Variabili di ambiente nelle proprietà MFT.

Proprietà della routine di uscita utente

Le routine di uscita utente vengono richiamate nell'ordine elencato nella seguente tabella. Per ulteriori informazioni sul file `agent.properties` , consultare Advanced agent properties: User exit routine.

Tabella 885. Proprietà agent per uscite utente

Nome proprietà	Descrizione
Classi sourceTransferEndExit	Specifica un elenco separato da virgole di classi che implementano una routine di uscita di fine trasferimento di origine.
Classi sourceTransferStartExit	Specifica un elenco separato da virgole di classi che implementano una routine di uscita di inizio trasferimento di origine.
Classi destinationTransferStartExit	Specifica un elenco separato da virgole di classi che implementano una routine di uscita utente di inizio trasferimento di destinazione.
Classi destinationTransferEndExit	Specifica un elenco separato da virgole di classi che implementano una routine di uscita utente di trasferimento di destinazione.

Tabella 885. Proprietà agent per uscite utente (Continua)	
Nome proprietà	**Descrizione**
Percorso exitClass	Specifica un elenco delimitato da caratteri, specifico della piattaforma, di indirizzari che fungono da percorso classi per le routine di uscita utente. La ricerca nella directory delle uscite dell'agent viene eseguita prima di qualsiasi voce in questo percorso di classe. Le parentesi, le virgole (,) e le barre retroverse (\) sono caratteri speciali nei comandi MFT e devono essere preceduti da un carattere barra retroversa (\). Windows I percorsi dei file su Windows possono essere specificati utilizzando doppie barre rovesciate (\\) come separatore o utilizzando singole barre (/). Ad esempio: `exitClassPath=C:/mycomp/mqft/exits/encryptFileExit.jar; C:/mycomp/mqft/exits/fileFilter.jar.` Il valore di questa proprietà può contenere variabili di ambiente.
exitNativeLibraryPath	Specifica un elenco delimitato da caratteri, specifico della piattaforma, di indirizzari che fungono da percorso della libreria nativa per le routine di uscita utente. Il valore di questa proprietà può contenere variabili di ambiente.
monitorExitClasses	Specifica un elenco separato da virgole di classi che implementano una routine di uscita di controllo. Per ulteriori informazioni, consultare Uscite utente di monitoraggio delle risorse MFT.
protocolBridgeCredentialExitClasses	Specifica un elenco separato da virgole di classi che implementano una routine uscita utente di credenziali del bridge di protocollo. Per ulteriori informazioni, consultare Associazione credenziali per un server file utilizzando le classi di uscita.
protocolBridgePropertiesExitClasses	Specifica un elenco separato da virgole di classi che implementano una routine uscita utente di proprietà del server bridge di protocollo. Per ulteriori informazioni, consultare ProtocolBridgePropertiesExit2: Looking up protocol file server properties.
IOExitClasses	Specifica un elenco separato da virgole di classi che implementano una routine uscita utente I/O. Elencare solo le classi che implementano l'interfaccia IOExit, ossia non elencare le classi che implementano le altre interfacce di uscita utente I/O, ad esempio IOExitResourcePath e IOExitChannel. Per ulteriori informazioni, consultare Utilizzo delle uscite utente I/O di trasferimento MFT.

Ordine di richiamo uscita

Le uscite di origine e destinazione vengono richiamate nel seguente ordine:

1. SourceTransferStartExit
2. DestinationTransferStartExit
3. DestinationTransferEndExit
4. SourceTransferEndExit

Concatenamento di uscite di origine e destinazione

Se si specificano più uscite, la prima uscita nell'elenco viene richiamata per prima, seguita dalla seconda uscita e così via. Tutte le modifiche apportate dalla prima uscita vengono passate come input all'uscita che viene successivamente richiamata e così via. Ad esempio, se ci sono due uscite di avvio del trasferimento di origine, tutte le modifiche apportate ai metadati del trasferimento dalla prima uscita vengono immesse nella seconda uscita. Ogni uscita restituisce il proprio risultato. Se tutte le uscite di un determinato tipo restituiscono PROCEED come codice di risultato trasferimento, il risultato complessivo è PROCEED. Se una o più uscite restituiscono CANCEL_TRANSFER, il risultato complessivo è CANCEL_TRANSFER. Tutti i codici di risultato e le stringhe restituiti dalle uscite vengono emessi nel log di trasferimento.

Se il risultato complessivo dell'uscita di inizio trasferimento di origine è PROCEED, il trasferimento procede utilizzando tutte le modifiche apportate dalle uscite. Se il risultato complessivo è CANCEL_TRANSFER, vengono richiamate le uscite di fine trasferimento di origine e il trasferimento viene annullato. Lo stato di completamento nel log di trasferimento è "annullato".

Se il risultato complessivo delle uscite di inizio trasferimento di destinazione è PROCEED, il trasferimento procede utilizzando le eventuali modifiche apportate dalle uscite. Se il risultato complessivo è CANCEL_TRANSFER, vengono richiamate le uscite di fine trasferimento di destinazione, quindi vengono richiamate le uscite di fine trasferimento di origine. Infine il trasferimento viene annullato. Lo stato di completamento nel log di trasferimento è "annullato".

Se un'uscita di origine o di destinazione deve passare le informazioni alle uscite successive nella catena o nell'ordine di esecuzione, deve essere eseguita aggiornando i metadati del trasferimento. L'utilizzo dei metadati di trasferimento è specifico dell'implementazione di uscita. Ad esempio, se un'uscita imposta il risultato di ritorno su CANCEL_TRANSFER e deve comunicare alle seguenti uscite che il trasferimento è stato annullato, deve essere eseguito impostando un valore di metadati del trasferimento in un modo compreso dalle altre uscite.

Esempio

```
sourceTransferStartExitClasses=com.ibm.wmqfte.test.MFTTestSourceTransferStartExit
sourceTransferEndExitClasses=com.ibm.wmqfte.test.MFTTestSourceTransferEndExit
destinationTransferStartExitClasses=com.ibm.wmqfte.test.MFTTestDestinationTransferStartExit
destinationTransferEndExitClasses=com.ibm.wmqfte.test.MFTTestDestinationTransferEndExit
exitClassPath=C:/mycomp/mqft/exits/encryptFileExit.jar;C:/mycomp/mqft/exits/fileFilter.jar
```

Concetti correlati

Personalizzare MFT con uscite utente

“Metadati per uscite utente MFT” a pagina 2180
Ci sono tre diversi tipi di metadati che possono essere forniti alle routine di uscita utente per Managed File Transfer: ambiente, trasferimento e metadati file. Questi metadati vengono presentati come associazioni di coppie chiave - valore Java .

“Interfacce Java per uscite utente MFT” a pagina 2190
Utilizzare gli argomenti in questa sezione per informazioni di riferimento sulle interfacce Java per le routine di uscita utente.

Riferimenti correlati

“Uscite utente del monitoraggio risorse MFT” a pagina 2184
Le uscite utente del monitoraggio risorse consentono di configurare codice personalizzato da eseguire quando viene soddisfatta una condizione di trigger del monitoraggio, prima che venga avviata l'attività associata.

Variabili di ambiente nelle proprietà MFT

Il file MFT `agent.properties`

Interfacce Java per uscite utente MFT

Utilizzare gli argomenti in questa sezione per informazioni di riferimento sulle interfacce Java per le routine di uscita utente.

Attività correlate

Personalizzare MFT con uscite utente

Riferimenti correlati

“Interfaccia DestinationTransferStartExit.java” a pagina 2194

“Interfaccia DestinationTransferEndExit.java” a pagina 2193

“Interfaccia IOExit.java” a pagina 2196

“Interfaccia IOExitChannel.java” a pagina 2198

“Interfaccia IOExitLock.java” a pagina 2200

Interfaccia CDCredentialExit.java

CDCredentialExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that are invoked as part of
 * user exit routine processing.  This interface defines methods that are
 * invoked by a Connect:Direct bridge agent to map the IBM MQ user ID of the transfer to credentials
 * that are used to access the Connect:Direct node.
 * There will be one instance of each implementation class per Connect:Direct bridge agent. The methods
 * can be called from different threads so the methods must be synchronized.
 */
public interface CDCredentialExit {

    /**
     * Invoked once when a Connect:Direct bridge agent is started. It is intended to initialize
     * any resources that are required by the exit
     *
     * @param bridgeProperties
     *            The values of properties defined for the Connect:Direct bridge.
     *            These values can only be read, they cannot be updated by
     *            the implementation.
     *
     * @return    true if the initialisation is successful and false if unsuccessful
     *            If false is returned from an exit the Connect:Direct bridge agent does not
     *            start.
     */
     public boolean initialize(final Map<String, String> bridgeProperties);

     /**
      * Invoked once per transfer to map the IBM MQ user ID in the transfer message to the
      * credentials to be used to access the Connect:Direct node.
       *
      * @param mqUserId The IBM MQ user ID from which to map to the credentials to be used
      *                 to access the Connect:Direct node
     * @param snode    The name of the Connect:Direct SNODE specified as the cdNode in the
     *                 file path. This is used to map the correct user ID and password for the
```

```
     *                 SNODE.
    * @return         A credential exit result object that contains the result of the map and
     *                 the credentials to use to access the Connect:Direct node
     */

    public CDCredentialExitResult mapMQUserId(final String mqUserId, final String snode);

     /**
     * Invoked once when a Connect:Direct bridge agent is shutdown. This method releases
     * any resources that were allocated by the exit
     *
      * @param bridgeProperties
     *            The values of properties defined for the Connect:Direct bridge.
     *            These values can only be read, they cannot be updated by
      *            the implementation.
      *
      * @return
    */
    public void shutdown(final Map<String, String> bridgeProperties);      }
```

Interfaccia CredentialExitResult.java

CredentialExitResult.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */

package com.ibm.wmqfte.exitroutine.api;

/**
 * The result of invoking a Credential mapMQUserId exit method.  It is composed of a result
 * code, which determines whether the mapping of the user id was successful, and an optional
 * Credentials object if the mapping is successful.
 */
public class CredentialExitResult {

    private final CredentialExitResultCode resultCode;
    private final Credentials credentials;

    /**
     * Constructor.  Creates a credential exit result object with a specified result
     * code and optionally credentials.
     *
     * @param resultCode
     *            The result code to associate with the exit result being created.
     *
     * @param credentials
     *            The credentials to associate with the exit result being created.
     *            A value of <code>null</code> can be specified to indicate no
     *            credentials. If the resultCode is USER_SUCCESSFULLY_MAPPED the
     *            credentials must be set to a non-null value,
     */
    public CredentialExitResult(CredentialExitResultCode resultCode, Credentials credentials) {
        this.resultCode = resultCode;
        this.credentials = credentials;
    }


    /**
     * Returns the result code associated with this credential exit result
     *
     * @return    the result code associated with this exit result.
     */
    public CredentialExitResultCode getResultCode() {
        return resultCode;
    }
```

```
    /**
     * Returns the credentials associated with this credential exit result
     *
     * @return    the explanation associated with this credential exit result.
     */
    public Credentials getCredentials() {
        return credentials;
    }
}
```

Attività correlate

Personalizzare MFT con uscite utente

Riferimenti correlati

“SourceTransferStartExit.java interface” a pagina 2219

“Interfaccia DestinationTransferStartExit.java” a pagina 2194

“Interfaccia DestinationTransferEndExit.java” a pagina 2193

“Interfaccia MonitorExit.java” a pagina 2212

“Interfaccia ProtocolBridgeCredentialExit.java” a pagina 2213

Interfaccia DestinationTransferEndExit.java

DestinationTransferEndExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately after completing a transfer on the agent acting as the
 * destination of the transfer.
 */
public interface DestinationTransferEndExit {

    /**
     * Invoked immediately after the completion of a transfer on the agent acting as
     * the destination of the transfer.
     *
     * @param transferExitResult
     *            a result object reflecting whether or not the transfer completed
     *            successfully.
     *
     * @param sourceAgentName
     *            the name of the agent acting as the source of the transfer.
     *
     * @param destinationAgentName
     *            the name of the agent acting as the destination of the
     *            transfer.     This is the name of the agent that the
     *            implementation of this method will be invoked from.
     *
     * @param environmentMetaData
     *            meta data about the environment in which the implementation
     *            of this method is running.  This information can only be read,
     *            it cannot be updated by the implementation.  The constants
     *            defined in <code>EnvironmentMetaDataConstants</code> class can
     *            be used to access the data held by this map.
     *
     * @param transferMetaData
     *            meta data to associate with the transfer.  The information can
     *            only be read, it cannot be updated by the implementation.  This
```

```
     *            map may also contain keys with IBM reserved names.  These
     *            entries are defined in the <code>TransferMetaDataConstants</code>
     *            class and have special semantics.
     *
     * @param fileResults
     *            a list of file transfer result objects that describe the source
     *            file name, destination file name and result of each file transfer
     *            operation attempted.
     *
     * @return    an optional description to enter into the log message describing
     *            transfer completion.  A value of <code>null</code> can be used
     *            when no description is required.
     */
    String onDestinationTransferEnd(TransferExitResult transferExitResult,
                    String sourceAgentName,
                    String destinationAgentName,
                    Map<String, String>environmentMetaData,
                    Map<String, String>transferMetaData,
                    List<FileTransferResult>fileResults);
}
```

Attività correlate

Personalizzare MFT con uscite utente

Riferimenti correlati

“SourceTransferStartExit.java interface” a pagina 2219

“Interfaccia SourceTransferEndExit.java” a pagina 2218

“Interfaccia DestinationTransferStartExit.java” a pagina 2194

“Interfaccia MonitorExit.java” a pagina 2212

“Interfaccia ProtocolBridgeCredentialExit.java” a pagina 2213

Interfaccia DestinationTransferStartExit.java

DestinationTransferStartExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately prior to starting a transfer on the agent acting as the
 * destination of the transfer.
 */
public interface DestinationTransferStartExit {

    /**
     * Invoked immediately prior to starting a transfer on the agent acting as
     * the destination of the transfer.
     *
     * @param sourceAgentName
     *            the name of the agent acting as the source of the transfer.
     *
     * @param destinationAgentName
     *            the name of the agent acting as the destination of the
     *            transfer.     This is the name of the agent that the
     *            implementation of this method will be invoked from.
     *
     * @param environmentMetaData
     *            meta data about the environment in which the implementation
     *            of this method is running.  This information can only be read,
```

```
     *            it cannot be updated by the implementation.  The constants
     *            defined in <code>EnvironmentMetaDataConstants</code> class can
     *            be used to access the data held by this map.
     *
     * @param transferMetaData
     *            meta data to associate with the transfer.  The information can
     *            only be read, it cannot be updated by the implementation.  This
     *            map may also contain keys with IBM reserved names.  These
     *            entries are defined in the <code>TransferMetaDataConstants</code>
     *            class and have special semantics.
     *
     * @param fileSpecs
     *            a list of file specifications that govern the file data to
     *            transfer.  The implementation of this method can modify the
     *            entries in this list and the changes will be reflected in the
     *            files transferred. However, new entries may not be added and
     *            existing entries may not be removed.
     *
     * @return    a transfer exit result object which is used to determine if the
     *            transfer should proceed, or be cancelled.
     */
    TransferExitResult onDestinationTransferStart(String sourceAgentName,
                                                  String destinationAgentName,
                                            Map<String, String> environmentMetaData,
                                            Map<String, String> transferMetaData,
                                            List<Reference<String>> fileSpecs);
```

Attività correlate

Personalizzare MFT con uscite utente

Riferimenti correlati

“SourceTransferStartExit.java interface” a pagina 2219

“Interfaccia SourceTransferEndExit.java” a pagina 2218

“Interfaccia DestinationTransferEndExit.java” a pagina 2193

“Interfaccia MonitorExit.java” a pagina 2212

“Interfaccia ProtocolBridgeCredentialExit.java” a pagina 2213

Interfaccia FileTransferResult.java

FileTransferResult.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */

package com.ibm.wmqfte.exitroutine.api;

/**
 * Result information about a file transfer.
 */
public interface FileTransferResult {

    /** An enumeration for the <code>getCorrelatorType()</code> method. */
    public enum CorrelationInformationType {
        /** No correlation information is available for this result */
        NONE,
        /**
         * The correlation information relates to work done in
         * IBM Sterling File Gateway.
         */
        SFG
    }
```

```
    /**
     * Returns the source file specification, from which the file was transferred.
     *
     * @return    the source file specification, from which the file was
     *            transferred.
     */
    String getSourceFileSpecification();

    /**
     * Returns the destination file specification, to which the file was transferred.
     *
     * @return    the destination file specification, to which the file was
     *            transferred.  A value of <code>null</code> may be returned
     *            if the transfer did not complete successfully.
     */
    String getDestinationFileSpecification();

    /**
     * Returns the result of the file transfer operation.
     *
     * @return    the result of the file transfer operation.
     */
    FileExitResult getExitResult();

    /**
     * @return an enumerated value that identifies the product to which this correlating
     *         information relates.
     */
    CorrelationInformationType getCorrelatorType();

    /**
     * @return the first string component of the correlating identifier that relates
     *         this transfer result to work done in another product.  A value of null
     *         may be returned either because the other product does not utilize a
     *         string based correlation information or because there is no correlation
     *         information.
     */
    String getString1Correlator();

    /**
     * @return the first long component of the correlating identifier that relates
     *         this transfer result to work done in another product.  A value of zero
     *         is returned when there is no correlation information or the other
     *         product does not utilize long based correlation information or because
     *         the value really is zero!
     */
    long getLong1Correlator();
}
```

Attività correlate

Personalizzare MFT con uscite utente

Riferimenti correlati

“SourceTransferStartExit.java interface” a pagina 2219

“Interfaccia DestinationTransferStartExit.java” a pagina 2194

“Interfaccia DestinationTransferEndExit.java” a pagina 2193

“Interfaccia MonitorExit.java” a pagina 2212

“Interfaccia ProtocolBridgeCredentialExit.java” a pagina 2213

Interfaccia IOExit.java

IOExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
```

```
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.util.Map;

import com.ibm.wmqfte.exitroutine.api.IOExitRecordResourcePath.RecordFormat;

/**
 * An interface that is implemented by classes that you want to be invoked as
 * part of user exit routine processing. This interface defines methods that
 * will be invoked during transfers to perform the underlying file system I/O
 * work for WMQFTE transfers.
 * <p>
 * The {@link #initialize(Map)} method will be called once when the exit is
 * first installed. The WMQFTE agent properties are passed to this method, thus
 * enabling the exit to understand its environment.
 * <p>
 * The {@link #isSupported(String)} method will be invoked during WMQFTE
 * transfers to determine whether the user exit should be used. If the
 * {@link #isSupported(String)} method returns a value of {@code true}, the
 * {@link #newPath(String)} method will be invoked for the paths specified for
 * the transfer request. The returned {@link IOExitPath} instance from a
 * {@link #newPath(String)} method invocation will then be used by the WMQFTE
 * transfer to obtain information about the resource and to transfer data to or
 * from the resource.
 * <p>
 * To obtain transfer context for an I/O exit, a {@link SourceTransferStartExit}
 * or {@link DestinationTransferStartExit} as appropriate, should be installed
 * to enable information to be seen by this exit. The
 * {@link SourceTransferStartExit} or {@link DestinationTransferStartExit} are
 * passed the transfer's environment, metadata, and a list of file
 * specifications for the transfer. The paths for the file specifications are
 * the paths passed to the I/O exit's {@link #newPath(String)} method.
 * <p>
 * Note also that the {@link #isSupported(String)} and {@link #newPath(String)}
 * methods might be called at other times by a WMQFTE agent and not just during
 * transfers. For example, at transfer setup time the I/O system is queried to
 * resolve the full resource paths for transfer.
 */
public interface IOExit {

    /**
     * Invoked once when the I/O exit is first required for use. It is intended
     * to initialize any resources that are required by the exit.
     *
     * @param agentProperties
     *            The values of properties defined for the WMQFTE agent. These
     *            values can only be read, they cannot be updated by the
     *            implementation.
     * @return {@code true} if the initialization is successful and {@code
     *         false} if unsuccessful. If {@code false} is returned from an
     *         exit, the exit will not be used.
     */
    boolean initialize(final Map<String, String> agentProperties);

    /**
     * Indicates whether this I/O user exit supports the specified path.
     * <p>
     * This method is used by WMQFTE to determine whether the I/O user exit
     * should be used within a transfer. If no I/O user exit returns true for
     * this method, the default WMQFTE file I/O function will be used.
     *
     * @param path
     *            The path to the required I/O resource.
     * @return {@code true} if the specified path is supported by the I/O exit,
     *         {@code false} otherwise
     */
    boolean isSupported(String path);

    /**
     * Obtains a new {@link IOExitPath} instance for the specified I/O resource
     * path.
     * <p>
     * This method will be invoked by WMQFTE only if the
     * {@link #isSupported(String)} method has been called for the path and
     * returned {@code true}.
     *
     * @param path
     *            The path to the required I/O resource.
```

```
     * @return A {@link IOExitPath} instance for the specified path.
     * @throws IOException
     *             If the path cannot be created for any reason.
     */
    IOExitPath newPath(String path) throws IOException;

    /**
     * Obtains a new {@link IOExitPath} instance for the specified I/O resource
     * path and passes record format and length information required by the
     * WMQFTE transfer.
     * <p>
     * Typically this method will be called for the following cases:
     * <ul>
     * <li>A path where a call to {@link #newPath(String)} has previously
     * returned a {@link IOExitRecordResourcePath} instance and WMQFTE is
     * re-establishing a new {@link IOExitPath} instance for the path, from an
     * internally-serialized state. The passed recordFormat and recordLength
     * will be the same as those for the original
     * {@link IOExitRecordResourcePath} instance.</li>
     * <li>A transfer destination path where the source of the transfer is
     * record oriented. The passed recordFormat and recordLength will be the
     * same as those for the source.</li>
     * </ul>
     * The implementation can act on the record format and length information as
     * deemed appropriate. For example, for a destination agent if the
     * destination does not already exist and the source of the transfer is
     * record oriented, the passed recordFormat and recordLength information
     * could be used to create an appropriate record-oriented destination path.
     * If the destination path already exists, the passed recordFormat and
     * recordLength information could be used to perform a compatibility check
     * and throw an {@link IOException} if the path is not compatible. A
     * compatibility check could ensure that a record oriented path's record
     * format is the same as the passed record format or that the record length
     * is greater or equal to the passed record length.
     * <p>
     * This method will be invoked by WMQFTE only if the
     * {@link #isSupported(String)} method has been called for the path and
     * returned {@code true}.
     *
     * @param path
     *            The path to the required I/O resource.
     * @param recordFormat
     *            The advised record format.
     * @param recordLength
     *            The advised record length.
     * @return A {@link IOExitPath} instance for the specified path.
     * @throws IOException
     *             If the path cannot be created for any reason. For example,
     *             the passed record format or length is incompatible with the
     *             path's actual record format or length.
     */
    IOExitPath newPath(String path, RecordFormat recordFormat, int recordLength)
            throws IOException;
```

Attività correlate

Utilizzo delle uscite utente I/O di trasferimento MFT

Personalizzare MFT con uscite utente

Interfaccia IOExitChannel.java

IOExitChannel.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   © Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;
```

```
import java.io.IOException;
import java.nio.ByteBuffer;

/**
 * Represents a channel that enables data to be read from or written to an
 * {@link IOExitResourcePath} resource.
 */
public interface IOExitChannel {

    /**
     * Obtains the data size for the associated {@link IOExitResourcePath} in
     * bytes.
     *
     * @return The data size in bytes.
     * @throws IOException
     *             If a problem occurs while attempting obtain the size.
     */
    long size() throws IOException;

    /**
     * Closes the channel, flushing any buffered write data to the resource and
     * releasing any locks.
     *
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while closing the resource.
     *             This means that WMQFTE can attempt to recover the transfer.
     * @throws IOException
     *             If some other I/O problem occurs. For example, the channel might
     *             already be closed.
     */
    void close() throws RecoverableIOException, IOException;

    /**
     * Reads data from this channel into the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data read.
     * <p>
     * Data is copied into the buffer starting at its current position and up to
     * its limit. On return, the buffer's position is updated to reflect the
     * number of bytes read.
     *
     * @param buffer
     *            The buffer that the data is to be copied into.
     * @return The number of bytes read, which might be zero, or -1 if the end of
     *         data has been reached.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while reading the data. For a
     *             WMQFTE transfer this means that it will attempt to recover.
     * @throws IOException
     *             If some other I/O problem occurs. For a WMQFTE transfer this
     *             means that it will be failed.
     */
    int read(ByteBuffer buffer) throws RecoverableIOException, IOException;

    /**
     * Writes data to this channel from the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data written. The channel's resource is grown to accommodate
     * the data, if necessary.
     * <p>
     * Data is copied from the buffer starting at its current position and up to
     * its limit. On return, the buffer's position is updated to reflect the
     * number of bytes written.
     *
     * @param buffer
     *            The buffer containing the data to be written.
     * @return The number of bytes written, which might be zero.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while writing the data. For a
     *             WMQFTE transfer this means that it will attempt to recover.
     * @throws IOException
     *             If some other I/O problem occurs. For a WMQFTE transfer this
     *             means that it will be failed.
     */
    int write(ByteBuffer buffer) throws RecoverableIOException, IOException;

    /**
     * Forces any updates to this channel's resource to be written to its
     * storage device.
     * <p>
     * This method is required to force changes to both the resource's content
```

```
     * and any associated metadata to be written to storage.
     *
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while performing the force.
     *             For a WMQFTE transfer this means that it will attempt to
     *             recover.
     * @throws IOException
     *             If some other I/O problem occurs. For a WMQFTE transfer this
     *             means that it will be failed.
     */
    void force() throws RecoverableIOException, IOException;

    /**
     * Attempts to lock the entire resource associated with the channel for
     * shared or exclusive access.
     * <p>
     * The intention is for this method not to block if the lock is currently
     * unavailable.
     *
     * @param shared
     *            {@code true} if a shared lock is required, {@code false} if an
     *            exclusive lock is required.
     * @return A {@link IOExitLock} instance representing the newly acquired
     *         lock or null if the lock cannot be obtained.
     * @throws IOException
     *             If a problem occurs while attempting to acquire the lock.
     */
    IOExitLock tryLock(boolean shared) throws IOException;
}
```

Attività correlate

Utilizzo delle uscite utente I/O di trasferimento MFT

Personalizzare MFT con uscite utente

Interfaccia IOExitLock.java

IOExitLock.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a lock on a resource for either shared or exclusive access.
 * {@link IOExitLock} instances are returned from
 * {@link IOExitChannel#tryLock(boolean)} calls and WMQFTE will request the
 * release of the lock at the appropriate time during a transfer. Additionally, when
 * a {@link IOExitChannel#close()} method is called it will be the
 * responsibility of the channel to release any associated locks.
 */
public interface IOExitLock {

    /**
     * Releases the lock.
     * <p>
     * After this method has been successfully called the lock is to be deemed as invalid.
     *
     * @throws IOException
     *             If the channel associated with the lock is not open or
     *             another problem occurs while attempting to release the lock.
     */
    void release() throws IOException;
```

```
    /**
     * Indicates whether this lock is valid.
     * <p>
     * A lock is considered valid until its @ {@link #release()} method is
     * called or the associated {@link IOExitChannel} is closed.
     *
     * @return {@code true} if this lock is valid, {@code false} otherwise.
     */
    boolean isValid();

    /**
     * @return {@code true} if this lock is for shared access, {@code false} if
     *         this lock is for exclusive access.
     */
    boolean isShared();
}
```

Attività correlate

Utilizzo delle uscite utente I/O di trasferimento MFT

Personalizzare MFT con uscite utente

Interfaccia IOExitPath.java

IOExitPath.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Represents an abstract path that can be inspected and queried by WMQFTE for
 * transfer purposes.
 * <p>
 * There are two types of path supported:
 * <ul>
 * <li>{@link IOExitResourcePath} - Represents a path that denotes a data
 * resource. For example, a file, directory, or group of database records.</li>
 * <li>{@link IOExitWildcardPath} - Represents a wildcard path that can be
 * expanded to multiple {@link IOExitResourcePath} instances.</li>
 * </ul>
 */
public abstract interface IOExitPath {

    /**
     * Obtains the abstract path as a {@link String}.
     *
     * @return The abstract path as a {@link String}.
     */
    String getPath();

    /**
     * Obtains the name portion of this abstract path as a {@link String}.
     * <p>
     * For example, a UNIX-style file system implementation evaluates the
     * path {@code /home/fteuser/file1.txt} as having a name of {@code
     * file1.txt}.
     *
     * @return the name portion of this abstract path as a {@link String}.
     */
    String getName();

    /**
     * Obtains the parent path for this abstract path as a {@link String}.
     * <p>
     * For example, a UNIX-style file system implementation evaluates the
```

```
     * path {@code /home/fteuser/file1.txt} as having a parent path of {@code
     * /home/fteuser}.
     *
     * @return The parent portion of the path as a {@link String}.
     */
    String getParent();

    /**
     * Obtains the abstract paths that match this abstract path.
     * <p>
     * If this abstract path denotes a directory resource, a list of paths
     * for all resources within the directory are returned.
     * <p>
     * If this abstract path denotes a wildcard, a list of all paths
     * matching the wildcard are returned.
     * <p>
     * Otherwise null is returned, because this abstract path probably denotes a
     * single file resource.
     *
     * @return An array of {@IOExitResourcePath}s that
     *         match this path, or null if this method is not applicable.
     */
    IOExitResourcePath[] listPaths();
}
```

Attività correlate

Utilizzo delle uscite utente I/O di trasferimento MFT

Personalizzare MFT con uscite utente

Interfaccia IOExitProperties.java

IOExitProperties.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Properties that determine how WMQFTE treats an {@link IOExitPath} for certain
 * aspects of I/O. For example, whether to use intermediate files.
 */
public class IOExitProperties {

    private boolean rereadSourceOnRestart = true;
    private boolean rechecksumSourceOnRestart = true;
    private boolean rechecksumDestinationOnRestart = true;
    private boolean useIntermediateFileAtDestination = true;
    private boolean requiresSingleThreadedChannelIO = false;

    /**
     * Determines whether the I/O exit implementation expects the resource to be
     * re-read from the start if a transfer is restarted.
     *
     * @return {@code true} if, on restart, the I/O exit expects the source
     *         resource to be opened at the beginning and re-read from the
     *         beginning (the {@link IOExitPath#openForRead(long)} method is
     *         always invoked with 0L as an argument). {@code false} if, on
     *         restart, the I/O exit expects the source to be opened at the
     *         offset that the source agent intends to start reading from (the
     *         {@link IOExitPath#openForRead(long)} method can be invoked with a
     *         non-zero value as its argument).
     */
    public boolean getRereadSourceOnRestart() {
        return rereadSourceOnRestart;
    }
```

```
/**
 * Sets the value to determine whether the I/O exit implementation expects
 * the resource to be re-read from the beginning if a transfer is restarted.
 * <p>
 * The default is {@code true}. The I/O exit should call this method when
 * required to change this value.
 *
 * @param rereadSourceOnRestart
 *            {@code true} if, on restart, the I/O exit expects the source
 *            resource to be opened at the beginning and re-read from the
 *            beginning (the {@link IOExitPath#openForRead(long)} method
 *            is always invoked with 0L as an argument). {@code false}
 *            if, on restart, the I/O exit expects the source to be opened
 *            at the offset that the source agent intends to start reading
 *            from (the {@link IOExitPath#openForRead(long)} method can be
 *            invoked with a non-zero value as its argument).
 */
public void setRereadSourceOnRestart(boolean rereadSourceOnRestart) {
    this.rereadSourceOnRestart = rereadSourceOnRestart;
}

/**
 * Determines whether the I/O exit implementation requires the source
 * resource to be re-checksummed if the transfer is restarted.
 * Re-checksumming takes place only if the
 * {@link #getRereadSourceOnRestart()} method returns {@code true}.
 *
 * @return {@code true} if, on restart, the I/O exit expects the already-
 *         transferred portion of the source to be re-checksummed for
 *         inconsistencies. Use this option in environments
 *         where the source could be changed during a restart. {@code
 *         false} if, on restart, the I/O exit does not require the
 *         already-transferred portion of the source to be re-checksummed.
 */
public boolean getRechecksumSourceOnRestart() {
    return rechecksumSourceOnRestart;
}

/**
 * Sets the value to determine whether the I/O exit implementation requires
 * the source resource to be re-checksummed if the transfer is restarted.
 * Re-checksumming takes place only if the
 * {@link #getRereadSourceOnRestart()} method returns {@code true}.
 * <p>
 * The default is {@code true}. The I/O exit should call this method when
 * required to change this value.
 *
 * @param rechecksumSourceOnRestart
 *            {@code true} if, on restart, the I/O exit expects the already
 *            transferred portion of the source to be re-checksummed
 *            for inconsistencies. Use this option in environments
 *            where the source could be changed during a restart.
 *            {@code false} if, on restart, the I/O exit does not
 *            require the already-transferred portion of the source to be
 *            re-checksummed.
 */
public void setRechecksumSourceOnRestart(boolean rechecksumSourceOnRestart) {
    this.rechecksumSourceOnRestart = rechecksumSourceOnRestart;
}

/**
 * Determines whether the I/O exit implementation requires the destination
 * resource to be re-checksummed if the transfer is restarted.
 *
 * @return {@code true} if, on restart, the I/O exit expects the already
 *         transferred portion of the destination to be re-checksummed to
 *         check for inconsistencies. This option should be used in
 *         environments where the destination could have been changed while
 *         a restart is occurring. {@code false} if, on restart, the I/O exit
 *         does not require the already transferred portion of the
 *         destination to be re-checksummed.
 */
public boolean getRechecksumDestinationOnRestart() {
    return rechecksumDestinationOnRestart;
}

/**
 * Sets the value to determine whether the I/O exit implementation requires
 * the destination resource to be re-checksummed if the transfer is
 * restarted.
 * <p>
```

```
     * The default is {@code true}. The I/O exit should call this method when
     * required to change this value.
     *
     * @param rechecksumDestinationOnRestart
     *            {@code true} if, on restart, the I/O exit expects the already-
     *            transferred portion of the destination to be re-checksummed
     *            for inconsistencies. Use this option in environments
     *            where the destination could have been changed during a
     *            restart. {@code false} if, on restart, the I/O exit does not
     *            require the already-transferred portion of the destination
     *            to be re-checksummed.
     */
    public void setRechecksumDestinationOnRestart(
            boolean rechecksumDestinationOnRestart) {
        this.rechecksumDestinationOnRestart = rechecksumDestinationOnRestart;
    }

    /**
     * Determines whether the I/O exit implementation requires the use of an
     * intermediate file when writing the data at the destination. The
     * intermediate file mechanism is typically used to prevent an incomplete
     * destination resource from being processed.
     *
     * @return {@code true} if data should be written to an intermediate file at
     *         the destination and then renamed (to the requested destination
     *         path name as specified in the transfer request) after the transfer is
     *         complete. {@code false} if data should be written directly to the
     *         requested destination path name without the use of an
     *         intermediate file.
     */
    public boolean getUseIntermediateFileAtDestination() {
        return useIntermediateFileAtDestination;
    }

    /**
     * Sets the value to determine whether the I/O exit implementation requires
     * the use of an intermediate file when writing the data at the destination.
     * The intermediate file mechanism is typically used to prevent an
     * incomplete destination resource from being processed.
     *
     * <p>
     * The default is {@code true}. The I/O exit should call this method when
     * required to change this value.
     *
     * @param useIntermediateFileAtDestination
     *            {@code true} if data should be written to an intermediate file
     *            at the destination and then renamed (to the requested
     *            destination path name as specified in the transfer request) after
     *            the transfer is complete. {@code false} if data should be written
     *            directly to the requested destination path name without the
     *            use of an intermediate file
     */
    public void setUseIntermediateFileAtDestination(
            boolean useIntermediateFileAtDestination) {
        this.useIntermediateFileAtDestination = useIntermediateFileAtDestination;
    }

    /**
     * Determines whether the I/O exit implementation requires
     * {@link IOExitChannel} instances to be accessed by a single thread only.
     *
     * @return {@code true} if {@link IOExitChannel} instances are to be
     *         accessed by a single thread only.
     */
    public boolean requiresSingleThreadedChannelIO() {
        return requiresSingleThreadedChannelIO;
    }

    /**
     * Sets the value to determine whether the I/O exit implementation requires
     * channel operations for a particular instance to be accessed by a
     * single thread only.
     * <p>
     * For certain I/O implementations it is necessary that resource path
     * operations such as open, read, write, and close are invoked only from a
     * single execution {@link Thread}. When set {@code true}, WMQFTE ensures
     * that the following are invoked on a single thread:
     * <ul>
     * <li>{@link IOExitResourcePath#openForRead(long) method and all methods of
     * the returned {@link IOExitChannel} instance.</li>
     * <li>{@link IOExitResourcePath#openForWrite(boolean)) method and all
     * methods of the returned {@link IOExitChannel} instance.</li>
```

```
     * </ul>
     * <p>
     * This has a slight performance impact, hence enable single-threaded channel
     * I/O only when absolutely necessary.
     * <p>
     * The default is {@code false}. The I/O exit should call this method when
     * required to change this value.
     *
     * @param requiresSingleThreadedChannelIO
     *            {@code true} if {@link IOExitChannel} instances are to be
     *            accessed by a single thread only.
     */
    public void setRequiresSingleThreadedChannelIO(boolean requiresSingleThreadedChannelIO) {
        this.requiresSingleThreadedChannelIO = requiresSingleThreadedChannelIO;
    }
}
```

Attività correlate

Utilizzo delle uscite utente I/O di trasferimento MFT

Personalizzare MFT con uscite utente

Interfaccia IOExitRecordChannel.java

IOExitRecordChannel.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.nio.ByteBuffer;

/**
 * Represents a channel that enables records of data to be read from or written
 * to an {@link IOExitRecordResourcePath} resource.
 * <p>
 * This is an extension of the {@link IOExitChannel} interface such that the
 * {@link #read(java.nio.ByteBuffer)} and {@link #write(java.nio.ByteBuffer)}
 * methods are expected to deal in whole records of data only. That is, the
 * {@link java.nio.ByteBuffer} returned from the read method and passed to the
 * write method is assumed to contain one or more complete records.
 */
public interface IOExitRecordChannel extends IOExitChannel {

    /**
     * Reads records from this channel into the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data read.
     * <p>
     * Record data is copied into the buffer starting at its current position
     * and up to its limit. On return, the buffer's position is updated to
     * reflect the number of bytes read.
     * <p>
     * Only whole records are copied into the buffer.
     * <p>
     * For a fixed-record-format resource, this might be multiple records. The
     * amount of data in the return buffer does not necessarily need to be a
     * multiple of the record length, but the last record is still to be treated
     * as a complete record and padded as required by the caller.
     * <p>
     * For a variable-format resource, this is a single whole record of a size
     * corresponding to the amount of return data or multiple whole records with
     * all except the last being treated as records of maximum size.
     *
     * @param buffer
```

```
     *            The buffer that the record data is to be copied into.
     * @return The number of bytes read, which might be zero, or -1 if the end of
     *         data has been reached.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while reading the data. For a
     *             WMQFTE transfer this means that it will attempt to recover.
     * @throws IOException
     *             If some other I/O problem occurs, for example, if the passed
     *             buffer is insufficient to contain at least one complete
     *             record). For a WMQFTE transfer this means that it will be
     *             failed.
     */
    int read(ByteBuffer buffer) throws RecoverableIOException, IOException;

    /**
     * Writes records to this channel from the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data written. The channel's resource is grown to accommodate
     * the data, if necessary.
     * <p>
     * Record data is copied from the buffer starting at its current position
     * and up to its limit. On return, the buffer's position is updated to
     * reflect the number of bytes written.
     * <p>
     * The buffer is expected to contain only whole records.
     * <p>
     * For a fixed-record-format resource, this might be multiple records and if
     * there is insufficient data in the buffer for a complete record, the
     * record is to be padded as required to complete the record.
     * <p>
     * For a variable-record format resource the buffer is normally expected to
     * contain a single record of length corresponding to the amount of data
     * within the buffer. However, if the amount of data within the buffer
     * exceeds the maximum record length, the implementation can either:
     * <ol>
     * <li>throw an {@link IOException} indicating that it cannot handle the
     * situation.</li>
     * <li>Consume a record's worth of data from the buffer, leaving the remaining
     * data within the buffer.</li>
     * <li>Consume all the buffer data and just write what it can to the current
     * record. This effectively truncates the data.</li>
     * <li>Consume all the buffer data and write to multiple records.</li>
     * </ol>
     *
     * @param buffer
     *            The buffer containing the data to be written.
     * @return The number of bytes written, which might be zero.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while writing the data. For a
     *             WMQFTE transfer this means that it will attempt to recover.
     * @throws IOException
     *             If some other I/O problem occurs. For a WMQFTE transfer this
     *             means that it will be failed.
     */
    int write(ByteBuffer buffer) throws RecoverableIOException, IOException;

}
```

Attività correlate

Utilizzo delle uscite utente I/O di trasferimento MFT

Personalizzare MFT con uscite utente

z/OS *IOExitRecordResourcePath.java interface*

IOExitRecordResourcePath.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
```

```
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a path that denotes a record-oriented data resource (for example,
 * a z/OS data set). It allows the data to be located, the record format to be
 * understood, and {@link IOExitRecordChannel} instances to be created for read
 * or write operations.
 */
public interface IOExitRecordResourcePath extends IOExitResourcePath {

    /**
     * Record formats for record-oriented resources.
     */
    public enum RecordFormat {
        FIXED, VARIABLE
    }

    /**
     * Obtains the record length for records that are maintained by the resource
     * denoted by this abstract path.
     * <p>
     * For a resource with fixed-length records, the data for each record read
     * and written is assumed to be this length.
     * <p>
     * For a resource with variable-length records, this is the maximum length
     * for a record's data.
     * <p>
     * This method should return a value greater than zero, otherwise it can
     * result in the failure of a WMQFTE transfer that involves this abstract
     * path.
     *
     * @return The record length, in bytes, for records maintained by the
     *         resource.
     */
    int getRecordLength();

    /**
     * Obtains record format, as a {@link RecordFormat} instance, for records
     * that are maintained by the resource denoted by this abstract path.
     *
     * @return A {@link RecordFormat} instance for the record format for records
     *         that are maintained by the resource denoted by this abstract
     *         path.
     */
    RecordFormat getRecordFormat();

    /**
     * Opens a {@link IOExitRecordChannel} instance for reading data from the
     * resource denoted by this abstract path. The current data byte position
     * for the resource is expected to be the passed position value, such that
     * when {@link IOExitRecordChannel#read(java.nio.ByteBuffer)} is called,
     * data starting from that position is read.
     * <p>
     * Note that the data byte read position will be on a record boundary.
     *
     * @param position
     *            The required data byte read position.
     * @return A new {@link IOExitRecordChannel} instance allowing data to be
     *         read from the resource denoted by this abstract path.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while attempting to open the
     *             resource for reading. This means that WMQFTE can attempt to
     *             recover the transfer.
     * @throws IOException
     *             If some other I/O problem occurs.
     */
    IOExitRecordChannel openForRead(long position)
            throws RecoverableIOException, IOException;

    /**
     * Opens a {@link IOExitRecordChannel} instance for writing data to the
     * resource denoted by this abstract path. Writing of data, using the
     * {@link IOExitRecordChannel#write(java.nio.ByteBuffer)} method, starts at
     * either the beginning of the resource or end of the current data for the
     * resource, depending on the specified append parameter.
     *
     * @param append
```

```
     *            When {@code true} indicates that data written to the resource
     *            should be appended to the end of the current data. When
     *            {@code false} indicates that writing of data is to start at
     *            the beginning of the resource; any existing data is lost.
     * @return A new {@link IOExitRecordChannel} instance allowing data to be
     *         written to the resource denoted by this abstract path.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while attempting to open the
     *             resource for writing. This means that WMQFTE can attempt to
     *             recover the transfer.
     * @throws IOException
     *             If some other I/O problem occurs.
     */
    IOExitRecordChannel openForWrite(boolean append)
            throws RecoverableIOException, IOException;
}
```

Related tasks

Using MFT transfer I/O user exits

Customizing MFT with user exits

Interfaccia IOExitResourcePath.java

IOExitResourcePath.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a path that denotes a data resource (for example, a file,
 * directory, or group of database records). It allows the data to be located
 * and {@link IOExitChannel} instances to be created for read or write
 * operations.
 * <p>
 * There are two types of data resources as follows:
 * <ul>
 * <li>Directory - a container for other data resources. The
 * {@link #isDirectory()} method returns {@code true} for these.</li>
 * <li>File - a data container. This allows data to be read from or written to
 * it. The {@link #isFile()} method returns {@code true} for these.</li>
 * </ul>
 */
public interface IOExitResourcePath extends IOExitPath {

    /**
     * Creates a new {@link IOExitResourcePath} instance for a child path of the
     * resource denoted by this abstract path.
     * <p>
     * For example, with a UNIX-style path, {@code
     * IOExitResourcePath("/home/fteuser/test").newPath("subtest")} could be
     * equivalent to: {@code IOExitResourcePath("/home/fteuser/test/subtest")}
     *
     * @param child
     *            The child path name.
     * @return A new {@link IOExitResourcePath} instance that represents a child
     *         of this path.
     */
    IOExitResourcePath newPath(final String child);

    /**
     * Creates the directory path for the resource denoted by this abstract
     * path, including any necessary but nonexistent parent directories. If the
```

```
     * directory path already exists, this method has no effect.
     * <p>
     * If this operation fails, it might have succeeded in creating some of the
     * necessary parent directories.
     *
     * @throws IOException
     *             If the directory path cannot be fully created, when it does
     *             not already exist.
     */
    void makePath() throws IOException;

    /**
     * Obtains the canonical path of the abstract path as a {@link String}.
     * <p>
     * A canonical path is defined as being absolute and unique. For example,
     * the path can be represented as UNIX-style relative path: {@code
     * test/file.txt} but the absolute and unique canonical path representation
     * is: {@code /home/fteuser/test/file.txt}
     *
     * @return The canonical path as a {@link String}.
     * @throws IOException
     *             If the canonical path cannot be determined for any reason.
     */
    String getCanonicalPath() throws IOException;

    /**
     * Tests if this abstract path is an absolute path.
     * <p>
     * For example, a UNIX-style path, {@code /home/fteuser/test} is an absolute
     * path, whereas {@code fteuser/test} is not.
     *
     * @return {@code true} if this abstract path is an absolute path, {@code
     *         false} otherwise.
     */
    boolean isAbsolute();

    /**
     * Tests if the resource denoted by this abstract path exists.
     *
     * @return {@code true} if the resource denoted by this abstract path
     *         exists, {@code false} otherwise.
     * @throws IOException
     *             If the existence of the resource cannot be determined for any
     *             reason.
     */
    boolean exists() throws IOException;

    /**
     * Tests whether the calling application can read the resource denoted by
     * this abstract path.
     *
     * @return {@code true} if the resource for this path exists and can be
     *         read, {@code false} otherwise.
     * @throws IOException
     *             If a problem occurs while attempting to determine if the
     *             resource can be read.
     */
    boolean canRead() throws IOException;

    /**
     * Tests whether the calling application can modify the resource denoted by
     * this abstract path.
     *
     * @return {@code true} if the resource for this path exists and can be
     *         modified, {@code false} otherwise.
     * @throws IOException
     *             If a problem occurs while attempting to determine if the
     *             resource can be modified.
     */
    boolean canWrite() throws IOException;

    /**
     * Tests whether the specified user is permitted to read the resource
     * denoted by this abstract path.
     * <p>
     * When WMQFTE invokes this method, the user identifier is the MQMD user
     * identifier for the requesting transfer.
     *
     * @param userId
     *            User identifier to test for access.
     * @return {@code true} if the resource for this abstract path exists and is
     *         permitted to be read by the specified user, {@code false}
```

```
     *         otherwise.
     * @throws IOException
     *             If a problem occurs while attempting to determine if the user
     *             is permitted to read the resource.
     */
    boolean readPermitted(String userId) throws IOException;

    /**
     * Tests whether the specified user is permitted to modify the resource
     * denoted by this abstract path.
     * <p>
     * When WMQFTE invokes this method, the user identifier is the MQMD user
     * identifier for the requesting transfer.
     *
     * @param userId
     *            User identifier to test for access.
     * @return {@code true} if the resource for this abstract path exists and is
     *         permitted to be modified by the specified user, {@code false}
     *         otherwise.
     * @throws IOException
     *             If a problem occurs while attempting to determine if the user
     *             is permitted to modify the resource.
     */
    boolean writePermitted(String userId) throws IOException;

    /**
     * Tests if the resource denoted by this abstract path is a directory-type
     * resource.
     *
     * @return {@code true} if the resource denoted by this abstract path is a
     *         directory type resource, {@code false} otherwise.
     */
    boolean isDirectory();

    /**
     * Creates the resource denoted by this abstract path, if it does not
     * already exist.
     *
     * @return {@code true} if the resource does not exist and was successfully
     *         created, {@code false} if the resource already existed.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while attempting to create
     *             the resource. This means that WMQFTE can attempt to recover
     *             the transfer.
     * @throws IOException
     *             If some other I/O problem occurs.
     */
    boolean createNewPath() throws RecoverableIOException, IOException;

    /**
     * Tests if the resource denoted by this abstract path is a file-type
     * resource.
     *
     * @return {@code true} if the resource denoted by this abstract path is a
     *         file type resource, {@code false} otherwise.
     */
    boolean isFile();

    /**
     * Obtains the last modified time for the resource denoted by this abstract
     * path.
     * <p>
     * This time is measured in milliseconds since the epoch (00:00:00 GMT,
     * January 1, 1970).
     *
     * @return The last modified time for the resource denoted by this abstract
     *         path, or a value of 0L if the resource does not exist or a
     *         problem occurs.
     */
    long lastModified();

    /**
     * Deletes the resource denoted by this abstract path.
     * <p>
     * If the resource is a directory, it must be empty for the delete to work.
     *
     * @throws IOException
     *             If the delete of the resource fails for any reason.
     */
    void delete() throws IOException;

    /**
```

```
     * Renames the resource denoted by this abstract path to the specified
     * destination abstract path.
     * <p>
     * The rename should still be successful if the resource for the specified
     * destination abstract path already exists and it is possible to replace
     * it.
     *
     * @param destination
     *            The new abstract path for the resource denoted by this
     *            abstract path.
     * @throws IOException
     *             If the rename of the resource fails for any reason.
     */
    void renameTo(IOExitResourcePath destination) throws IOException;

    /**
     * Creates a new path to use for writing to a temporary resource that did
     * not previously exist.
     * <p>
     * The implementation can choose the abstract path name for the temporary
     * resource. However, for clarity and problem diagnosis, the abstract path
     * name for the temporary resource should be based on this abstract path
     * name with the specified suffix appended and additional characters to make
     * the path unique (for example, sequence numbers), as required.
     * <p>
     * When WMQFTE transfers data to a destination it normally attempts to first
     * write to a temporary resource then on transfer completion renames the
     * temporary resource to the required destination. This method is called by
     * WMQFTE to create a new temporary resource path. The returned path should
     * be new and the resource should not previously exist.
     *
     * @param suffix
     *            Recommended suffix to use for the generated temporary path.
     *
     * @return A new {@link IOExitResourcePath} instance for the temporary
     *         resource path, that did not previously exist.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs whilst attempting to create
     *             the temporary resource. This means that WMQFTE can attempt to
     *             recover the transfer.
     * @throws IOException
     *             If some other I/O problem occurs.
     */
    IOExitResourcePath createTempPath(String suffix)
            throws RecoverableIOException, IOException;

    /**
     * Opens a {@link IOExitChannel} instance for reading data from the resource
     * denoted by this abstract path. The current data byte position for the
     * resource is expected to be the passed position value, such that when
     * {@link IOExitChannel#read(java.nio.ByteBuffer)} is called, data starting
     * from that position is read.
     *
     * @param position
     *            The required data byte read position.
     * @return A new {@link IOExitChannel} instance allowing data to be read
     *         from the resource denoted by this abstract path.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while attempting to open the
     *             resource for reading. This means that WMQFTE can attempt to
     *             recover the transfer.
     * @throws IOException
     *             If some other I/O problem occurs.
     */
    IOExitChannel openForRead(long position) throws RecoverableIOException,
            IOException;

    /**
     * Opens a {@link IOExitChannel} instance for writing data to the resource
     * denoted by this abstract path. Writing of data, using the
     * {@link IOExitChannel#write(java.nio.ByteBuffer)} method, starts at either
     * the beginning of the resource or end of the current data for the
     * resource, depending on the specified append parameter.
     *
     * @param append
     *            When {@code true} indicates that data written to the resource
     *            should be appended to the end of the current data. When
     *            {@code false} indicates that writing of data is to start at
     *            the beginning of the resource; any existing data is lost.
     * @return A new {@link IOExitChannel} instance allowing data to be written
     *         to the resource denoted by this abstract path.
     * @throws RecoverableIOException
```

```
     *             If a recoverable problem occurs whilst attempting to open the
     *             resource for writing. This means that WMQFTE can attempt to
     *             recover the transfer.
     * @throws IOException
     *             If some other I/O problem occurs.
     */
    IOExitChannel openForWrite(boolean append) throws RecoverableIOException,
            IOException;

    /**
     * Tests if the resource denoted by this abstract path is in use by another
     * application. Typically, this is because another application has a lock on
     * the resource either for shared or exclusive access.
     *
     * @return {code true} if resource denoted by this abstract path is in use
     *         by another application, {@code false} otherwise.
     */
    boolean inUse();

    /**
     * Obtains a {@link IOExitProperties} instance for properties associated
     * with the resource denoted by this abstract path.
     * <p>
     * WMQFTE will read these properties to govern how a transfer behaves when
     * interacting with the resource.
     *
     * @return A {@link IOExitProperties} instance for properties associated
     *         with the resource denoted by this abstract path.
     */
    IOExitProperties getProperties();

}
```

Attività correlate

Utilizzo delle uscite utente I/O di trasferimento MFT

Personalizzare MFT con uscite utente

Interfaccia IOExitWildcardPath.java

IOExitWildcardPath.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Represents a path that denotes a wildcard. This can be used to match multiple
 * resource paths.
 */
public interface IOExitWildcardPath extends IOExitPath {
```

Attività correlate

Utilizzo delle uscite utente I/O di trasferimento MFT

Personalizzare MFT con uscite utente

Interfaccia MonitorExit.java

MonitorExit.java

```
 /*
```

```
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   Copyright IBM Corp. 2009, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately prior to starting a task as the result of a monitor trigger
 */
public interface MonitorExit {

    /**
     * Invoked immediately prior to starting a task as the result of a monitor
     * trigger.
     *
     * @param environmentMetaData
     *            meta data about the environment in which the implementation
     *            of this method is running.  This information can only be read,
     *            it cannot be updated by the implementation.  The constant
     *            defined in <code>EnvironmentMetaDataConstants</code> class can
     *            be used to access the data held by this map.
     *
     * @param monitorMetaData
     *            meta data to associate with the monitor. The meta data passed
     *            to this method can be altered, and the changes will be
     *            reflected in subsequent exit routine invocations. This map
     *            also contains keys with IBM reserved names.  These entries are
     *            defined in the <code>MonitorMetaDataConstants</code> class and
     *            have special semantics. The the values of the IBM reserved names
     *            cannot be modified by the exit
     *
     * @param taskDetails
     *            An XML String representing the task to be executed as a result of
     *            the monitor triggering. This XML string may be modified by the
     *            exit
     *
     * @return    a monitor exit result object which is used to determine if the
     *            task should proceed, or be cancelled.
     */
    MonitorExitResult onMonitor(Map<String, String> environmentMetaData,
                                Map<String, String> monitorMetaData,
                                Reference<String> taskDetails);
}
```

Attività correlate

Monitoraggio delle risorse MFT

Personalizzare MFT con uscite utente

Riferimenti correlati

“SourceTransferStartExit.java interface” a pagina 2219

“Interfaccia SourceTransferEndExit.java” a pagina 2218

“Interfaccia DestinationTransferStartExit.java” a pagina 2194

“Interfaccia DestinationTransferEndExit.java” a pagina 2193

“Interfaccia ProtocolBridgeCredentialExit.java” a pagina 2213

Interfaccia ProtocolBridgeCredentialExit.java

ProtocolBridgeCredentialExit.java

```
/*
```

```
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   © Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing.  This interface defines methods that will
 * be invoked by a protocol bridge agent to map the MQ user ID of the transfer to credentials
 * that are to be used to access the protocol server.
 * There will be one instance of each implementation class per protocol bridge agent. The methods
 * can be called from different threads so the methods must be synchronized.
 */
public interface ProtocolBridgeCredentialExit {

    /**
     * Invoked once when a protocol bridge agent is started. It is intended to initialize
     * any resources that are required by the exit
     *
     * @param bridgeProperties
     *            The values of properties defined for the protocol bridge.
     *            These values can only be read, they cannot be updated by
     *            the implementation.
     *
     *
     * @return    true if the initialization is successful and false if unsuccessful
     *            If false is returned from an exit the protocol bridge agent will not
     *            start
     */

   public boolean initialize(final Map<String>  bridgeProperties);

      /**
      * Invoked once for each transfer to map the MQ user ID in the transfer message to the
      * credentials to be used to access the protocol server
       *
       * @param mqUserId The MQ user ID from which to map to the credentials to be used
      *                 access the protocol server
      * @return         A credential exit result object that contains the result of the map and
      *                 the credentials to use to access the protocol server
     */

   public CredentialExitResult mapMQUserId(final String mqUserId);

   /**
      * Invoked once when a protocol bridge agent is shutdown. It is intended to release
      * any resources that were allocated by the exit
      *
       * @param bridgeProperties
      *            The values of properties defined for the protocol bridge.
      *            These values can only be read, they cannot be updated by
       *            the implementation.
     *
       * @return
   */

   public void shutdown(final Map<String> bridgeProperties);

}
```

Attività correlate

Personalizzare MFT con uscite utente

Associazione delle credenziali per un server di file utilizzando le classi di uscita

Interfaccia ProtocolBridgeCredentialExit2.java

ProtocolBridgeCredentialExit2.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * An interface that is implemented by classes that are invoked as part of user
 * exit routine processing. This interface defines methods that are invoked by a
 * protocol bridge agent to map the MQ user ID of the transfer to credentials
 * used to access a specified protocol bridge server. There will be one instance
 * of each implementation class for each protocol bridge agent. The methods can
 * be called from different threads so the methods must be synchronized.
 */
public interface ProtocolBridgeCredentialExit2 extends
        ProtocolBridgeCredentialExit {

    /**
     * Invoked once for each transfer to map the MQ user ID in the transfer
     * message to the credentials used to access a specified protocol server.
     *
     * @param endPoint
     *            Information that describes the protocol server to be accessed.
     * @param mqUserId
     *            The MQ user ID from which to map the credentials used to
     *            access the protocol server.
     * @return A {@link CredentialExitResult) instance that contains the result
     *         of the map and the credentials to use to access the protocol
     *         server.
     */
    public CredentialExitResult mapMQUserId(
            final ProtocolServerEndPoint endPoint, final String mqUserId);
}
```

Attività correlate

Personalizzare MFT con uscite utente

Associazione delle credenziali per un server di file utilizzando le classi di uscita

Interfaccia ProtocolBridgePropertiesExit2.java

ProtocolBridgePropertiesExit2.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;
import java.util.Properties;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing. This interface defines methods that will be
 * invoked by a protocol bridge agent to look up properties for protocol servers
```

```
 * that are referenced in transfers.
 * <p>
 * There will be one instance of each implementation class for each protocol
 * bridge agent. The methods can be called from different threads so the methods
 * must be synchronised.
 */
public interface ProtocolBridgePropertiesExit2 {

    /**
     * Invoked once when a protocol bridge agent is started. It is intended to
     * initialize any resources that are required by the exit.
     *
     * @param bridgeProperties
     *            The values of properties defined for the protocol bridge.
     *            These values can only be read, they cannot be updated by the
     *            implementation.
     * @return {@code true} if the initialization is successful and {@code
     *         false} if unsuccessful. If {@code false} is returned from an exit
     *         the protocol bridge agent will not start.
     */
    public boolean initialize(final Map<String, String> bridgeProperties);

          /**
     * Invoked when the Protocol Bridge needs to access the protocol bridge credentials XML file.
     *
     * @return a {@link String} object giving the location of the ProtocolBridgeCredentials.xml
     */
    public String getCredentialLocation ();

    /**
     * Obtains a set of properties for the specified protocol server name.
     * <p>
     * The returned {@link Properties} must contain entries with key names
     * corresponding to the constants defined in
     * {@link ProtocolServerPropertyConstants} and in particular must include an
     * entry for all appropriate constants described as required.
     *
     * @param protocolServerName
     *            The name of the protocol server whose properties are to be
     *            returned. If a null or a blank value is specified, properties
     *            for the default protocol server are to be returned.
     * @return The {@link Properties} for the specified protocol server, or null
     *         if the server cannot be found.
     */
    public Properties getProtocolServerProperties(
            final String protocolServerName);

    /**
     * Invoked once when a protocol bridge agent is shut down. It is intended to
     * release any resources that were allocated by the exit.
     *
     * @param bridgeProperties
     *            The values of properties defined for the protocol bridge.
     *            These values can only be read, they cannot be updated by the
     *            implementation.
     */
    public void shutdown(final Map<String, String> bridgeProperties);

}
```

Attività correlate

ProtocolBridgePropertiesExit: ricerca delle propriet ... del server di file di protocollo

Personalizzare MFT con uscite utente

Associazione delle credenziali per un server di file utilizzando le classi di uscita

Classe SourceFileExitFileSpecification.java

SourceFileExitFileSpecification.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
```

```
 *   Copyright IBM Corp. 2012, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * A specification of the file names to use for a file transfer, as evaluated by the
 * agent acting as the source of the transfer.
 */
public final class SourceFileExitFileSpecification {

    private final String sourceFileSpecification;
    private final String destinationFileSpecification;
    private final Map<String, String> sourceFileMetaData;
    private final Map<String, String> destinationFileMetaData;

    /**
     * Constructor.  Creates a source file exit file specification.
     *
     * @param sourceFileSpecification
     *            the source file specification to associate with the source file
     *            exit file specification.
     *
     * @param destinationFileSpecification
     *            the destination file specification to associate with the
     *            source file exit file specification.
     *
     * @param sourceFileMetaData
     *            the source file meta data.
     *
     * @param destinationFileMetaData
     *            the destination file meta data   .
     */
    public SourceFileExitFileSpecification(final String sourceFileSpecification,
                                           final String destinationFileSpecification,
                                           final Map<String, String> sourceFileMetaData,
                                           final Map<String, String> destinationFileMetaData) {
        this.sourceFileSpecification = sourceFileSpecification;
        this.destinationFileSpecification = destinationFileSpecification;
        this.sourceFileMetaData = sourceFileMetaData;
        this.destinationFileMetaData = destinationFileMetaData;
    }

    /**
     * Returns the destination file specification.
     *
     * @return    the destination file specification.  This represents the location,
     *            on the agent acting as the destination for the transfer, where the
     *            file should be written.  Exit routines installed into the agent
     *            acting as the destination for the transfer may override this value.
     */
    public String getDestination() {
        return destinationFileSpecification;
    }

    /**
     * Returns the source file specification.
     *
     * @return    the source file specification.  This represents the location where
     *            the file data will be read from.
     */
    public String getSource() {
        return sourceFileSpecification;
    }

    /**
     * Returns the file meta data that relates to the source file specification.
     *
     * @return the file meta data that relates to the source file specification.
     */
    public Map<String, String> getSourceFileMetaData() {
        return sourceFileMetaData;
    }

    /**
     * Returns the file meta data that relates to the destination file specification.
     *
```

```
     * @return the file meta data that relates to the destination file specification.
     */
    public Map<String, String> getDestinationFileMetaData() {
        return destinationFileMetaData;
    }
}
```

Concetti correlati

“Metadati per uscite utente MFT” a pagina 2180

Ci sono tre diversi tipi di metadati che possono essere forniti alle routine di uscita utente per Managed File Transfer: ambiente, trasferimento e metadati file. Questi metadati vengono presentati come associazioni di coppie chiave - valore Java .

Interfaccia SourceTransferEndExit.java

SourceTransferEndExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately after completing a transfer on the agent acting as the
 * source of the transfer.
 */
public interface SourceTransferEndExit {

    /**
     * Invoked immediately after the completion of a transfer on the agent acting as
     * the source of the transfer.
     *
     * @param transferExitResult
     *            a result object reflecting whether or not the transfer completed
     *            successfully.
     *
     * @param sourceAgentName
     *            the name of the agent acting as the source of the transfer.
     *            This is the name of the agent that the implementation of this
     *            method will be invoked from.
     *
     * @param destinationAgentName
     *            the name of the agent acting as the destination of the
     *            transfer.
     *
     * @param environmentMetaData
     *            meta data about the environment in which the implementation
     *            of this method is running.  This information can only be read,
     *            it cannot be updated by the implementation.  The constants
     *            defined in <code>EnvironmentMetaDataConstants</code> class can
     *            be used to access the data held by this map.
     *
     * @param transferMetaData
     *            meta data to associate with the transfer.  The information can
     *            only be read, it cannot be updated by the implementation.  This
     *            map may also contain keys with IBM reserved names.  These
     *            entries are defined in the <code>TransferMetaDataConstants</code>
     *            class and have special semantics.
     *
     * @param fileResults
     *            a list of file transfer result objects that describe the source
     *            file name, destination file name and result of each file transfer
     *            operation attempted.
     *
```

```
     * @return    an optional description to enter into the log message describing
     *            transfer completion.  A value of <code>null</code> can be used
     *            when no description is required.
     */
    String onSourceTransferEnd(TransferExitResult transferExitResult,
                   String sourceAgentName,
                   String destinationAgentName,
                   Map<String, String>environmentMetaData,
                   Map<String, String>transferMetaData,
                   List<FileTransferResult>fileResults);


}
```

Attività correlate

Personalizzare MFT con uscite utente

Riferimenti correlati

“SourceTransferStartExit.java interface” a pagina 2219

“Interfaccia DestinationTransferStartExit.java” a pagina 2194

“Interfaccia DestinationTransferEndExit.java” a pagina 2193

“Interfaccia MonitorExit.java” a pagina 2212

“Interfaccia ProtocolBridgeCredentialExit.java” a pagina 2213

SourceTransferStartExit.java interface

SourceTransferStartExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

import java.util.List;
import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately prior to starting a transfer on the agent acting as the
 * source of the transfer.
 */
public interface SourceTransferStartExit {

    /**
     * Invoked immediately prior to starting a transfer on the agent acting as
     * the source of the transfer.
     *
     * @param sourceAgentName
     *            the name of the agent acting as the source of the transfer.
     *            This is the name of the agent that the implementation of this
     *            method will be invoked from.
     *
     * @param destinationAgentName
     *            the name of the agent acting as the destination of the
     *            transfer.
     *
     * @param environmentMetaData
     *            meta data about the environment in which the implementation
     *            of this method is running.  This information can only be read,
     *            it cannot be updated by the implementation.  The constants
     *            defined in <code>EnvironmentMetaDataConstants</code> class can
     *            be used to access the data held by this map.
```

```
     *
     * @param transferMetaData
     *            meta data to associate with the transfer.  The meta data passed
     *            to this method can be altered, and the changes to will be
     *            reflected in subsequent exit routine invocations.  This map may
     *            also contain keys with IBM reserved names.  These entries are
     *            defined in the <code>TransferMetaDataConstants</code> class and
     *            have special semantics.
     *
     * @param fileSpecs
     *            a list of file specifications that govern the file data to
     *            transfer.  The implementation of this method can add entries,
     *            remove entries, or modify entries in this list and the changes
     *            will be reflected in the files transferred.
     *
     * @return    a transfer exit result object which is used to determine if the
     *            transfer should proceed, or be cancelled.
     */
    TransferExitResult onSourceTransferStart(String sourceAgentName,
                        String destinationAgentName,
                        Map<String, String> environmentMetaData,
                        Map<String, String>transferMetaData,
                        List<SourceFileExitFileSpecification>fileSpecs);
}
```

Attività correlate

Personalizzare MFT con uscite utente

Riferimenti correlati

“Classe SourceFileExitFileSpecification.java” a pagina 2216

“Interfaccia SourceTransferEndExit.java” a pagina 2218

“Interfaccia DestinationTransferStartExit.java” a pagina 2194

“Interfaccia DestinationTransferEndExit.java” a pagina 2193

“Interfaccia MonitorExit.java” a pagina 2212

“Interfaccia ProtocolBridgeCredentialExit.java” a pagina 2213

Interfaccia TransferExitResult.java

TransferExitResult.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */

package com.ibm.wmqfte.exitroutine.api;

/**
 * The result of invoking a transfer exit routine.  It is composed of a result
 * code, which determines if the transfer should proceed, and an optional explanatory
 * message.  The explanation, if present, is entered into the log message.
 */
public class TransferExitResult {

    private final TransferExitResultCode resultCode;
    private final String explanation;

    /**
     * For convenience, a static "proceed" result with no associated explanation
     * message.
     */
    public static final TransferExitResult PROCEED_RESULT =
        new TransferExitResult(TransferExitResultCode.PROCEED, null);
```

```
    /**
     * Constructor.  Creates a transfer exit result object with a specified result
     * code and explanation.
     *
     * @param resultCode
     *            The result code to associate with the exit result being created.
     *
     * @param explanation
     *            The explanation to associate with the exit result being created.
     *            A value of <code>null</code> can be specified to indicate no
     *            explanation.
     */
    public TransferExitResult(TransferExitResultCode resultCode, String explanation) {
        this.resultCode = resultCode;
        this.explanation = explanation;
    }

    /**
     * Returns the explanation associated with this transfer exit result.
     *
     * @return    the explanation associated with this exit result.
     */
    public String getExplanation() {
        return explanation;
    }

    /**
     * Returns the result code associated with this transfer exit result.
     *
     * @return    the result code associated with this exit result.
     */
    public TransferExitResultCode getResultCode() {
        return resultCode;
    }
}
```

Attività correlate

Personalizzare MFT con uscite utente

Riferimenti correlati

“SourceTransferStartExit.java interface” a pagina 2219

“Interfaccia DestinationTransferStartExit.java” a pagina 2194

“Interfaccia DestinationTransferEndExit.java” a pagina 2193

“Interfaccia MonitorExit.java” a pagina 2212

“Interfaccia ProtocolBridgeCredentialExit.java” a pagina 2213

I formati dei messaggi che è possibile inserire nella coda comandi dell'agente MFT

Questi schemi XML definiscono i formati per i messaggi che possono essere inseriti nella coda comandi dell'agent per richiedere che l'agent esegua un'azione. Il messaggio XML può essere inserito nella coda comandi dell'agente utilizzando i comandi della riga comandi o da un'applicazione.

- Formato del messaggio di richiesta di trasferimento file
- Formati dei messaggi di richiesta di monitoraggio MFT
- Formato del messaggio di richiesta dell'agent MFT
- Formato del messaggio di replica dell'agent MFT

Riferimento REST API di messaggistica

Informazioni di riferimento su messaging REST API.

Per ulteriori informazioni sull'utilizzo di messaging REST API, consultare Messaggistica utilizzando REST API.

REST API risorse

Questa raccolta di argomenti fornisce informazioni di riferimento per ognuna delle risorse messaging REST API .

Per ulteriori informazioni sull'utilizzo di messaging REST API, consultare Messaggistica utilizzando REST API.

/messaging/qmgr/{qmgrName}/queue/{queueName}/message

L'API REST di messaggistica consente ai messaggi di essere inseriti in una coda o ai messaggi di essere esplorati o ricevuti in modo distruttivo da una coda, utilizzando la risorsa `/messaging/qmgr/{qmgrName}/queue/{queueName}/message` .

PUBBLICA /messaging/qmgr/{qmgrName}/queue/{queueName}/message

È possibile utilizzare il metodo HTTP POST con la risorsa `/messaging/qmgr/`*`{qmgrName}`*`/queue/`*`{queueName}`*`/message` per inserire i messaggi nella coda specificata sul gestore code specificato.

Inserisce un messaggio IBM MQ contenente il corpo della richiesta HTTP nel gestore code e nella coda specificati. Il gestore code deve essere sulla stessa macchina del server mqweb. Il metodo supporta solo il corpo della richiesta HTTP basato sul testo. I messaggi vengono inviati come messaggi formattati `MQSTR` o `JMS TextMessage` e vengono inseriti utilizzando il contesto utente corrente.

L'API REST V3 aggiunge la possibilità di specificare le proprietà del messaggio definite dall'utente e di includere la priorità del messaggio. Le intestazioni di richiesta ibm - mq - md - priority e ibm - mq - usr sono disponibili solo con l'API REST V3. L'intestazione della richiesta ibm - mq - md -correlationId ha un formato differente nell'API REST V3. L'intestazione può essere un ID specifico dell'applicazione oppure, se una stringa codificata richiede il prefisso `ID:` . Se la richiesta POST contiene messaggi definiti dall'utente o un ID di correlazione specifico dell'applicazione, il messaggio viene formattato come `JMS TextMessage`.

URL

`https://`*`host`*`:`*`port`*`/ibmmq/rest/v1/messaging/qmgr/`*`{qmgrName}`*`/queue/`*`{queueName}`*`/message`

`https://`*`host`*`:`*`port`*`/ibmmq/rest/v2/messaging/qmgr/`*`{qmgrName}`*`/queue/`*`{queueName}`*`/message`

`https://`*`host`*`:`*`port`*`/ibmmq/rest/v3/messaging/qmgr/`*`{qmgrName}`*`/queue/`*`{queueName}`*`/message`

qmgrName

Specifica il nome del gestore code a cui connettersi per la messaggistica. Il gestore code deve essere sulla stessa macchina del server mqweb.

V 9.4.0 Da IBM MQ 9.3.3, è possibile connettersi a un gestore code locale o remoto. Il nome specificato per **qmgrName** dipende dalla modalità di configurazione del server mqweb:

- Se il server mqweb è configurato per connettersi solo ai gestori code locali, utilizzare il nome del gestore code.
- Se il server mqweb è configurato per connettere i gestori code remoti, utilizzare il nome univoco per il gestore code come specificato nelle informazioni di connessione del gestore code remoto.

È possibile determinare se il server mqweb è configurato per la connessione ai gestori code locali o remoti utilizzando il comando **dspmqweb properties** e visualizzando la proprietà **mqRestMessagingConnectionMode** .

Il nome del gestore code è sensibile alle maiuscole/minuscole.

Se il nome del gestore code include una barra, un punto o un segno percentuale, a questi caratteri deve essere applicata la codifica URL.

- Una barra deve essere codificata come %2F.
- Un punto deve essere codificato come %2E.
- Un segno percentuale deve essere codificato come %25.

queueName
Specifica il nome della coda in cui inserire il messaggio.

La coda deve essere definita come locale, remota o come un alias per il gestore code specificato - può anche fare riferimento a una coda con cluster.

Il nome della coda è sensibile al maiuscolo / minuscolo.

Se il nome della coda include una barra o un segno di percentuale, questi caratteri devono essere codificati URL:

- Una barra, /, deve essere codificata come %2F.
- Un segno di percentuale,%, deve essere codificato come %25.

È possibile utilizzare HTTP invece di HTTPS se si abilitano le connessioni HTTP. Per ulteriori informazioni sull'abilitazione di HTTP, consultare Configurazione delle porte HTTP e HTTPS.

Intestazioni richiesta

Le seguenti intestazioni devono essere inviate con la richiesta:

Autorizzazione
Questa intestazione deve essere inviata se si sta utilizzando l'autenticazione di base. Per ulteriori informazioni, consultare la sezione relativa all'utilizzo dell'autenticazione di base HTTP con la REST API.

Content-Type
Questa intestazione deve essere inviata con uno dei seguenti valori:

- `text/plain;charset=utf-8`
- `text/html;charset=utf-8`
- `text/xml;charset=utf-8`
- `application/json;charset=utf-8`
- `application/xml;charset=utf-8`

ibm-mq-rest-csrf-token
Questa intestazione deve essere impostata ma il valore può essere qualsiasi cosa e può anche essere vuoto.

Le seguenti intestazioni possono essere facoltativamente inviate con la richiesta:

Accept-Language
Questa intestazione specifica la lingua richiesta per eventuali eccezioni o messaggi di errore restituiti nel corpo del messaggio di risposta.

REST API V2 REST API V1 **ibm - mq - md -correlationId**

Questa intestazione imposta l'ID di correlazione del messaggio creato. L'intestazione viene specificata come stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte. Non anteporre al valore `"ID:"`, l'API REST aggiunge automaticamente tale stringa.

Ad esempio: `ibm-mq-md-correlationId: 414d5120514d4144455620202020202067d8bf5923582e02`

REST API V3 **ibm - mq - md -correlationId**

Questa intestazione imposta l'ID di correlazione del messaggio creato. L'ID correlazione può assumere una delle seguenti forme:

- Una stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte, con prefisso `"ID:"`. Ad esempio: `ibm-mq-md-correlationId: ID:414d5120514d4144455620202020202067d8bf5923582e02`
- Un valore specifico dell'applicazione. Il valore è una stringa specifica dell'applicazione: `ibm-mq-md-correlationId: My-Custom-CorrelId`

  Se si specifica questo formato di ID correlazione, la destinazione del messaggio è WMQ_CLIENT_JMS_COMPLIANT e quindi incorpora un'intestazione MQRFH2 .

ibm - mq - md - scadenza

Questa intestazione imposta la scadenza del messaggio creato. La scadenza di un messaggio inizia dal momento in cui il messaggio arriva sulla coda. Di conseguenza, la latenza di rete viene ignorata. L'intestazione deve essere specificata come uno dei seguenti valori:

illimitato
Il messaggio non scade.

Questo è il valore predefinito.

Valore intero
Millisecondi prima della scadenza del messaggio.

Limitato all'intervallo 0 - 99999999900.

persistenza mq - mq - ibm

Questa intestazione imposta la persistenza per il messaggio creato. L'intestazione è specificata come uno dei seguenti valori:

nonPersistent
Il messaggio non sopravvive agli errori di sistema o al riavvio del gestore code.

Questo è il valore predefinito.

permanente
Il messaggio sopravvive agli errori di sistema o al riavvio del gestore code.

REST API V3 **ibm - mq-md - priorità**

Per l'API REST V3, questa intestazione imposta la priorità del messaggio creato. L'intestazione è specificata come uno dei seguenti valori:

asDestination
Il messaggio utilizza la priorità specificata nell'attributo DEFPRTY dell'oggetto coda IBM MQ sottostante.

Questo è il valore predefinito.

Valore intero
Specificare la priorità effettiva come un numero intero compreso nell'intervallo 0-9.

Ad esempio: `ibm-mq-md-priority: asDestination`

Per l'API REST V1 e REST V2, la priorità del messaggio per POST è sempre 4.

ibm - mq - md -replyTo

Questa intestazione imposta la destinazione di risposta per il messaggio creato. Il formato dell'intestazione utilizza la notazione standard di fornitura della coda di risposta e di un gestore code facoltativo: `replyQueue[@replyQmgr]`

Ad esempio: `ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr`

REST API V3 **ibm - mq - usr**

Questa intestazione imposta le proprietà definite dall'utente del messaggio di richiesta.

Le proprietà hanno la seguente sintassi:

`ibm-mq-usr: property_name; user_value; user_type`

property_name
Il nome della proprietà utente specificata. Deve essere un nome proprietà JMS valido.

valore_utente
Il valore della proprietà.

tipo_utente
Il tipo di proprietà:

- `boolean` (true/false, MQBOOL)
- `byte` (numero intero a 8 bit, MQINT8)
- `short` (numero intero a 16 bit, MQINT16)
- `integer` (numero intero a 32 bit, MQINT32)
- `long` (numero intero a 64 bit, MQINT64)
- `float` (reale a 32 bit, MQFLOAT32)
- `double` (reale a 64 bit, MQFLOAT64)
- `string` (stringa tra virgolette)

È possibile impostare più proprietà su un messaggio. È possibile specificare più proprietà separate da virgole in una singola intestazione di richiesta ibm - mq - usr oppure è possibile utilizzare due o più istanze separate dell'intestazione della richiesta ibm - mq - usr.

Ad esempio, è possibile impostare più proprietà definite dall'utente in una singola intestazione:
`ibm-mq-usr: userPropertyA;5;byte,userPropertyB;-10;integer`

Oppure è possibile impostare più proprietà definite dall'utente in
istanze separate dell'intestazione:`ibm-mq-usr: userPropertyA;5;byte ibm-mq-usr: userPropertyB;-10;integer`

Formato corpo richiesta

Il corpo della richiesta deve essere testo e utilizzare la codifica UTF-8 . Non è richiesta alcuna struttura di testo specifica. Un messaggio formattato MQSTR contenente il testo del corpo della richiesta viene creato e inserito nella coda specificata.

REST API V3 Se vengono utilizzate le proprietà definite dall'utente dell'API REST V3 o le funzioni ID di correlazione specifiche dell'applicazione, viene creato un messaggio formattato `JMS TextMessage` contenente il testo del corpo della richiesta e inserito nella coda specificata.

Per ulteriori informazioni, vedi esempi.

Requisiti di sicurezza

Il chiamante deve essere autenticato sul server mqweb. I ruoli `MQWebAdmin` e `MQWebAdminRO` non sono applicabili per messaging REST API. Per ulteriori informazioni sulla sicurezza di REST API, consultare Sicurezza di IBM MQ Console e REST API.

Una volta autenticato sul server mqweb, l'utente è in grado di utilizzare sia messaging REST API che administrative REST API.

Al principal di sicurezza del chiamante deve essere concessa la possibilità di inserire i messaggi nella coda specificata:

- La coda specificata dalla parte *{queueName}* dell'URL di risorsa, deve essere abilitata a PUT.
- MQ Appliance ALW Per la coda specificata dalla parte *{queueName}* dell'URL della risorsa, l'autorizzazione +PUT deve essere concessa al principal di sicurezza del chiamante.
- z/OS Per la coda specificata dalla parte *{queueName}* dell'URL della risorsa, l'accesso UPDATE deve essere concesso al principal di sicurezza del chiamante.

Questo principal di sicurezza potrebbe essere l'utente che ha avviato il server mqweb o l'utente che ha eseguito l'accesso al server mqweb. Se il gestore code a cui ci si connette è un gestore code remoto, il principal di sicurezza potrebbe essere invece l'utente fornito dalle credenziali nelle informazioni di connessione del gestore code remoto o un altro utente determinato dalle regole di protezione del canale. Per ulteriori informazioni, consultare Determinazione del principal di sicurezza utilizzato da messaging REST API.

ALW Su AIX, Linux, and Windows, è possibile concedere l'autorizzazione ai principal di sicurezza per utilizzare le risorse IBM MQ servendosi del comando **setmqaut**. Per ulteriori informazioni, consultare la sezione relativa a **setmqaut** (concessione o revoca dell'autorizzazione).

Su z/OS, consultare la sezione relativa all'impostazione della sicurezza su z/OS.

Se si utilizza AMS (Advanced Message Security) con messaging REST API, tenere presente che tutti i messaggi vengono codificati utilizzando il contesto del server mqweb, non il contesto dell'utente che pubblica il messaggio.

Codici di stato della risposta

201
Messaggio creato e inviato correttamente.

400
Forniti dati non validi.

Ad esempio, è stato specificato un valore di intestazione richiesta non valido.

401
Non autenticato.

Il chiamante deve essere autenticato presso il server mqweb e deve essere membro di uno o più ruoli `MQWebAdmin`, `MQWebAdminRO` o `MQWebUser`. È necessario specificare anche l'intestazione `ibm-mq-rest-csrf-token` . Per ulteriori informazioni, fare riferimento a “Requisiti di sicurezza” a pagina 2225.

403
Non autorizzato.

Il chiamante viene autenticato sul server mqweb ed è associato ad un principal valido. Tuttavia, il principal non ha accesso a tutte o a un sottoinsieme delle risorse IBM MQ richieste oppure non è nel ruolo `MQWebUser` . Per ulteriori informazioni sull'accesso richiesto, consultare “Requisiti di sicurezza” a pagina 2225.

404
La coda non esiste.

405
La coda è inibita da PUT.

415
Un'intestazione o un corpo del messaggio è un tipo di supporto non supportato.

Ad esempio, l'intestazione `Content-Type` è impostata su un tipo di supporto non supportato.

500
Problema del server o codice di errore da IBM MQ.

502
Il principal di sicurezza corrente non può inviare il messaggio poiché il fornitore di messaggistica non supporta la funzione richiesta. Ad esempio, se il percorso di classe del server mqweb non è valido.

503
Il gestore code non è in esecuzione.

Intestazioni della risposta

Con la risposta vengono restituite le seguenti intestazioni:

Contenuto - Lingua
Specifica l'identificativo della lingua del messaggio di risposta in caso di errori o eccezioni. Utilizzato insieme all'intestazione della richiesta `Accept-Language` per indicare la lingua richiesta per eventuali condizioni di errore o di eccezione. Il valore predefinito del server mqweb viene utilizzato se la lingua richiesta non è supportata.

Content-Length
Specifica la lunghezza del corpo della risposta HTTP, anche quando non c'è contenuto. In caso di esito positivo il valore è zero.

Content-Type
Specifica il tipo di corpo della risposta. In caso di esito positivo il valore è `text/plain;charset=utf-8`. In caso di errori o eccezioni, il valore è `application/json;charset=utf-8`.

ibm - mq - md -messageId
Specifica l'ID messaggio assegnato da IBM MQ a questo messaggio. Come l'intestazione della richiesta `ibm-mq-md-correlationId` , è rappresentata come una stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte.

Ad esempio:

```
ibm-mq-md-messageId: 414d5120514d414445562020202020067d8ce5923582f07
```

ibm - mq - md -messageId
Specifica l'ID messaggio assegnato da IBM MQ a questo messaggio. Come l'intestazione della richiesta `ibm-mq-md-correlationId` , è rappresentata come stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte, preceduta dalla stringa `ID:`.

Ad esempio:

```
ibm-mq-md-messageId: ID:414d5120514d414445562020202020067d8ce5923582f07
```

ibm - mq - risolto - qmgr
Specifica il nome del gestore code utilizzato per la richiesta. Se è stato utilizzato un nome univoco nell'URL della risorsa, questa intestazione identifica il nome del gestore code associato a tale nome univoco. Se il nome univoco utilizzato nell'URL della risorsa fa riferimento a un gruppo di gestori code, questa intestazione identifica quale gestore code all'interno del gruppo è stato utilizzato.

Formato corpo della risposta

Il corpo della risposta è vuoto se il messaggio viene inviato correttamente. Se si verifica un errore, il corpo della risposta contiene un messaggio di errore. Per ulteriori informazioni, fare riferimento a REST API gestione degli errori.

Esempi

I seguenti esempi utilizzano l'URL risorsa v2. Se si sta utilizzando una versione di IBM MQ antecedente a IBM MQ 9.1.5, è necessario utilizzare invece l'URL di risorsa v1. Ossia, nell'URL di risorsa, sostituire v1 dove l'URL di esempio utilizza v2.

Il seguente esempio accede a un utente denominato `mquser` con la password `mquser`. In cURL, la richiesta di accesso potrebbe essere simile al seguente Windows esempio. Il token LTPA viene memorizzato nel file `cookiejar.txt` utilizzando l'indicatore `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Una volta eseguito l'accesso dell'utente, il token LTPA e l'intestazione HTTP `ibm-mq-rest-csrf-token` vengono utilizzati per autenticare ulteriori richieste. `ibm-mq-rest-csrf-token` `token_value` può essere qualsiasi valore, incluso uno spazio vuoto.

- Il seguente esempio di Windows cURL invia un messaggio alla coda `Q1` sul gestore code `QM1`, utilizzando opzioni predefinite. Il messaggio contiene il testo *"Hello World!"*:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
  -X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
  -H "Content-Type: text/plain;charset=utf-8" --data "Hello World!"
  ```

- Il seguente esempio Windows cURL invia un messaggio persistente alla coda `Q1` sul gestore code `QM1`, con una scadenza di 2 minuti. Il messaggio contiene il testo *"Hello World!"*:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
  -X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
  -H "Content-Type: text/plain;charset=utf-8" -H "ibm-mq-md-persistence: persistent"
  -H "ibm-mq-md-expiry: 120000" --data "Hello World!"
  ```

- Il seguente esempio di Windows cURL invia un messaggio non persistente alla coda `Q1` sul gestore code `QM1`, senza scadenza e ID di correlazione definito. Il messaggio contiene il testo *"Hello World!"*:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
  -X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Content-Type: text/plain;charset=utf-8" -H "ibm-mq-md-persistence: nonPersistent"
  -H "ibm-mq-md-expiry: unlimited" -H "ibm-mq-md-correlationId:
  414d5120514d414445562020202020202067d8b
  f5923582e02" --data "Hello World!"
  ```

GET /messaging/qmgr/{qmgrName}/queue/{queueName}/message

È possibile utilizzare il metodo HTTP GET con la risorsa `/messaging/qmgr/{qmgrName}/queue/{queueName}/message` per sfogliare i messaggi dal gestore code e dalla coda associati.

Esamina il primo messaggio disponibile dal gestore code e dalla coda specificati. Il gestore code deve essere sulla stessa macchina del server mqweb. Il corpo del messaggio viene restituito nel corpo della risposta HTTP. Il messaggio deve avere il formato `MQSTR` o `JMS TextMessage` e viene ricevuto utilizzando il contesto utente corrente.

Tutti i messaggi vengono lasciati sulla coda e viene restituito al chiamante un codice di stato appropriato per eventuali messaggi inappropriati. Ad esempio, un messaggio che non ha un formato `MQSTR` o `JMS TextMessage` .

L'API REST V3 aggiunge la possibilità di specificare le proprietà del messaggio definite dall'utente e di includere la priorità del messaggio con i messaggi. Le intestazioni di risposta ibm - mq - md - priority e ibm - mq - usr sono disponibili solo con l'API REST V3. L'intestazione della richiesta ibm - mq - md -correlationId ha un formato differente nell'API REST V3. L'intestazione può essere un ID specifico dell'applicazione o, se una stringa codificata, conserva il prefisso `ID:` . L'intestazione della risposta ibm - mq - md -messageId e il parametro della query hanno un formato differente nell'API REST V3, conserva il prefisso `ID:` .

- “URL” a pagina 2229
- “Parametri di query facoltativi” a pagina 2229
- “Intestazioni richiesta” a pagina 2230
- “Formato corpo richiesta” a pagina 2231
- “Requisiti di sicurezza” a pagina 2231

- “Codici di stato della risposta” a pagina 2231
- “Intestazioni della risposta” a pagina 2232
- “Formato corpo della risposta” a pagina 2234
- “Esempi” a pagina 2234

URL

`https://host:port/ibmmq/rest/v1/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

qmgrName
Specifica il nome del gestore code a cui connettersi per la messaggistica. Il gestore code deve essere sulla stessa macchina del server mqweb.

V 9.4.0 Da IBM MQ 9.3.3, è possibile connettersi a un gestore code locale o remoto. Il nome specificato per **qmgrName** dipende dalla modalità di configurazione del server mqweb:

- Se il server mqweb è configurato per connettersi solo ai gestori code locali, utilizzare il nome del gestore code.
- Se il server mqweb è configurato per connettere i gestori code remoti, utilizzare il nome univoco per il gestore code come specificato nelle informazioni di connessione del gestore code remoto.

È possibile determinare se il server mqweb è configurato per la connessione ai gestori code locali o remoti utilizzando il comando **dspmqweb properties** e visualizzando la proprietà **mqRestMessagingConnectionMode** .

Il nome del gestore code è sensibile alle maiuscole/minuscole.

Se il nome del gestore code include una barra, un punto o un segno percentuale, a questi caratteri deve essere applicata la codifica URL.

- Una barra (/) deve essere codificata come %2F.
- Un segno di percentuale (%) deve essere codificato come %25.

queueName
Specifica il nome della coda da cui esaminare il messaggio.

La coda deve essere definita come locale o come un alias che punta a una coda locale.

Il nome della coda è sensibile al maiuscolo / minuscolo.

Se il nome della coda include una barra o un segno di percentuale, questi caratteri devono essere codificati URL:

- Una barra, /, deve essere codificata come %2F.
- Un segno di percentuale,%, deve essere codificato come %25.

È possibile utilizzare HTTP invece di HTTPS se si abilitano le connessioni HTTP. Per ulteriori informazioni sull'abilitazione di HTTP, consultare Configurazione delle porte HTTP e HTTPS.

Parametri di query facoltativi

REST API V2 REST API V1 **correlationId=*hexValue***
Specifica che il metodo HTTP restituisce il messaggio successivo con l'ID di correlazione corrispondente.

hexValue
Il parametro query deve essere specificato come stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte.

Ad esempio:

```
../message?correlationId=414d5120514d4144455620202020202067d8bf5923582e02
```

REST API V3 **correlationId= ID:*hexValue* o correlationId=*application_specific_value***
Specifica che il metodo HTTP restituisce il messaggio successivo con l'ID di correlazione corrispondente.

hexValue
Il parametro query deve essere specificato come una stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte e preceduta dalla stringa `"ID:"`.

Ad esempio:

```
../message?correlationId=ID:414d5120514d4144455620202020202067d8bf5923582e02
```

valore_specifica_applicazione
Il parametro di query può essere specificato come stringa specifica dell'applicazione.

Ad esempio:

```
../message?correlationId=My-Custom-CorrelId
```

REST API V2 REST API V1 **messageId=*hexValue***
Specifica che il metodo HTTP restituisce il messaggio successivo con l'ID messaggio corrispondente.

hexValue
Il parametro query deve essere specificato come stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte.

Ad esempio:

```
../message?messageId=414d5120514d4144455620202020202067d8ce5923582f07
```

REST API V3 **messageId= ID:*hexValue***
Specifica che il metodo HTTP restituisce il messaggio successivo con l'ID messaggio corrispondente.

hexValue
Il parametro query deve essere specificato come una stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte e preceduta dalla stringa `"ID:"`.

Ad esempio:

```
../message?messageId=ID:414d5120514d4144455620202020202067d8ce5923582f07
```

Intestazioni richiesta

Le seguenti intestazioni devono essere inviate con la richiesta:

Autorizzazione
Questa intestazione deve essere inviata se si sta utilizzando l'autenticazione di base. Per ulteriori informazioni, consultare la sezione relativa all'utilizzo dell'autenticazione di base HTTP con la REST API.

ibm-mq-rest-csrf-token
Questa intestazione deve essere impostata ma il valore può essere qualsiasi cosa e può anche essere vuoto.

Le seguenti intestazioni possono essere facoltativamente inviate con la richiesta:

Accetta - Serie di caratteri
Questa intestazione può essere utilizzata per indicare la serie di caratteri accettabile per la risposta. Se specificato, questa intestazione deve essere impostata come `UTF-8`.

Accept-Language
Questa intestazione specifica la lingua richiesta per eventuali eccezioni o messaggi di errore restituiti nel corpo del messaggio di risposta.

Formato corpo richiesta

Nessuna.

Requisiti di sicurezza

Il chiamante deve essere autenticato sul server mqweb. I ruoli `MQWebAdmin` e `MQWebAdminRO` non sono applicabili per messaging REST API. Per ulteriori informazioni sulla sicurezza di REST API, consultare Sicurezza di IBM MQ Console e REST API.

Una volta autenticato sul server mqweb, l'utente è in grado di utilizzare sia messaging REST API che administrative REST API.

Al principal dl sicurezza del chiamante deve essere concessa la capacità di esplorare i messaggi dalla coda specificata;

- La coda specificata dalla parte *{queueName}* dell'URL di risorsa, deve essere abilitata a `BROWSE`.
- ALW MQ Appliance Per la coda specificata dalla parte *{queueName}* dell'URL di risorsa, devono essere concesse le autorizzazioni `+GET`, `+INQ` e `+BROWSE` al principal di sicurezza del chiamante.
- z/OS Per la coda specificata dalla parte *{queueName}* dell'URL di risorsa, deve essere concesso l'accesso `UPDATE` al principal di sicurezza del chiamante.

Questo principal di sicurezza potrebbe essere l'utente che ha avviato il server mqweb o l'utente che ha eseguito l'accesso al server mqweb. Se il gestore code a cui ci si connette è un gestore code remoto, il principal di sicurezza potrebbe essere invece l'utente fornito dalle credenziali nelle informazioni di connessione del gestore code remoto o un altro utente determinato dalle regole di protezione del canale. Per ulteriori informazioni, consultare Determinazione del principal di sicurezza utilizzato da messaging REST API.

ALW Su AIX, Linux, and Windows, è possibile concedere l'autorizzazione ai principal di sicurezza per utilizzare le risorse IBM MQ servendosi del comando **setmqaut**. Per ulteriori informazioni, consultare la sezione relativa a **setmqaut** (concessione o revoca dell'autorizzazione).

Su z/OS, consultare la sezione relativa all'impostazione della sicurezza su z/OS.

Codici di stato della risposta

200
Messaggio ricevuto correttamente.

204
Nessun messaggio disponibile.

400
Forniti dati non validi.

Ad esempio, è stato specificato un valore di parametro query non valido.

401
Non autenticato.

Il chiamante deve essere autenticato presso il server mqweb e deve essere membro di uno o più ruoli `MQWebAdmin`, `MQWebAdminRO` o `MQWebUser`. È necessario specificare anche l'intestazione `ibm-`

`mq-rest-csrf-token` . Per ulteriori informazioni, fare riferimento a “Requisiti di sicurezza” a pagina 2231.

403
Non autorizzato.

Il chiamante viene autenticato sul server mqweb ed è associato ad un principal valido. Tuttavia, il principal non ha accesso a tutte o a un sottoinsieme delle risorse IBM MQ richieste oppure non è nel ruolo `MQWebUser` . Per ulteriori informazioni sull'accesso richiesto, consultare “Requisiti di sicurezza” a pagina 2231.

404
La coda non esiste.

500
Problema del server o codice di errore da IBM MQ.

501
Non è possibile creare la risposta HTTP.

Ad esempio, il messaggio ricevuto ha un tipo non corretto o ha il tipo corretto, ma non è stato possibile elaborare il corpo.

502
Il principal di sicurezza corrente non può ricevere il messaggio poiché il provider di messaggistica non supporta la funzione richiesta. Ad esempio, se il percorso di classe del server mqweb non è valido.

503
Il gestore code non è in esecuzione.

Intestazioni della risposta

Con la risposta vengono restituite le seguenti intestazioni:

Contenuto - Lingua
Specifica l'identificativo della lingua del messaggio di risposta in caso di errori o eccezioni. Utilizzato insieme all'intestazione della richiesta `Accept-Language` per indicare la lingua richiesta per eventuali condizioni di errore o di eccezione. Il valore predefinito del server mqweb viene utilizzato se la lingua richiesta non è supportata.

Content-Length
Specifica la lunghezza del corpo della risposta HTTP, anche quando non c'è contenuto. Il valore contiene la lunghezza (byte) dei dati del messaggio.

Content-Type
Specifica il tipo di contenuto restituito nel corpo della risposta del messaggio ricevuto. In caso di esito positivo il valore è `text/plain;charset=utf-8`. In caso di errori o eccezioni, il valore è `application/json;charset=utf-8`.

REST API V2 REST API V1 **ibm - mq - md -correlationId**
Specifica l'ID di correlazione del messaggio ricevuto. L'intestazione viene restituita se il messaggio ricevuto contiene un ID di correlazione valido. Viene rappresentato come una stringa codificata esadecimale a 48 caratteri, che rappresenta 24 byte.

Ad esempio:

```
ibm-mq-md-correlationId: 414d5120514d4144455620202020202067d8bf5923582e02
```

REST API V3 **ibm - mq - md -correlationId**
Specifica l'ID di correlazione del messaggio ricevuto. L'intestazione viene restituita se il messaggio ricevuto contiene un ID di correlazione valido. L'ID correlazione può assumere una delle seguenti forme:

- Una stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte, con prefisso `"ID:"`. Ad esempio:

```
ibm-mq-md-correlationId: ID:414d5120514d414445562020202020202067d8bf5923582e02
```

- Un valore specifico dell'applicazione. Il valore è una stringa specifica dell'applicazione:

```
ibm-mq-md-correlationId: My-Custom-CorrelId
```

ibm - mq - md - scadenza
Specifica la durata di scadenza rimanente del messaggio ricevuto. L'intestazione può essere uno dei seguenti valori:

illimitato
Il messaggio non scade.

Valore intero
Millisecondi rimanenti prima della scadenza del messaggio.

REST API V2 REST API V1 **ibm - mq - md -messageId**
Specifica l'ID messaggio assegnato da IBM MQ a questo messaggio. Come l'intestazione `ibm-mq-md-correlationId` , è rappresentata come una stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte.

Ad esempio:

```
ibm-mq-md-messageId: 414d5120514d414445562020202020202067d8ce5923582f07
```

REST API V3 **ibm - mq - md -messageId**
Specifica l'ID messaggio assegnato da IBM MQ a questo messaggio. Come l'intestazione `ibm-mq-md-correlationId` , è rappresentata come una stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte, con prefisso `"ID:"`

Ad esempio:

```
ibm-mq-md-messageId: ID:414d5120514d414445562020202020202067d8ce5923582f07
```

persistenza mq - mq - ibm
Specifica la persistenza del messaggio ricevuto. L'intestazione può essere uno dei seguenti valori:

nonPersistent
Il messaggio non sopravvive agli errori di sistema o al riavvio del gestore code.

permanente
Il messaggio sopravvive agli errori di sistema o al riavvio del gestore code.

REST API V3 **ibm - mq-md - priorità**
Restituisce l'impostazione della priorità del messaggio. Ad esempio:

```
ibm-mq-md-priority: 3
```

ibm - mq - md -replyTo
Specifica la destinazione di risposta per il messaggio ricevuto. Il formato dell'intestazione utilizza la notazione standard della coda di risposta e del gestore code, `replyQueue@replyQmgr`.

Ad esempio:

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr
```

V 9.4.0 **ibm - mq - risolto - qmgr**

Specifica il nome del gestore code utilizzato per la richiesta. Se è stato utilizzato un nome univoco nell'URL della risorsa, questa intestazione identifica il nome del gestore code associato a tale nome univoco. Se il nome univoco utilizzato nell'URL della risorsa fa riferimento a un gruppo di gestori code, questa intestazione identifica quale gestore code all'interno del gruppo è stato utilizzato.

REST API V3 **ibm - mq - usr**

Restituisce le proprietà definite dall'utente del messaggio. È possibile impostare più proprietà su un messaggio, nel qual caso ci saranno due o più istanze separate dell'intestazione della risposta ibm - mq - usr.

Ad esempio:

```
ibm-mq-usr: myIProp;5;short
ibm-mq-usr: mySProp;"hi";string
ibm-mq-usr: myBProp;true;boolean
```

Le proprietà hanno la seguente sintassi:

```
ibm-mq-usr: property_name; user_value; user_type
```

property_name
Il nome della proprietà utente specificata. Deve essere un nome proprietà JMS valido.

valore_utente
Il valore della proprietà.

tipo_utente
Il tipo di proprietà:

- `boolean` (true/false, MQBOOL)
- `byte` (numero intero a 8 bit, MQINT8)
- `short` (numero intero a 16 bit, MQINT16)
- `integer` (numero intero a 32 bit, MQINT32)
- `long` (numero intero a 64 bit, MQINT64)
- `float` (reale a 32 bit, MQFLOAT32)
- `double` (reale a 64 bit, MQFLOAT64)
- `string` (stringa tra virgolette)

Formato corpo della risposta

In caso di esito positivo, il corpo della risposta contiene il corpo del messaggio ricevuto. Se si verifica un errore, il corpo della risposta contiene un messaggio di errore in formato JSON. Entrambe le risposte sono codificate `UTF-8` . Per ulteriori informazioni, fare riferimento a REST API gestione degli errori.

Tenere presente che quando si riceve un messaggio sono supportati solo i messaggi formattati IBM MQ `MQSTR` o `JMS TextMessage` .

L'esplorazione di una coda contrassegnata come GET inibita non restituisce alcun contenuto.

Se la coda che si sta ricercando contiene messaggi con identificativi di messaggi duplicati, il primo messaggio viene restituito quando si filtra in base all'identificativo del messaggio.

Esempi

I seguenti esempi utilizzano l'URL risorsa v2. Se si sta utilizzando una versione di IBM MQ antecedente a IBM MQ 9.1.5, è necessario utilizzare invece l'URL di risorsa v1. Ossia, nell'URL di risorsa, sostituire v1 dove l'URL di esempio utilizza v2.

Il seguente esempio accede a un utente denominato `mquser` con la password `mquser`. In cURL, la richiesta di accesso potrebbe essere simile al seguente Windows esempio. Il token LTPA viene memorizzato nel file `cookiejar.txt` utilizzando l'indicatore `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Una volta eseguito l'accesso dell'utente, il token LTPA e l'intestazione HTTP `ibm-mq-rest-csrf-token` vengono utilizzati per autenticare ulteriori richieste. `ibm-mq-rest-csrf-token` `token_value` può essere qualsiasi valore, incluso uno spazio vuoto.

- Il seguente esempio Windows cURL sfoglia il successivo messaggio disponibile dalla coda `Q1` nel gestore code `QM1`, utilizzando le opzioni predefinite:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
  -X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Accept: text/plain"
  ```

- REST API V2 REST API V1 Il seguente esempio Windows cURL sfoglia un messaggio con uno specifico ID di correlazione, `000000000000000000000000000000000000000000abcdabcd`, dalla coda `Q1` sul gestore code `QM1`:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message?
  correlationId=000000000000000000000000000000000000000000abcdabcd"
  -X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Accept: text/plain"
  ```

- REST API V3

  Il seguente esempio Windows cURL è uguale all'esempio precedente, ma utilizza l'API REST V3. L'esempio elenca solo i messaggi con l'ID correlazione corrispondente, `000000000000000000000000000000000000000000abcdabcd`, dalla coda `Q1` sul gestore code `QM1`:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v3/messaging/qmgr/QM1/queue/Q1/message?
  correlationId=ID:000000000000000000000000000000000000000000abcdabcd"
  -X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Accept: text/plain"
  ```

Elimina /messaging/qmgr/{qmgrName}/queue/{queueName}/message

È possibile utilizzare il metodo HTTP DELETE con la risorsa `/messaging/qmgr/{qmgrName}/queue/{queueName}/message` per ottenere in modo distruttivo i messaggi dal gestore code e dalla coda associati.

Richiama in modo distruttivo il successivo messaggio disponibile dal gestore code e dalla coda specificati, restituendo il corpo del messaggio nel corpo della risposta HTTP. Il gestore code deve essere sulla stessa macchina del server mqweb. Il messaggio deve avere il formato `MQSTR` o `JMS TextMessage`e viene ricevuto utilizzando il contesto utente corrente.

I messaggi incompatibili vengono lasciati nella coda e viene restituito un codice di stato appropriato al chiamante. Ad esempio, un messaggio che non ha un formato `MQSTR` o `JMS TextMessage` .

L'API REST V3 aggiunge la possibilità di specificare le proprietà del messaggio definite dall'utente e di includere la priorità del messaggio con i messaggi. Le intestazioni di risposta ibm - mq - md - priority e ibm - mq - usr sono disponibili solo con l'API REST V3. L'intestazione della richiesta ibm - mq - md -correlationId ha un formato differente nell'API REST V3. L'intestazione può essere un ID specifico dell'applicazione o, se una stringa codificata, conserva il prefisso `ID:` . L'intestazione della risposta ibm - mq - md -messageId e il parametro della query hanno un formato differente nell'API REST V3, conserva il prefisso `ID:` .

- “URL” a pagina 2236
- “Parametri di query facoltativi” a pagina 2236
- “Intestazioni richiesta” a pagina 2238

- “Formato corpo richiesta” a pagina 2238
- “Requisiti di sicurezza” a pagina 2238
- “Codici di stato della risposta” a pagina 2238
- “Intestazioni della risposta” a pagina 2239
- “Formato corpo della risposta” a pagina 2241
- “Esempi” a pagina 2242

URL

`https://host:port/ibmmq/rest/v1/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

qmgrName
Specifica il nome del gestore code a cui connettersi per la messaggistica. Il gestore code deve essere sulla stessa macchina del server mqweb.

V 9.4.0 Da IBM MQ 9.3.3, è possibile connettersi a un gestore code locale o remoto. Il nome specificato per **qmgrName** dipende dalla modalità di configurazione del server mqweb:

- Se il server mqweb è configurato per connettersi solo ai gestori code locali, utilizzare il nome del gestore code.
- Se il server mqweb è configurato per connettere i gestori code remoti, utilizzare il nome univoco per il gestore code come specificato nelle informazioni di connessione del gestore code remoto.

È possibile determinare se il server mqweb è configurato per la connessione ai gestori code locali o remoti utilizzando il comando **dspmqweb properties** e visualizzando la proprietà **mqRestMessagingConnectionMode** .

Il nome del gestore code è sensibile alle maiuscole/minuscole.

Se il nome del gestore code include una barra, un punto o un segno percentuale, a questi caratteri deve essere applicata la codifica URL.

- Una barra (/) deve essere codificata come %2F.
- Un segno di percentuale (%) deve essere codificato come %25.

queueName
Specifica il nome della coda da cui richiamare il messaggio successivo.

La coda deve essere definita come locale o come un alias che punta a una coda locale.

Il nome della coda è sensibile al maiuscolo / minuscolo.

Se il nome della coda include una barra o un segno di percentuale, questi caratteri devono essere codificati URL:

- Una barra, /, deve essere codificata come %2F.
- Un segno di percentuale,%, deve essere codificato come %25.

È possibile utilizzare HTTP invece di HTTPS se si abilitano le connessioni HTTP. Per ulteriori informazioni sull'abilitazione di HTTP, consultare Configurazione delle porte HTTP e HTTPS.

Parametri di query facoltativi

REST API V2 REST API V1 **correlationId=*hexValue***
Specifica che il metodo HTTP restituisce il messaggio successivo con l'ID di correlazione corrispondente.

hexValue
Il parametro query deve essere specificato come stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte.

Ad esempio:

```
../message?correlationId=414d5120514d4144455620202020202067d8bf5923582e02
```

REST API V3 **correlationId= ID:*hexValue* o correlationId=*application_specific_value***
Specifica che il metodo HTTP restituisce il messaggio successivo con l'ID di correlazione corrispondente.

hexValue
Il parametro query deve essere specificato come una stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte e preceduta dalla stringa `"ID:"`.

Ad esempio:

```
../message?correlationId=ID:414d5120514d4144455620202020202067d8bf5923582e02
```

valore_specifica_applicazione
Il parametro di query può essere specificato come stringa specifica dell'applicazione.

Ad esempio:

```
../message?correlationId=My-Custom-CorrelId
```

REST API V2 REST API V1 **messageId=*hexValue***
Specifica che il metodo HTTP restituisce il messaggio successivo con l'ID messaggio corrispondente.

hexValue
Il parametro query deve essere specificato come stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte.

Ad esempio:

```
../message?messageId=414d5120514d4144455620202020202067d8ce5923582f07
```

REST API V3 **messageId= ID:*hexValue***
Specifica che il metodo HTTP restituisce il messaggio successivo con l'ID messaggio corrispondente.

hexValue
Il parametro query deve essere specificato come una stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte e preceduta dalla stringa `"ID:"`.

Ad esempio:

```
../message?messageId=ID:414d5120514d4144455620202020202067d8ce5923582f07
```

wait=*integerValue*
Specifica che il metodo HTTP attenderà *integerValue* millisecondi per rendere disponibile il messaggio successivo.

integerValue
Il parametro di query deve essere specificato come un valore intero che rappresenta la durata in millisecondi. Il valore massimo è 2147483647.

Ad esempio:

```
../message?wait=120000
```

Intestazioni richiesta

Le seguenti intestazioni devono essere inviate con la richiesta:

Autorizzazione
Questa intestazione deve essere inviata se si sta utilizzando l'autenticazione di base. Per ulteriori informazioni, consultare la sezione relativa all'utilizzo dell'autenticazione di base HTTP con la REST API.

ibm-mq-rest-csrf-token
Questa intestazione deve essere impostata ma il valore può essere qualsiasi cosa e può anche essere vuoto.

Le seguenti intestazioni possono essere facoltativamente inviate con la richiesta:

Accetta - Serie di caratteri
Questa intestazione può essere utilizzata per indicare la serie di caratteri accettabile per la risposta. Se specificato, questa intestazione deve essere impostata come `UTF-8`.

Accept-Language
Questa intestazione specifica la lingua richiesta per eventuali eccezioni o messaggi di errore restituiti nel corpo del messaggio di risposta.

Formato corpo richiesta

Nessuna.

Requisiti di sicurezza

Il chiamante deve essere autenticato sul server mqweb. I ruoli `MQWebAdmin` e `MQWebAdminRO` non sono applicabili per messaging REST API. Per ulteriori informazioni sulla sicurezza di REST API, consultare Sicurezza di IBM MQ Console e REST API.

Una volta autenticato sul server mqweb, l'utente è in grado di utilizzare sia messaging REST API che administrative REST API.

Al principal di sicurezza del chiamante deve essere concessa la possibilità di richiamare i messaggi dalla coda specificata:

- La coda specificata dalla parte *{queueName}* dell'URL di risorsa, deve essere abilitata a `GET`.
- MQ Appliance ALW Per la coda specificata dalla parte *{queueName}* dell'URL di risorsa, devono essere concesse le autorizzazioni `+GET`, `+INQ` e `+BROWSE` al principal di sicurezza del chiamante.
- z/OS Per la coda specificata dalla parte *{queueName}* dell'URL di risorsa, deve essere concesso l'accesso `UPDATE` al principal di sicurezza del chiamante.

Questo principal di sicurezza potrebbe essere l'utente che ha avviato il server mqweb o l'utente che ha eseguito l'accesso al server mqweb. Se il gestore code a cui ci si connette è un gestore code remoto, il principal di sicurezza potrebbe essere invece l'utente fornito dalle credenziali nelle informazioni di connessione del gestore code remoto o un altro utente determinato dalle regole di protezione del canale. Per ulteriori informazioni, consultare Determinazione del principal di sicurezza utilizzato da messaging REST API.

ALW Su AIX, Linux, and Windows, è possibile concedere l'autorizzazione ai principal di sicurezza per utilizzare le risorse IBM MQ servendosi del comando **setmqaut**. Per ulteriori informazioni, consultare la sezione relativa a **setmqaut** (concessione o revoca dell'autorizzazione).

Su z/OS, consultare la sezione relativa all'impostazione della sicurezza su z/OS.

Codici di stato della risposta

200
Messaggio ricevuto correttamente.

204
Nessun messaggio disponibile.

400
Forniti dati non validi.

Ad esempio, è stato specificato un valore di parametro query non valido.

401
Non autenticato.

Il chiamante deve essere autenticato presso il server mqweb e deve essere membro di uno o più ruoli `MQWebAdmin`, `MQWebAdminRO` o `MQWebUser`. È necessario specificare anche l'intestazione `ibm-mq-rest-csrf-token` . Per ulteriori informazioni, fare riferimento a “Requisiti di sicurezza” a pagina 2238.

403
Non autorizzato.

Il chiamante viene autenticato sul server mqweb ed è associato ad un principal valido. Tuttavia, il principal non ha accesso a tutte o a un sottoinsieme delle risorse IBM MQ richieste oppure non è nel ruolo `MQWebUser` . Per ulteriori informazioni sull'accesso richiesto, consultare “Requisiti di sicurezza” a pagina 2238.

404
La coda non esiste.

405
La coda è GET inibita.

500
Problema del server o codice di errore da IBM MQ.

501
Non è possibile creare la risposta HTTP.

Ad esempio, il messaggio ricevuto ha un tipo non corretto o ha il tipo corretto, ma non è stato possibile elaborare il corpo.

502
Il principal di sicurezza corrente non può ricevere il messaggio poiché il provider di messaggistica non supporta la funzione richiesta. Ad esempio, se il percorso di classe del server mqweb non è valido.

503
Il gestore code non è in esecuzione.

Intestazioni della risposta

Con la risposta vengono restituite le seguenti intestazioni:

Contenuto - Lingua
Specifica l'identificativo della lingua del messaggio di risposta in caso di errori o eccezioni. Utilizzato insieme all'intestazione della richiesta `Accept-Language` per indicare la lingua richiesta per eventuali condizioni di errore o di eccezione. Il valore predefinito del server mqweb viene utilizzato se la lingua richiesta non è supportata.

Content-Length
Specifica la lunghezza del corpo della risposta HTTP, anche quando non c'è contenuto. Il valore contiene la lunghezza (byte) dei dati del messaggio.

Content-Type
Specifica il tipo di contenuto restituito nel corpo della risposta del messaggio ricevuto. In caso di esito positivo il valore è `text/plain;charset=utf-8`. In caso di errori o eccezioni, il valore è `application/json;charset=utf-8`.

REST API V2 REST API V1 **ibm - mq - md -correlationId**

Specifica l'ID di correlazione del messaggio ricevuto. L'intestazione viene restituita se il messaggio ricevuto contiene un ID di correlazione valido. Viene rappresentato come una stringa codificata esadecimale a 48 caratteri, che rappresenta 24 byte.

Ad esempio:

```
ibm-mq-md-correlationId: 414d5120514d4144455620202020202067d8bf5923582e02
```

REST API V3 **ibm - mq - md -correlationId**

Specifica l'ID di correlazione del messaggio ricevuto. L'intestazione viene restituita se il messaggio ricevuto contiene un ID di correlazione valido. L'ID correlazione può assumere una delle seguenti forme:

- Una stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte, con prefisso `"ID:"`. Ad esempio:

  ```
  ibm-mq-md-correlationId: ID:414d5120514d4144455620202020202067d8bf5923582e02
  ```

- Un valore specifico dell'applicazione. Il valore è una stringa specifica dell'applicazione:

  ```
  ibm-mq-md-correlationId: My-Custom-CorrelId
  ```

ibm - mq - md - scadenza

Specifica la durata di scadenza rimanente del messaggio ricevuto. L'intestazione può essere uno dei seguenti valori:

illimitato
: Il messaggio non scade.

Valore intero
: Millisecondi rimanenti prima della scadenza del messaggio.

REST API V2 REST API V1 **ibm - mq - md -messageId**

Specifica l'ID messaggio assegnato da IBM MQ a questo messaggio. Come l'intestazione `ibm-mq-md-correlationId` , è rappresentata come una stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte.

Ad esempio:

```
ibm-mq-md-messageId: 414d5120514d4144455620202020202067d8ce5923582f07
```

REST API V3 **ibm - mq - md -messageId**

Specifica l'ID messaggio assegnato da IBM MQ a questo messaggio. Come l'intestazione `ibm-mq-md-correlationId` , è rappresentata come una stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte, con prefisso `"ID:"`

Ad esempio:

```
ibm-mq-md-messageId: ID:414d5120514d4144455620202020202067d8ce5923582f07
```

persistenza mq - mq - ibm

Specifica la persistenza del messaggio ricevuto. L'intestazione può essere uno dei seguenti valori:

nonPersistent
: Il messaggio non sopravvive agli errori di sistema o al riavvio del gestore code.

permanente
: Il messaggio sopravvive agli errori di sistema o al riavvio del gestore code.

REST API V3 **ibm - mq-md - priorità**

Restituisce l'impostazione della priorità del messaggio. Ad esempio:

```
ibm-mq-md-priority: 3
```

ibm - mq - md -replyTo

Specifica la destinazione di risposta per il messaggio ricevuto. Il formato dell'intestazione utilizza la notazione standard della coda di risposta e del gestore code, `replyQueue@replyQmgr`.

Ad esempio:

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr
```

V 9.4.0 **ibm - mq - risolto - qmgr**

Specifica il nome del gestore code utilizzato per la richiesta. Se è stato utilizzato un nome univoco nell'URL della risorsa, questa intestazione identifica il nome del gestore code associato a tale nome univoco. Se il nome univoco utilizzato nell'URL della risorsa fa riferimento a un gruppo di gestori code, questa intestazione identifica quale gestore code all'interno del gruppo è stato utilizzato.

REST API V3 **ibm - mq - usr**

Restituisce le proprietà definite dall'utente del messaggio. È possibile impostare più proprietà su un messaggio, nel qual caso ci saranno due o più istanze separate dell'intestazione della risposta ibm - mq - usr.

Ad esempio:

```
ibm-mq-usr: myIProp;5;short
ibm-mq-usr: mySProp;"hi";string
ibm-mq-usr: myBProp;true;boolean
```

Le proprietà hanno la seguente sintassi:

```
ibm-mq-usr: property_name; user_value; user_type
```

property_name
Il nome della proprietà utente specificata. Deve essere un nome proprietà JMS valido.

valore_utente
Il valore della proprietà.

tipo_utente
Il tipo di proprietà:

- `boolean` (true/false, MQBOOL)
- `byte` (numero intero a 8 bit, MQINT8)
- `short` (numero intero a 16 bit, MQINT16)
- `integer` (numero intero a 32 bit, MQINT32)
- `long` (numero intero a 64 bit, MQINT64)
- `float` (reale a 32 bit, MQFLOAT32)
- `double` (reale a 64 bit, MQFLOAT64)
- `string` (stringa tra virgolette)

Formato corpo della risposta

In caso di esito positivo, il corpo della risposta contiene il corpo del messaggio ricevuto. Se si verifica un errore, il corpo della risposta contiene un messaggio di errore in formato JSON. Entrambe le risposte sono codificate UTF-8 . Per ulteriori informazioni, fare riferimento a REST API gestione degli errori.

Tenere presente che quando si riceve un messaggio sono supportati solo i messaggi formattati IBM MQ MQSTR e JMS `TextMessage` . Successivamente, tutti i messaggi vengono ricevuti nel punto di sincronizzazione e tutti i messaggi non gestiti vengono lasciati nella coda. La coda IBM MQ può essere configurata per spostare questi messaggi non elaborabili in una destinazione alternativa. Per ulteriori informazioni, consultare Gestione dei messaggi non elaborabili in IBM MQ classes per JMS.

Esempi

I seguenti esempi utilizzano l'URL risorsa v2. Se si sta utilizzando una versione di IBM MQ antecedente a IBM MQ 9.1.5, è necessario utilizzare invece l'URL di risorsa v1. Ossia, nell'URL di risorsa, sostituire `v1` dove l'URL di esempio utilizza `v2`.

Il seguente esempio accede a un utente denominato `mquser` con la password `mquser`. In cURL, la richiesta di accesso potrebbe essere simile al seguente Windows esempio. Il token LTPA viene memorizzato nel file `cookiejar.txt` utilizzando l'indicatore `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Una volta eseguito l'accesso dell'utente, il token LTPA e l'intestazione HTTP `ibm-mq-rest-csrf-token` vengono utilizzati per autenticare ulteriori richieste. `ibm-mq-rest-csrf-token` `token_value` può essere qualsiasi valore, incluso uno spazio vuoto.

- L'esempio seguente Windows cURL rimuove il successivo messaggio disponibile dalla coda `Q1` nel gestore code `QM1`, utilizzando le opzioni predefinite:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
  -X DELETE -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Accept: text/plain"
  ```

- Il seguente esempio di Windows cURL rimuove un messaggio con un ID correlazione specifico, `00000000000000000000000000000000000000000000abcdabcd`, dalla coda `Q1` sul gestore code `QM1`:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message?
  correlationId=00000000000000000000000000000000000000000000abcdabcd"
  -X DELETE -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Accept: text/plain"
  ```

- Il seguente esempio di Windows cURL rimuove un messaggio con un ID correlazione specifico, `00000000000000000000000000000000000000000000abcdabcd`, dalla coda `Q1` sul gestore code `QM1`, in attesa che il messaggio diventi disponibile per un massimo di 30 secondi. Se passano 30 secondi senza che il messaggio specificato venga inserito nella coda, la chiamata DELETE restituisce senza un messaggio:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message?
  correlationId=00000000000000000000000000000000000000000000abcdabcd&wait=30000"
  -X DELETE -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Accept: text/plain"
  ```

/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist

È possibile utilizzare il metodo HTTP GET con la risorsa `/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist` per ottenere un elenco di messaggi disponibili dalla coda specificata sul gestore code specificato.

GET /messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist

È possibile utilizzare il metodo HTTP GET con la risorsa `/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist` per ottenere un elenco di messaggi disponibili dalla coda specificata sul gestore code specificato.

Sfoglia un elenco di riepilogo dei messaggi dalla coda e dal gestore code specificati. Il gestore code deve essere sulla stessa macchina del server mqweb. I dati di riepilogo vengono restituiti nel corpo della

risposta HTTP come un array formattato JSON. I dati non contengono il payload dei messaggi e vengono ricevuti utilizzando il contesto utente corrente. Nessun messaggio viene rimosso dalla coda associata.

Se viene effettuata una richiesta per ottenere un elenco di messaggi disponibili da una coda che è GET inibita, viene restituito un array JSON vuoto.

URL

`https://host:port/ibmmq/rest/v1/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist`

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist`

`https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist`

qmgrName
Specifica il nome del gestore code a cui connettersi per la messaggistica. Il gestore code deve essere sulla stessa macchina del server mqweb.

V 9.4.0 Da IBM MQ 9.3.3, è possibile connettersi a un gestore code locale o remoto. Il nome specificato per **qmgrName** dipende dalla modalità di configurazione del server mqweb:

- Se il server mqweb è configurato per connettersi solo ai gestori code locali, utilizzare il nome del gestore code.
- Se il server mqweb è configurato per connettere i gestori code remoti, utilizzare il nome univoco per il gestore code come specificato nelle informazioni di connessione del gestore code remoto.

È possibile determinare se il server mqweb è configurato per la connessione ai gestori code locali o remoti utilizzando il comando **dspmqweb properties** e visualizzando la proprietà **mqRestMessagingConnectionMode** .

Il nome del gestore code è sensibile alle maiuscole/minuscole.

Se il nome del gestore code include una barra, un punto o un segno percentuale, a questi caratteri deve essere applicata la codifica URL.

- Una barra (/) deve essere codificata come %2F.
- Un segno di percentuale (%) deve essere codificato come %25.

queueName
Specifica il nome della coda da cui ricercare i messaggi.

La coda deve essere definita come locale o come un alias che punta a una coda locale.

Il nome della coda è sensibile al maiuscolo / minuscolo.

Se il nome della coda include una barra o un segno di percentuale, questi caratteri devono essere codificati URL:

- Una barra, /, deve essere codificata come %2F.

- Un segno di percentuale,%, deve essere codificato come %25.

È possibile utilizzare HTTP invece di HTTPS se si abilitano le connessioni HTTP. Per ulteriori informazioni sull'abilitazione di HTTP, consultare Configurazione delle porte HTTP e HTTPS.

Parametri di query facoltativi

REST API V2 **correlationId=*hexValue***

Specifica che il metodo HTTP restituisce il messaggio successivo con l'ID di correlazione corrispondente.

hexValue

Il parametro query deve essere specificato come stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte.

Ad esempio:

```
../messagelist?correlationId=414d5120514d4144455620202020202067d8bf5923582e02
```

REST API V3 **correlationId= ID:*hexValue* o correlationId=*application_specific_value***

Specifica che il metodo HTTP restituisce un elenco di messaggi con l'ID di correlazione corrispondente.

hexValue

Il parametro query deve essere specificato come una stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte e preceduta dalla stringa `"ID:"`.

Ad esempio:

```
../message?correlationId=ID:414d5120514d4144455620202020202067d8bf5923582e02
```

valore_specifica_applicazione

Il parametro di query può essere specificato come stringa specifica dell'applicazione.

Ad esempio:

```
../message?correlationId=My-Custom-CorrelId
```

REST API V2 REST API V1 **messageId=*hexValue***

Specifica che il metodo HTTP restituisce il messaggio successivo con l'ID messaggio corrispondente.

hexValue

Il parametro query deve essere specificato come stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte.

Ad esempio:

```
../message?messageId=414d5120514d4144455620202020202067d8ce5923582f07
```

REST API V3 **messageId= ID:*hexValue***

Specifica che il metodo HTTP restituisce il messaggio successivo con l'ID messaggio corrispondente.

hexValue

Il parametro query deve essere specificato come una stringa codificata esadecimale di 48 caratteri, che rappresenta 24 byte e preceduta dalla stringa `"ID:"`.

Ad esempio:

```
../message?messageId=ID:414d5120514d4144455620202020202067d8ce5923582f07
```

limit=*integerValue*
Specifica che il corpo della risposta del metodo HTTP è limitato agli elementi JSON *integerValue* .

integerValue
Il parametro di query deve essere specificato come un valore intero che rappresenta il numero massimo di elementi contenuti nel corpo della risposta JSON.

Il valore predefinito è 10 e il valore massimo è 2147483647.

Ad esempio:

```
../messagelist?limit=250
```

Intestazioni richiesta

Le seguenti intestazioni devono essere inviate con la richiesta:

Autorizzazione
Questa intestazione deve essere inviata se si sta utilizzando l'autenticazione di base. Per ulteriori informazioni, consultare la sezione relativa all'utilizzo dell'autenticazione di base HTTP con la REST API.

ibm-mq-rest-csrf-token
Questa intestazione deve essere impostata ma il valore può essere qualsiasi cosa e può anche essere vuoto.

Le seguenti intestazioni possono essere facoltativamente inviate con la richiesta:

Accetta - Serie di caratteri
Questa intestazione può essere utilizzata per indicare la serie di caratteri accettabile per la risposta. Se specificato, questa intestazione deve essere impostata come UTF-8.

Accept-Language
Questa intestazione specifica la lingua richiesta per eventuali eccezioni o messaggi di errore restituiti nel corpo del messaggio di risposta.

Formato corpo richiesta

Nessuna.

Requisiti di sicurezza

Il chiamante deve essere autenticato sul server mqweb. I ruoli `MQWebAdmin` e `MQWebAdminRO` non sono applicabili per messaging REST API. Per ulteriori informazioni sulla sicurezza di REST API, consultare Sicurezza di IBM MQ Console e REST API.

Una volta autenticato sul server mqweb, l'utente è in grado di utilizzare sia messaging REST API che administrative REST API.

Al principal dl sicurezza del chiamante deve essere concessa la capacità di esplorare i messaggi dalla coda specificata;

- La coda specificata dalla parte *{queueName}* dell'URL di risorsa, deve essere abilitata a `BROWSE`.
- ALW MQ Appliance Per la coda specificata dalla parte *{queueName}* dell'URL di risorsa, devono essere concesse le autorizzazioni `+GET`, `+INQ` e `+BROWSE` al principal di sicurezza del chiamante.
- z/OS Per la coda specificata dalla parte *{queueName}* dell'URL di risorsa, deve essere concesso l'accesso `UPDATE` al principal di sicurezza del chiamante.

Questo principal di sicurezza potrebbe essere l'utente che ha avviato il server mqweb o l'utente che ha eseguito l'accesso al server mqweb. Se il gestore code a cui ci si connette è un gestore code remoto, il principal di sicurezza potrebbe essere invece l'utente fornito dalle credenziali nelle informazioni di connessione del gestore code remoto o un altro utente determinato dalle regole di protezione del canale.

Per ulteriori informazioni, consultare Determinazione del principal di sicurezza utilizzato da messaging REST API.

ALW Su AIX, Linux, and Windows, è possibile concedere l'autorizzazione ai principal di sicurezza per utilizzare le risorse IBM MQ servendosi del comando **setmqaut**. Per ulteriori informazioni, consultare la sezione relativa a **setmqaut** (concessione o revoca dell'autorizzazione).

Su z/OS, consultare la sezione relativa all'impostazione della sicurezza su z/OS.

Codici di stato della risposta

200
Elenco messaggi ricevuto correttamente.

400
Forniti dati non validi.

Ad esempio, è stato specificato un valore di parametro query non valido.

401
Non autenticato.

Il chiamante deve essere autenticato presso il server mqweb e deve essere membro di uno o più ruoli `MQWebAdmin`, `MQWebAdminRO` o `MQWebUser`. È necessario specificare anche l'intestazione `ibm-mq-rest-csrf-token` . Per ulteriori informazioni, fare riferimento a “Requisiti di sicurezza” a pagina 2245.

403
Non autorizzato.

Il chiamante viene autenticato sul server mqweb ed è associato ad un principal valido. Tuttavia, il principal non ha accesso a tutte o a un sottoinsieme delle risorse IBM MQ richieste oppure non è nel ruolo `MQWebUser` . Per ulteriori informazioni sull'accesso richiesto, consultare “Requisiti di sicurezza” a pagina 2245.

404
La coda non esiste.

500
Problema del server o codice di errore da IBM MQ.

501
Non è possibile creare la risposta HTTP.

Ad esempio, il messaggio ricevuto ha un tipo non corretto o ha il tipo corretto, ma non è stato possibile elaborare il corpo.

502
Il principal di sicurezza corrente non può ricevere il messaggio poiché il provider di messaggistica non supporta la funzione richiesta. Ad esempio, se il percorso di classe del server mqweb non è valido.

503
Il gestore code non è in esecuzione.

Intestazioni della risposta

Contenuto - Lingua
Specifica l'identificativo della lingua del messaggio di risposta in caso di errori o eccezioni. Utilizzato insieme all'intestazione della richiesta `Accept-Language` per indicare la lingua richiesta per eventuali condizioni di errore o di eccezione. Il valore predefinito del server mqweb viene utilizzato se la lingua richiesta non è supportata.

Content-Length
Specifica la lunghezza del corpo della risposta HTTP, anche quando non c'è contenuto. Il valore contiene la lunghezza dei dati del messaggio, in byte.

Content-Type
Specifica il tipo di corpo della risposta. Il valore è `application/json;charset=utf-8`.

V 9.4.0 **ibm - mq - risolto - qmgr**

Specifica il nome del gestore code utilizzato per la richiesta. Se è stato utilizzato un nome univoco nell'URL della risorsa, questa intestazione identifica il nome del gestore code associato a tale nome univoco. Se il nome univoco utilizzato nell'URL della risorsa fa riferimento a un gruppo di gestori code, questa intestazione identifica quale gestore code all'interno del gruppo è stato utilizzato.

Formato corpo della risposta

In caso di esito positivo, il corpo della risposta è una risposta codificata UTF-8 . La risposta contiene un oggetto JSON esterno che contiene un singolo array JSON denominato `messages`. Ogni elemento nell'array è un oggetto JSON che contiene informazioni su un messaggio nella coda. Ciascun elemento contiene i seguenti attributi:

REST API V2 REST API V1 **correlationId**

Specifica l'ID correlazione del messaggio. Il valore viene restituito se il messaggio contiene un ID di correlazione valido.

REST API V3 **correlationId**

Specifica l'ID correlazione del messaggio. Il valore viene restituito se il messaggio contiene un ID di correlazione valido. L'ID di correlazione è preceduto dalla stringa "`ID:`" o può essere un valore specifico dell'applicazione.

REST API V2 REST API V1 **messageId**

Specifica l'ID messaggio assegnato da IBM MQ a questo messaggio. Viene rappresentato come una stringa codificata esadecimale a 48 caratteri, che rappresenta 24 byte.

REST API V3 **messageId**

Specifica l'ID messaggio assegnato da IBM MQ a questo messaggio. Viene rappresentato come una stringa codificata esadecimale a 48 caratteri, che rappresenta 24 byte. L'ID messaggio è preceduto dalla stringa "`ID:`".

formato

Specifica il campo del formato MQMD. In circostanze normali, i messaggi di testo conterranno il valore IBM MQ `MQSTR` .

Se viene effettuata una richiesta per ottenere un elenco di messaggi su una coda che è GET inibita, viene restituito un array JSON vuoto.

Se si verifica un errore, il corpo della risposta contiene un messaggio di errore in formato JSON. Per ulteriori informazioni, fare riferimento a REST API gestione degli errori.

Esempi

I seguenti esempi utilizzano l'URL risorsa v2. Se si sta utilizzando una versione di IBM MQ antecedente a IBM MQ 9.1.5, è necessario utilizzare invece l'URL di risorsa v1. Ossia, nell'URL di risorsa, sostituire `v1` dove l'URL di esempio utilizza `v2`.

Il seguente esempio accede a un utente denominato `mquser` con la password `mquser`. In cURL, la richiesta di accesso potrebbe essere simile al seguente Windows esempio. Il token LTPA viene memorizzato nel file `cookiejar.txt` utilizzando l'indicatore `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Una volta eseguito l'accesso dell'utente, il token LTPA e l'intestazione HTTP `ibm-mq-rest-csrf-token` vengono utilizzati per autenticare ulteriori richieste. `ibm-mq-rest-csrf-token` `token_value` può essere qualsiasi valore, incluso uno spazio vuoto.

- Il seguente esempio Windows cURL elenca i successivi dieci messaggi disponibili dalla coda Q1 nel gestore code QM1, utilizzando le opzioni predefinite:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/messagelist"
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
-H "Accept: application/json"
```

- Il seguente esempio di Windows cURL elenca i successivi duecento messaggi disponibili dalla coda Q1 sul gestore code QM1, utilizzando le opzioni predefinite:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/messagelist?
limit=200"
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
-H "Accept: application/json"
```

- REST API V2 REST API V1 Il seguente esempio di Windows cURL elenca solo i messaggi con l'ID correlazione corrispondente, 000000000000000000000000000000000000000000abcdabcd, dalla coda Q1 sul gestore code QM1:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/messagelist?
correlationId=000000000000000000000000000000000000000000abcdabcd"
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
-H "Accept: application/json"
```

- REST API V3

  Il seguente esempio Windows cURL è uguale all'esempio precedente, ma utilizza l'API REST V3. L'esempio elenca solo i messaggi con l'ID correlazione corrispondente, 000000000000000000000000000000000000000000abcdabcd, dalla coda Q1 sul gestore code QM1:

```
curl -k "https://localhost:9443/ibmmq/rest/v3/messaging/qmgr/QM1/queue/Q1/messagelist?
correlationId=ID:000000000000000000000000000000000000000000abcdabcd"
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
-H "Accept: application/json"
```

/messaging/qmgr/{qmgrName}/topic/{topicString}/message

È possibile utilizzare il metodo HTTP POST con la risorsa /messaging/qmgr/{qmgrName}/topic/{topicString}/message per pubblicare i messaggi nell'argomento specificato sul gestore code specificato.

PUBBLICA /messaging/qmgr/{qmgrName}/topic/{topicString}/message

È possibile utilizzare il metodo HTTP POST con la risorsa /messaging/qmgr/{qmgrName}/topic/{topicString}/message per pubblicare i messaggi nell'argomento specificato sul gestore code specificato.

Pubblica un messaggio di testo nel corpo della richiesta HTTP nel gestore code e nell'argomento specificati. Il gestore code deve trovarsi sulla stessa macchina del server mqweb e sono supportati solo i messaggi basati su testo. I messaggi vengono pubblicati come messaggi formattati MQSTR o JMS TextMessage utilizzando il contesto utente corrente e hanno una priorità di messaggio predefinita di 4 per l'API REST V2e asDestination per l'API REST V3.

L'API REST V3 aggiunge la possibilità di specificare le proprietà del messaggio definite dall'utente e di includere la priorità del messaggio. Le intestazioni di richiesta ibm - mq - md - priority e ibm - mq - usr sono disponibili solo con l'API REST V3. L'intestazione della richiesta ibm - mq - md -correlationId ha un formato differente nell'API REST V3. L'intestazione può essere un ID specifico dell'applicazione oppure, se una stringa codificata richiede il prefisso ID: . Se la richiesta POST contiene messaggi definiti dall'utente o un ID di correlazione specifico dell'applicazione, il messaggio viene formattato come JMS TextMessage.

- "URL" a pagina 2249
- "Intestazioni richiesta" a pagina 2250

URL

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/topic/{topicString}/message`

`https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/topic/{topicString}/message`

qmgrName
Specifica il nome di un gestore code a cui connettersi per la messaggistica.

V 9.4.0 Da IBM MQ 9.3.3, è possibile connettersi a un gestore code locale o remoto. Il nome specificato per **qmgrName** dipende dalla modalità di configurazione del server mqweb:

- Se il server mqweb è configurato per connettersi solo ai gestori code locali, utilizzare il nome del gestore code.
- Se il server mqweb è configurato per connettere i gestori code remoti, utilizzare il nome univoco per il gestore code come specificato nelle informazioni di connessione del gestore code remoto.

È possibile determinare se il server mqweb è configurato per la connessione ai gestori code locali o remoti utilizzando il comando **dspmqweb properties** e visualizzando la proprietà **mqRestMessagingConnectionMode** .

Questo valore è sensibile al maiuscolo/minuscolo.

Se il nome include una barra, un punto o un segno di percentuale, questi caratteri devono essere codificati URL:

- Una barra deve essere codificata come %2F.
- Un punto deve essere codificato come %2E.
- Un segno percentuale deve essere codificato come %25.

topicString
Specifica la stringa di argomenti su cui pubblicare il messaggio.

La stringa dell'argomento è sensibile al maiuscolo / minuscolo. La stringa di argomento può contenere più livelli di argomento, separati dal delimitatore barra.

Se la stringa di argomenti contiene un segno di percentuale, un punto o un punto interrogativo, questi caratteri devono essere codificati URL:

- Un segno percentuale deve essere codificato come %25.
- Un punto deve essere codificato come %2E.
- Un punto interrogativo deve essere codificato come %3F.

Se la stringa dell'argomento inizia o termina con una barra, deve essere codificata con un %2F.

Ad esempio, per pubblicare la stringa di argomenti:

- `sport/football` sul gestore code `MY.QMGR`, utilizzare il seguente URL:

```
https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/MY%2EQMGR/topic/sport/football/message
```

- `/sport/football` sul gestore code `MY.QMGR`, utilizzare il seguente URL:

```
https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/MY%2EQMGR/topic/%2Fsport/football/
message
```

È possibile utilizzare HTTP invece di HTTPS se si abilitano le connessioni HTTP. Per ulteriori informazioni sull'abilitazione di HTTP, consultare Configurazione delle porte HTTP e HTTPS.

Intestazioni richiesta

Le seguenti intestazioni devono essere inviate con la richiesta:

Autorizzazione
Questa intestazione deve essere inviata se si sta utilizzando l'autenticazione di base. Per ulteriori informazioni, consultare la sezione relativa all'utilizzo dell'autenticazione di base HTTP con la REST API.

Content-Type
Questa intestazione deve essere inviata con uno dei seguenti valori:

- `text/plain;charset=utf-8`
- `text/html;charset=utf-8`
- `text/xml;charset=utf-8`
- `application/json;charset=utf-8`
- `application/xml;charset=utf-8`

ibm-mq-rest-csrf-token
Questa intestazione deve essere impostata ma il valore può essere qualsiasi cosa e può anche essere vuoto.

Le seguenti intestazioni possono essere facoltativamente inviate con la richiesta:

Accept-Language
Questa intestazione specifica la lingua richiesta per eventuali eccezioni o messaggi di errore restituiti nel corpo del messaggio di risposta.

ibm - mq - md - scadenza
Questa intestazione imposta la scadenza del messaggio creato. La scadenza di un messaggio inizia dal momento in cui il messaggio arriva al gestore code. Di conseguenza, la latenza di rete viene ignorata. L'intestazione deve essere specificata come uno dei seguenti valori:

illimitato
Il messaggio non scade.

Questo è il valore predefinito.

Valore intero
Millisecondi prima della scadenza del messaggio.

Limitato all'intervallo 0 - 99999999900.

persistenza mq - mq - ibm
Questa intestazione imposta la persistenza per il messaggio creato. L'intestazione deve essere specificata come uno dei seguenti valori:

nonPersistent
Il messaggio non sopravvive agli errori di sistema o al riavvio del gestore code.

Questo è il valore predefinito.

permanente
Il messaggio sopravvive agli errori di sistema o al riavvio del gestore code.

REST API V3 **ibm - mq-md - priorità**
Questa intestazione imposta la priorità del messaggio creato. L'intestazione deve essere specificata come uno dei seguenti valori:

asDestination
Il messaggio utilizza la priorità specificata nell'attributo DEFPRTY dell'oggetto coda IBM MQ sottostante.

Valore intero
Specificare la priorità effettiva come un numero intero compreso nell'intervallo 0-9.

Ad esempio:

```
ibm-mq-md-priority: asDestination
```

ibm - mq - md -replyTo
Questa intestazione imposta la destinazione di risposta per il messaggio creato. Il formato dell'intestazione utilizza la notazione standard di fornitura della coda di risposta e di un gestore code facoltativo: `replyQueue[@replyQmgr]`

Ad esempio:

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr
```

REST API V3 **ibm - mq - usr**
Impostare le proprietà definite dall'utente del messaggio di richiesta. È possibile impostare più proprietà su un messaggio. È possibile specificare più proprietà separate da virgole in una singola intestazione della richiesta ibm - mq - usr oppure è possibile utilizzare due o più istanze separate dell'intestazione della richiesta ibm - mq - usr.

Ad esempio:

```
ibm-mq-usr: myIProp;5;short
ibm-mq-usr: mySProp;"hi";string
ibm-mq-usr: myBProp;true;boolean
ibm-mq-usr: myA;5;byte,myB;-10;integer
```

Le proprietà hanno la seguente sintassi:

```
ibm-mq-usr: property_name; user_value; user_type
```

property_name
Il nome della proprietà utente specificata. Deve essere un nome proprietà JMS valido.

valore_utente
Il valore della proprietà.

tipo_utente
Il tipo di proprietà:

- `boolean` (true/false, MQBOOL)
- `byte` (numero intero a 8 bit, MQINT8)
- `short` (numero intero a 16 bit, MQINT16)
- `integer` (numero intero a 32 bit, MQINT32)
- `long` (numero intero a 64 bit, MQINT64)
- `float` (reale a 32 bit, MQFLOAT32)
- `double` (reale a 64 bit, MQFLOAT64)
- `string` (stringa tra virgolette)

Formato corpo richiesta

Il corpo della richiesta deve essere testo e utilizzare la codifica UTF-8 . Non è richiesta alcuna struttura di testo specifica. Un messaggio formattato MQSTR contenente il testo del corpo della richiesta viene creato e pubblicato nell'argomento specificato.

REST API V3 Se vengono utilizzate le proprietà definite dall'utente dell'API REST V3 o le funzioni ID di correlazione specifiche dell'applicazione, viene creato un messaggio formattato `JMS TextMessage` contenente il testo del corpo della richiesta e inserito nella coda specificata.

Per ulteriori informazioni, vedi esempi.

Requisiti di sicurezza

Il chiamante deve essere autenticato sul server mqweb. I ruoli `MQWebAdmin` e `MQWebAdminRO` non sono applicabili per messaging REST API. Per ulteriori informazioni sulla sicurezza di REST API, consultare Sicurezza di IBM MQ Console e REST API.

Una volta autenticato sul server mqweb, l'utente è in grado di utilizzare sia messaging REST API che administrative REST API.

Al principal di sicurezza del chiamante deve essere concessa la possibilità di pubblicare i messaggi nell'argomento specificato:

- L'argomento specificato dalla parte *{topicString}* dell'URL della risorsa deve essere `PUBLISH` abilitato.
- MQ Appliance ALW Per l'argomento che viene specificato dalla sezione *{topicString}* dell'URL della risorsa, l'autorizzazione +PUB deve essere concessa al principal di protezione del chiamante.
- z/OS Per l'argomento specificato nella parte *{topicString}* dell'URL della risorsa, l'accesso `UPDATE` deve essere concesso al principal di sicurezza del chiamante.

Questo principal di sicurezza potrebbe essere l'utente che ha avviato il server mqweb o l'utente che ha eseguito l'accesso al server mqweb. Se il gestore code a cui ci si connette è un gestore code remoto, il principal di sicurezza potrebbe essere invece l'utente fornito dalle credenziali nelle informazioni di connessione del gestore code remoto o un altro utente determinato dalle regole di protezione del canale. Per ulteriori informazioni, consultare Determinazione del principal di sicurezza utilizzato da messaging REST API.

ALW Su AIX, Linux, and Windows, è possibile concedere l'autorizzazione ai principal di sicurezza per utilizzare le risorse IBM MQ servendosi del comando **setmqaut**. Per ulteriori informazioni, consultare la sezione relativa a **setmqaut** (concessione o revoca dell'autorizzazione).

Su z/OS, consultare la sezione relativa all'impostazione della sicurezza su z/OS.

Se si utilizza AMS (Advanced Message Security) con messaging REST API, tenere presente che tutti i messaggi vengono codificati utilizzando il contesto del server mqweb, non il contesto dell'utente che pubblica il messaggio.

Codici di stato della risposta

201
Messaggio creato e pubblicato correttamente.

400
Forniti dati non validi.

Ad esempio, è stato specificato un valore di intestazione richiesta non valido.

401
Non autenticato.

Il chiamante deve essere autenticato presso il server mqweb e deve essere membro di uno o più ruoli `MQWebAdmin`, `MQWebAdminRO` o `MQWebUser`. È necessario specificare anche l'intestazione `ibm-mq-rest-csrf-token` . Per ulteriori informazioni, fare riferimento a “Requisiti di sicurezza” a pagina 2252.

403
Non autorizzato.

Il chiamante viene autenticato sul server mqweb ed è associato ad un principal valido. Tuttavia, il principal non ha accesso a tutte o a un sottoinsieme delle risorse IBM MQ richieste oppure non è nel ruolo `MQWebUser` . Per ulteriori informazioni sull'accesso richiesto, consultare “Requisiti di sicurezza” a pagina 2252.

404
Il gestore code non esiste.

405
L'argomento è PUBLISH inibito.

415
Un'intestazione o un corpo del messaggio è un tipo di supporto non supportato.

Ad esempio, l'intestazione `Content-Type` è impostata su un tipo di supporto non supportato.

500
Problema del server o codice di errore da IBM MQ.

502
Il principal di sicurezza corrente non è in grado di pubblicare il messaggio poiché il provider dei messaggi non supporta la funzione richiesta. Ad esempio, se il percorso di classe del server mqweb non è valido.

503
Il gestore code non è in esecuzione.

Intestazioni della risposta

Con la risposta vengono restituite le seguenti intestazioni:

Contenuto - Lingua
Specifica l'identificativo della lingua del messaggio di risposta in caso di errori o eccezioni. Utilizzato insieme all'intestazione della richiesta `Accept-Language` per indicare la lingua richiesta per eventuali condizioni di errore o di eccezione. Il valore predefinito del server mqweb viene utilizzato se la lingua richiesta non è supportata.

Content-Length
Specifica la lunghezza del corpo della risposta HTTP, anche quando non c'è contenuto. In caso di esito positivo il valore è zero.

Content-Type
Specifica il tipo di corpo della risposta. In caso di esito positivo il valore è `text/plain;charset=utf-8`. In caso di errori o eccezioni, il valore è `application/json;charset=utf-8`.

V 9.4.0 **ibm - mq - risolto - qmgr**
Specifica il nome del gestore code utilizzato per la richiesta. Se è stato utilizzato un nome univoco nell'URL della risorsa, questa intestazione identifica il nome del gestore code associato a tale nome univoco. Se il nome univoco utilizzato nell'URL della risorsa fa riferimento a un gruppo di gestori code, questa intestazione identifica quale gestore code all'interno del gruppo è stato utilizzato.

Formato corpo della risposta

Il corpo della risposta è vuoto se il messaggio è stato pubblicato correttamente. Se si verifica un errore, il corpo della risposta contiene un messaggio di errore. Per ulteriori informazioni, fare riferimento a REST API gestione degli errori.

Esempi

Il seguente esempio accede a un utente denominato `mquser` con la password `mquser`. In cURL, la richiesta di accesso potrebbe essere simile al seguente Windows esempio. Il token LTPA viene memorizzato nel file `cookiejar.txt` utilizzando l'indicatore `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v1/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Una volta eseguito l'accesso dell'utente, il token LTPA e l'intestazione HTTP `ibm-mq-rest-csrf-token` vengono utilizzati per autenticare ulteriori richieste. `ibm-mq-rest-csrf-token` `token_value` può essere qualsiasi valore, incluso uno spazio vuoto.

- Il seguente esempio Windows cURL pubblica un messaggio nella stringa di argomento `myTopic` sul gestore code `QM1`, utilizzando le opzioni predefinite. Il messaggio contiene il testo *"Hello World!"*:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/topic/myTopic/message"
  -X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
  -H "Content-Type: text/plain;charset=utf-8" --data "Hello World!"
  ```

- Il seguente esempio Windows cURL pubblica un messaggio persistente nella stringa di argomenti `myTopic/thisTopic` sul gestore code `QM1`, con una scadenza di due minuti. Il messaggio contiene il testo *"Hello World!"*:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/topic/myTopic%2FthisTopic/
  message"
  -X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
  -H "Content-Type: text/plain;charset=utf-8" -H "ibm-mq-md-persistence: persistent"
  -H "ibm-mq-md-expiry: 120000" --data "Hello World!"
  ```

Informazioni particolari

Queste informazioni sono state sviluppate per prodotti e servizi offerti negli Stati Uniti.

IBM potrebbe non offrire i prodotti, i servizi o le funzioni descritti in questo documento in altri paesi. Consultare il rappresentante IBM locale per informazioni sui prodotti e sui servizi disponibili nel proprio paese. Ogni riferimento relativo a prodotti, programmi o servizi IBM non implica che solo quei prodotti, programmi o servizi IBM possano essere utilizzati. In sostituzione a quelli forniti da IBM possono essere usati prodotti, programmi o servizi funzionalmente equivalenti che non comportino la violazione dei diritti di proprietà intellettuale o di altri diritti dell'IBM. Tuttavia, è responsabilità dell'utente valutare e verificare il funzionamento di qualsiasi prodotto, programma o servizio non IBM.

IBM potrebbe disporre di applicazioni di brevetti o brevetti in corso relativi all'argomento descritto in questo documento. La fornitura di tale documento non concede alcuna licenza a tali brevetti. Chi desiderasse ricevere informazioni relative a licenze può rivolgersi per iscritto a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Per richieste di licenze relative ad informazioni double-byte (DBCS), contattare il Dipartimento di Proprietà Intellettuale IBM nel proprio paese o inviare richieste per iscritto a:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

Il seguente paragrafo non si applica al Regno Unito o a qualunque altro paese in cui tali dichiarazioni sono incompatibili con le norme locali: INTERNATIONAL BUSINESS MACHINES CORPORATION FORNISCE LA PRESENTE PUBBLICAZIONE "NELLO STATO IN CUI SI TROVA" SENZA GARANZIE DI ALCUN TIPO, ESPRESSE O IMPLICITE, IVI INCLUSE, A TITOLO DI ESEMPIO, GARANZIE IMPLICITE DI NON VIOLAZIONE, DI COMMERCIABILITÀ E DI IDONEITÀ PER UNO SCOPO PARTICOLARE. Alcuni stati non consentono la rinuncia a garanzie esplicite o implicite in determinate transazioni; quindi la presente dichiarazione potrebbe non essere applicabile.

Questa pubblicazione potrebbe contenere imprecisioni tecniche o errori tipografici. Le informazioni incluse in questo documento vengono modificate su base periodica; tali modifiche vengono incorporate nelle nuove edizioni della pubblicazione. IBM si riserva il diritto di apportare miglioramenti o modifiche al prodotto/i e/o al programma/i descritti nella pubblicazione in qualsiasi momento e senza preavviso.

Tutti i riferimenti a siti Web non dell'IBM contenuti in questo documento sono forniti solo per consultazione e non rappresenta in alcun modo un'approvazione di tali siti. I materiali reperibili in tali siti Web non fanno parte dei materiali relativi a questo prodotto IBM e l'utilizzo di tali siti è responsabilità dell'utente.

Tutti i commenti e i suggerimenti inviati potranno essere utilizzati liberamente da IBM e diventeranno esclusiva della stessa.

Coloro che detengono la licenza su questo programma e desiderano avere informazioni su di esso allo scopo di consentire (i) uno scambio di informazioni tra programmi indipendenti ed altri (compreso questo) e (ii) l'uso reciproco di tali informazioni, dovrebbero rivolgersi a:

IBM Corporation
Coordinatore interoperabilità software, Dipartimento 49XA
Autostrada 3605 52 N

Rochester, MN 55901
U.S.A.

Queste informazioni possono essere rese disponibili secondo condizioni contrattuali appropriate, compreso, in alcuni casi, il pagamento di un addebito.

Il programma su licenza descritto in queste informazioni e tutto il materiale su licenza disponibile per esso sono forniti da IBM in base ai termini dell' IBM Customer Agreement, IBM International Program License Agreement o qualsiasi altro accordo equivalente tra le parti.

Tutti i dati relativi alle prestazioni contenuti in questo documento sono stati determinati in un ambiente controllato. Pertanto, i risultati ottenuti in altri ambienti operativi possono variare in modo significativo. Alcune misurazioni potrebbero essere state fatte su sistemi a livello di sviluppo e non vi è alcuna garanzia che queste misurazioni saranno le stesse sui sistemi generalmente disponibili. Inoltre, alcune misurazioni potrebbero essere state stimate mediante estrapolazione. I risultati quindi possono variare. Gli utenti di questo documento dovrebbero verificare i dati applicabili per il loro ambiente specifico.

Le informazioni relative a prodotti non IBM provengono dai fornitori di tali prodotti, dagli annunci pubblicati o da altre fonti pubblicamente disponibili. IBM non ha verificato tali prodotti e, pertanto, non può garantirne l'accuratezza delle prestazioni. Eventuali commenti relativi alle prestazioni dei prodotti non IBM devono essere indirizzati ai fornitori di tali prodotti.

Tutte le dichiarazioni riguardanti la direzione o l'intento futuro di IBM sono soggette a modifica o ritiro senza preavviso e rappresentano solo scopi e obiettivi.

Questa pubblicazione contiene esempi di dati e prospetti utilizzati quotidianamente nelle operazioni aziendali, Per poterli illustrare nel modo più completo possibile, gli esempi riportano nomi di persone, società, marchi e prodotti. Tutti questi nomi sono fittizi e qualsiasi somiglianza con nomi ed indirizzi adoperati da imprese realmente esistenti sono una mera coincidenza.

LICENZA SUL COPYRIGHT:

Queste informazioni contengono programmi applicativi di esempio in lingua originale, che illustrano le tecniche di programmazione su diverse piattaforme operative. È possibile copiare, modificare e distribuire questi programmi di esempio sotto qualsiasi forma senza alcun pagamento alla IBM, allo scopo di sviluppare, utilizzare, commercializzare o distribuire i programmi applicativi in conformità alle API (application programming interface) a seconda della piattaforma operativa per cui i programmi di esempio sono stati scritti. Questi esempi non sono stati testati approfonditamente tenendo conto di tutte le condizioni possibili. IBM, quindi, non può garantire o sottintendere l'affidabilità, l'utilità o il funzionamento di questi programmi.

Se si sta visualizzando queste informazioni in formato elettronico, le fotografie e le illustrazioni a colori potrebbero non apparire.

Informazioni sull'interfaccia di programmazione

Le informazioni sull'interfaccia di programmazione, se fornite, consentono di creare software applicativo da utilizzare con questo programma.

Questo manuale contiene informazioni sulle interfacce di programmazione che consentono al cliente di scrivere programmi per ottenere i servizi di IBM MQ.

Queste informazioni, tuttavia, possono contenere diagnosi, modifica e regolazione delle informazioni. La diagnosi, la modifica e la regolazione delle informazioni vengono fornite per consentire il debug del software applicativo.

Importante: Non utilizzare queste informazioni di diagnosi, modifica e ottimizzazione come interfaccia di programmazione poiché sono soggette a modifica.

Marchi

IBM, il logo IBM , ibm.com, sono marchi di IBM Corporation, registrati in molte giurisdizioni nel mondo. Un elenco aggiornato dei marchi IBM è disponibile sul web in "Copyright and trademark

information"www.ibm.com/legal/copytrade.shtml. Altri nomi di prodotti e servizi potrebbero essere marchi di IBM o altre società.

Microsoft e Windows sono marchi di Microsoft Corporation negli Stati Uniti, in altri paesi o entrambi.

UNIX è un marchio registrato di The Open Group negli Stati Uniti e/o in altri paesi.

Linux è un marchi registrato di Linus Torvalds negli Stati Uniti e/o in altri paesi.

Questo prodotto include il software sviluppato da Eclipse Project (https://www.eclipse.org/).

Java e tutti i marchi e i logo Java sono marchi registrati di Oracle e/o di società affiliate.

Numero parte:

(1P) P/N: