7.5

IBM WebSphere MQ Developing Applications Reference

Nota

Antes de utilizar esta información y el producto al que se refiere, lea la información en “Avisos” en la página 1485.

Esta edición se aplica a la versión 7 release 5 de IBM® WebSphere MQ y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones.

Cuando envía información a IBM, otorga a IBM un derecho no exclusivo para utilizar o distribuir la información de la forma que considere adecuada, sin incurrir por ello en ninguna obligación con el remitente.

Contenido

Guía de consulta para el desarrollo de aplicaciones

Utilice la información de esta sección para ayudarle a desarrollar sus aplicaciones IBM WebSphere MQ :

Tareas relacionadas

Desarrollo de aplicaciones

Referencia de aplicaciones MQI

Utilice los enlaces que se proporcionan en esta sección para ayudarle a desarrollar las aplicaciones MQI:

- “Ejemplos de código” en la página 8
- “Constantes” en la página 50
- “Tipos de datos utilizados en la MQI” en la página 218
- “Llamadas de funciones” en la página 603
- “objeto, atributos de” en la página 780
- “Códigos de retorno” en la página 855
- “Reglas para validar las opciones de MQI” en la página 856
- “Codificaciones de máquina” en la página 881
- “Opciones de informe y distintivos de mensaje” en la página 884
- “salida de conversión de datos” en la página 888
- “Propiedades especificadas como elementos MQRFH2” en la página 912
- “Conversión de páginas de códigos” en la página 920

Conceptos relacionados

“Referencia de salidas de usuarios, salidas de API y servicios instalables” en la página 1011
Utilice los enlaces que se proporcionan en esta sección para ayudarle a desarrollar las salidas de usuario, las salidas de API y las aplicaciones de servicios instalables:

“Las clases IBM WebSphere MQ para bibliotecas Java” en la página 1427
La ubicación de las clases IBM WebSphere MQ para bibliotecas Java varía según la plataforma. Especifique esta ubicación cuando inicie una aplicación.

Tareas relacionadas

Desarrollo de aplicaciones

Referencia relacionada

“Referencia SOAP” en la página 954
Transporte de WebSphere MQ para información de referencia SOAP ordenada alfabéticamente.

“Material de referencia para el puente IBM WebSphere MQ para HTTP” en la página 1216
Temas de referencia para el puente IBM WebSphere MQ para HTTP, ordenados alfabéticamente

“Las clases e interfaces .NET deIBM WebSphere MQ” en la página 1252
Las clases e interfaces .NET de IBM WebSphere MQ se listan alfabéticamente. Se describen las propiedades, métodos y constructores.

“IBM WebSphere MQ clases C++” en la página 1315
Las clases C++ de IBM WebSphere MQ encapsulan la interfaz de cola de mensajes (MQI) de IBM WebSphere MQ . Hay un único archivo de cabecera C++, **imqi.hpp**, que cubre todas estas clases.

Información relacionada

../com.ibm.mq.javadoc.doc/WMQJMSClasses/index.html

Ejemplos de código

Utilice la información de esta sección para llevar a cabo las tareas que están relacionadas con las necesidades de su negocio.

Ejemplos de lenguaje C

Esta colección de temas se toma principalmente de las aplicaciones de ejemplo de WebSphere MQ for z/OS . Son aplicables a todas las plataformas, excepto cuando se indique lo contrario.

conectar con un gestor de colas

Este ejemplo muestra cómo utilizar la llamada MQCONN para conectar un programa a un gestor de colas en el proceso por lotes z/OS .

Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BCA1) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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;
      }
.
   }
```

desconectarse de un gestor de colas

Este ejemplo muestra cómo utilizar la llamada MQDISC para desconectar un programa de un gestor de colas en el proceso por lotes z/OS .

Las variables utilizadas en esta extracción de código son las que se han establecido en “conectar con un gestor de colas” en la página 8. Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BCA1) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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;
      }
⋮
```

Creación de una cola dinámica

Este ejemplo muestra cómo utilizar la llamada MQOPEN para crear una cola dinámica.

Este extracto se toma de la aplicación de ejemplo Gestor de correo (programa CSQ4TCD1) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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 de una cola existente

Este ejemplo muestra cómo utilizar la llamada MQOPEN para abrir una cola que ya se ha definido.

Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BCA1) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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 */
```

cerrar una cola

Este ejemplo muestra cómo utilizar la llamada MQCLOSE para cerrar una cola.

Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BCA1) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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;
      }
.
```

Colocación de un mensaje utilizando MQPUT

Este ejemplo muestra cómo utilizar la llamada MQPUT para colocar un mensaje en una cola.

Este extracto no se toma de las aplicaciones de ejemplo proporcionadas con WebSphere MQ. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo, consulte Programas de ejemplo (plataformas excepto 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 */
        }
}
```

Colocación de un mensaje utilizando MQPUT1

Este ejemplo muestra cómo utilizar la llamada MQPUT1 para abrir una cola, colocar un único mensaje en la cola y, a continuación, cerrar la cola.

Este extracto se toma de la aplicación de ejemplo Comprobación de crédito (programa CSQ4CCB5) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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;
   }
.
```

obtener un mensaje

Este ejemplo muestra cómo utilizar la llamada MQGET para eliminar un mensaje de una cola.

Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BCA1) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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 */
```

Obtención de un mensaje utilizando la opción de espera

Este ejemplo muestra cómo utilizar la opción de espera de la llamada MQGET.

Este código acepta mensajes truncados. Este extracto se toma de la aplicación de ejemplo Comprobación de crédito (programa CSQ4CCB5) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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();
         }
⋮
```

Obtención de un mensaje utilizando la señalización

La señalización sólo está disponible con WebSphere MQ para z/OS.

Este ejemplo muestra cómo utilizar la llamada MQGET para establecer una señal para que se le notifique cuando llegue un mensaje adecuado a una cola. Este extracto no se toma de las aplicaciones de ejemplo proporcionadas con WebSphere 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;
}
```

Consulta de los atributos de un objeto

Este ejemplo muestra cómo utilizar la llamada MQINQ para consultar los atributos de una cola.

Este extracto se toma de la aplicación de ejemplo Atributos de cola (programa CSQ4CCC1) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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 */
```

Establecimiento de los atributos de una cola

Este ejemplo muestra cómo utilizar la llamada MQSET para cambiar los atributos de una cola.

Este extracto se toma de la aplicación de ejemplo Atributos de cola (programa CSQ4CCC1) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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 */
```

Recuperación de información de estado con MQSTAT

Este ejemplo muestra cómo emitir un MQPUT asíncrono y recuperar la información de estado con MQSTAT.

Este extracto se toma de la aplicación de ejemplo Llamando a MQSTAT (programa amqsapt0) suministrados con sistemas WebSphere MQ para Windows . Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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);
}
```

Ejemplos de COBOL

Esta colección de temas se toma de las aplicaciones de ejemplo de WebSphere MQ for z/OS . Son aplicables a todas las plataformas, excepto cuando se indique lo contrario.

conectar con un gestor de colas

Este ejemplo muestra cómo utilizar la llamada MQCONN para conectar un programa a un gestor de colas en el proceso por lotes z/OS .

Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BVA1) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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.
⋮
```

desconectarse de un gestor de colas

Este ejemplo muestra cómo utilizar la llamada MQDISC para desconectar un programa de un gestor de colas en el proceso por lotes z/OS .

Las variables utilizadas en esta extracción de código son las que se han establecido en “conectar con un gestor de colas” en la página 24. Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BVA1) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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.
⋮
```

Creación de una cola dinámica

Este ejemplo muestra cómo utilizar la llamada MQOPEN para crear una cola dinámica.

Este extracto se toma de la aplicación de ejemplo Comprobación de crédito (programa CSQ4CVB1) que se proporciona con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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 de una cola existente

Este ejemplo muestra cómo utilizar la llamada MQOPEN para abrir una cola existente.

Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BVA1) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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
```

cerrar una cola

Este ejemplo muestra cómo utilizar la llamada MQCLOSE.

Las variables utilizadas en esta extracción de código son las que se han establecido en “conectar con un gestor de colas” en la página 24. Este extracto se toma de la aplicación de ejemplo Examinar (programa CSQ4BVA1) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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.
*
```

Colocación de un mensaje utilizando MQPUT

Este ejemplo muestra cómo utilizar la llamada MQPUT utilizando el contexto.

Este extracto se toma de la aplicación de ejemplo Comprobación de crédito (programa CSQ4CVB1) que se proporciona con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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.
```

Colocación de un mensaje utilizando MQPUT1

Este ejemplo muestra cómo utilizar la llamada MQPUT1 .

Este extracto se toma de la aplicación de ejemplo Comprobación de crédito (programa CSQ4CVB5) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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.
*
```

obtener un mensaje

Este ejemplo muestra cómo utilizar la llamada MQGET para eliminar un mensaje de una cola.

Este extracto se toma de la aplicación de ejemplo Comprobación de crédito (programa CSQ4CVB1) que se proporciona con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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.
```

Obtención de un mensaje utilizando la opción de espera

Este ejemplo muestra cómo utilizar la llamada MQGET con la opción de espera y aceptando mensajes truncados.

Este extracto se toma de la aplicación de ejemplo Comprobación de crédito (programa CSQ4CVB5) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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.
⋮
```

Obtención de un mensaje utilizando la señalización

Este ejemplo muestra cómo utilizar la llamada MQGET con señalización. Este extracto se toma de la aplicación de ejemplo Comprobación de crédito (programa CSQ4CVB2) proporcionada con WebSphere MQ para z/OS.

La señalización sólo está disponible con WebSphere MQ para 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
*
⋮
```

Consulta de los atributos de un objeto

Este ejemplo muestra cómo utilizar la llamada MQINQ para consultar los atributos de una cola.

Este extracto se toma de la aplicación de ejemplo Atributos de cola (programa CSQ4CVC1) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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.
⋮
```

Establecimiento de los atributos de una cola

Este ejemplo muestra cómo utilizar la llamada MQSET para cambiar los atributos de una cola.

Este extracto se toma de la aplicación de ejemplo Atributos de cola (programa CSQ4CVC1) proporcionada con WebSphere MQ para z/OS. Para ver los nombres y ubicaciones de las aplicaciones de ejemplo en otras plataformas, consulte Programas de ejemplo (plataformas excepto 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.
```

Ejemplos de lenguaje ensamblador de System/390

Esta colección de temas se toma principalmente de las aplicaciones de ejemplo de WebSphere MQ for z/OS .

conectar con un gestor de colas

Este ejemplo muestra cómo utilizar la llamada MQCONN para conectar un programa a un gestor de colas en el proceso por lotes z/OS .

Este extracto se toma del programa de ejemplo Browse (CSQ4BAA1) proporcionado con WebSphere MQ para z/OS.

```
⋮
WORKAREA DSECT
*
PARMLIST CALL ,(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
*
```

desconectarse de un gestor de colas

Este ejemplo muestra cómo utilizar la llamada MQDISC para desconectar un programa de un gestor de colas en el proceso por lotes z/OS .

Este extracto no se toma de las aplicaciones de ejemplo proporcionadas con WebSphere 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
```

Creación de una cola dinámica

Este ejemplo muestra cómo utilizar la llamada MQOPEN para crear una cola dinámica.

Este extracto no se toma de las aplicaciones de ejemplo proporcionadas con WebSphere 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
```

Apertura de una cola existente

Este ejemplo muestra cómo utilizar la llamada MQOPEN para abrir una cola que ya se ha definido.

Muestra cómo especificar dos opciones. Este extracto no se toma de las aplicaciones de ejemplo proporcionadas con WebSphere 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
```

cerrar una cola

Este ejemplo muestra cómo utilizar la llamada MQCLOSE para cerrar una cola.

Este extracto no se toma de las aplicaciones de ejemplo proporcionadas con WebSphere 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
```

Colocación de un mensaje utilizando MQPUT

Este ejemplo muestra cómo utilizar la llamada MQPUT para colocar un mensaje en una cola.

Este extracto no se toma de las aplicaciones de ejemplo proporcionadas con WebSphere 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
```

Colocación de un mensaje utilizando MQPUT1

Este ejemplo muestra cómo utilizar la llamada MQPUT1 para abrir una cola, colocar un único mensaje en la cola y, a continuación, cerrar la cola.

Este extracto no se toma de las aplicaciones de ejemplo proporcionadas con WebSphere 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
```

obtener un mensaje

Este ejemplo muestra cómo utilizar la llamada MQGET para eliminar un mensaje de una cola.

Este extracto no se toma de las aplicaciones de ejemplo proporcionadas con WebSphere 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
```

Obtención de un mensaje utilizando la opción de espera

Este ejemplo muestra cómo utilizar la opción de espera de la llamada MQGET.

Este código acepta mensajes truncados. Este extracto no se toma de las aplicaciones de ejemplo proporcionadas con WebSphere 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 ,(0,0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
⋮
         END
```

Obtención de un mensaje utilizando la señalización

Este ejemplo muestra cómo utilizar la llamada MQGET para establecer una señal para que se le notifique cuando llegue un mensaje adecuado a una cola.

Este extracto no se toma de las aplicaciones de ejemplo proporcionadas con WebSphere 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
```

Consulta y establecimiento de los atributos de una cola

Este ejemplo muestra cómo utilizar la llamada MQINQ para consultar los atributos de una cola y utilizar la llamada MQSET para cambiar los atributos de una cola.

Este extracto se toma de la aplicación de ejemplo Atributos de cola (programa CSQ4CAC1) proporcionada con WebSphere MQ para 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?
⋮
```

Constantes

Utilice la información de esta sección para llevar a cabo las tareas que están relacionadas con las necesidades de su negocio.

Archivos COPY, de cabecera, de inclusión y de módulo de IBM WebSphere MQ

Esta información es información de interfaz de programación de uso general.

Esta sección contiene información para ayudarle a utilizar MQI para diversos lenguajes de programación, como se indica a continuación.

Archivos de cabecera C

Los archivos de cabecera se proporcionan para ayudarle a escribir programas de aplicación C que utilizan la MQI. Estos archivos de cabecera se resumen en la tabla:

Tabla 1. Archivos de cabecera C: llamar a prototipos, tipos de datos, códigos de retorno, constantes y estructuras

Nombre de archivo	Descripción	IBM i	Sistemas UNIX y Linux®	Windows	z/OS
Llamar a prototipos, tipos de datos, códigos de retorno, constantes y estructuras					
CMQC	Definiciones MQI	C	C	C	C
CMQBC	Definiciones de MQAI	C	C	C	
CMQEC	Definición de puntos de entrada de interfaz (incluye CMQC, CMQXC y CMQZC)		C	C	
CMQCFC	PCF, definiciones	C	C	C	C
CMQPSC	Definiciones de publicación/ suscripción	C	C	C	C
CMQXC	Definiciones de canal y salida	C	C	C	C
CMQZC	Definiciones de servicios instalables	C	C	C	
Clave: C= Archivos proporcionados					

Archivos de copia COBOL

Se proporcionan varios archivos COPY para ayudarle a escribir programas de aplicación COBOL que utilizan MQI. Estos archivos se resumen en la tabla:

Tabla 2. Archivos de copia COBOL: códigos de retorno, constantes y estructuras

Nombre de archivo	Descripción	IBM i	Sistemas UNIX	Windows	z/OS
Códigos de retorno y constantes					
CMQx	Definiciones MQI	V	V	V	V
CMQCFx	PCF, definiciones	V	V	V	V
CMQPSx	Definiciones de publicación/ suscripción	V	V	V	V
CMQXx	Definiciones de canal y salida	V	V	V	V
Estructuras					
CMQAIRx	MQAIR-Registro de información de autenticación		V L	V L	
CMQBOx	MQBO-Opciones de inicio	V L	V L	V L	
CMQCDx	MQCD-Definición de canal	V L	V L	V L	V L
CMQCFBFx	MQCFBF-Parámetro de filtro de serie de bytes PCF	V L	V L	V L	V L

Tabla 2. Archivos de copia COBOL: códigos de retorno, constantes y estructuras (continuación)					
Nombre de archivo	**Descripción**	**IBM i**	**Sistemas UNIX**	**Windows**	**z/OS**
CMQCFBSx	MQCFBS-Parámetro de serie de bytes PCF	V L	V L	V L	V L
CMQCFGRx	MQCFGR-Parámetro de grupo PCF	V L	V L	V L	V L
CMQCFHx	MQCFH-cabecera PCF	V L	V L	V L	V L
CMQCFIFx	MQCFIF-Parámetro de filtro de enteros PCF	V L	V L	V L	V L
CMQCFILx	MQCFIL-Parámetro de lista de enteros PCF	V L	V L	V L	V L
CMQCFINx	MQCFIN-Parámetro entero PCF	V L	V L	V L	V L
CMQCFSFx	MQCFSF-Parámetro de filtro de serie PCF	V L	V L	V L	V L
CMQCFSLx	MQCFSL-Parámetro de lista de series PCF	V L	V L	V L	V L
CMQCFSTx	MQCFST-Parámetro de serie PCF	V L	V L	V L	V L
CMQCFXLx	MQCFIL64 -Parámetro de lista de enteros de PCF de 64 bits	V L	V L	V L	V L
CMQCFXNx	MQCFIN64 -Parámetro entero de PCF de 64 bits	V L	V L	V L	V L
CMQCHRVx	MQCHARV-Serie de longitud variable	V L	V L	V L	V L
CMQCIHx	MQCIH-Cabecera de puente CICS	V L	V L	V L	V L
CMQCNOx	MQCNO - Opciones de conexión	V L	V L	V L	V L
CMQCSPx	MQCSP-Parámetros de seguridad	V L	V L	V L	V L
CMQCXPx	MQCXP-Parámetros de salida de canal	V L			V L
CMQDHx	MQDH - Cabecera de distribución	V L	V L	V L	V L
CMQDLHx	MQDLH-Cabecera de mensaje no entregado	V L	V L	V L	V L
CMQDXPx	MQDXP-Parámetros de salida de conversión de datos	V L		V L	
CMQEPHx	MQEPH - Cabecera PCF incrustada	V L	V L	V L	V L
CMQGMOx	MQGMO-Obtener opciones de mensaje	V L	V L	V L	V L
CMQIIHx	MQIIH-Cabecera de información de IMS	V L	V L	V L	V L
CMQMDx	MQMD - Descriptor de mensaje	V L	V L	V L	V L
CMQMD1x	MQMD1 -Descriptor de mensaje versión 1	V L	V L	V L	V L

Tabla 2. Archivos de copia COBOL: códigos de retorno, constantes y estructuras (continuación)					
Nombre de archivo	**Descripción**	**IBM i**	**Sistemas UNIX**	**Windows**	**z/OS**
CMQMD2x	MQMD2 -Descriptor de mensaje versión 2	V L	V L	V L	V L
CMQMDEx	MQMDE-Descriptor de mensaje ampliado	V L	V L	V L	V L
CMQODx	MQOD-Descriptor de objeto	V L	V L	V L	V L
CMQORx	MQOR-Registro de objeto	V L	V L	V L	V L
CMQPMOx	MQPMO-Opciones de transferencia de mensajes	V L	V L	V L	V L
CMQRFHx	MQRFH - Cabecera de reglas y formato	V L	V L	V L	V L
CMQRFH2x	MQRFH2 – Reglas y formato de la cabecera 2	V L	V L	V L	V L
CMQRMHx	MQRMH - Cabecera de mensaje de referencia	V L	V L	V L	V L
CMQRRx	MQRR-Registro de respuesta	V L	V L	V L	
CMQSCOx	MQSCO-Opciones de configuración SSL		V L	V L	
CMQTMx	MQTM-Mensaje de desencadenante	V L		V L	V L
CMQTMCx	MQTMC-Carácter de mensaje de desencadenante	V L	V L		
CMQTMC2x	MQTMC2 -Mensaje de desencadenante de 2 caracteres	V L	V L	V L	V L
CMQWIHx	MQWIH - Cabecera de información de trabajo	V L	V L	V L	V L
CMQXQHx	MQXQH - Cabecera de cola de transmisión	V L	V L	V L	V L
Clave: • Archivos con valores iniciales proporcionados, x = V • Archivos sin valores iniciales proporcionados, x = L					

Archivos de inclusión PL/I

Se proporcionan los siguientes archivos INCLUDE para el lenguaje de programación PL/I. Estos archivos solo están disponibles en z/OS .

Tabla 3. Archivos de inclusión PL/I: tipos de datos, códigos de retorno, constantes y estructuras					
Nombre de archivo	**Descripción**	**IBM i**	**Sistemas UNIX**	**Windows**	**z/OS**
Tipos de datos, códigos de retorno, constantes y estructuras					
CMQP	Definiciones MQI				P
CMQCFP	PCF, definiciones				P

Tabla 3. Archivos de inclusión PL/I: tipos de datos, códigos de retorno, constantes y estructuras (continuación)					
Nombre de archivo	**Descripción**	**IBM i**	**Sistemas UNIX**	**Windows**	**z/OS**
CMQEPP	Definiciones de punto de entrada				P
CMQPSP	Definiciones de publicación/ suscripción				P
CMQXP	Definiciones de canal y salida				P
Clave: P= Archivo proporcionado					

Archivos de copia RPG

Se proporcionan los siguientes archivos COPY para el lenguaje de programación RPG. Estos archivos sólo están disponibles en IBM i.

Tabla 4. Archivos de copia RPG-códigos de retorno, constantes y estructuras					
Nombre de archivo	**Descripción**	**IBM i**	**Sistemas UNIX**	**Windows**	**z/OS**
Códigos de retorno y constantes					
CMQx	Definiciones MQI	G R			
CMQCFx	PCF, definiciones	G			
CMQPSx	Definiciones de publicación/ suscripción	G			
CMQXx	Definiciones de canal y salida	G R			
Estructuras					
CMQBOx	MQBO-Opciones de inicio	G H			
CMQCDx	MQCD-Definición de canal	G H R			
CMQCFBFx	MQCFBF-Parámetro de filtro de serie de bytes PCF	G H			
CMQCFBSx	MQCFBS-Parámetro de serie de bytes PCF	G H			
CMQCFGRx	MQCFGR-Parámetro de grupo PCF	G H			
CMQCFHx	MQCFH-cabecera PCF	G H			
CMQCFIFx	MQCFIF-Parámetro de filtro de enteros PCF	G H			
CMQCFILx	MQCFIL-Parámetro de lista de enteros PCF	G H			
CMQCFINx	MQCFIN-Parámetro entero PCF	G H			
CMQCFSFx	MQCFSF-Parámetro de filtro de serie PCF	G H			

Tabla 4. Archivos de copia RPG-códigos de retorno, constantes y estructuras (continuación)					
Nombre de archivo	**Descripción**	**IBM i**	**Sistemas UNIX**	**Windows**	**z/OS**
CMQCFSLx	MQCFSL-Parámetro de lista de series PCF	G H			
CMQCFSTx	MQCFST-Parámetro de serie PCF	G H			
CMQCFXLx	MQCFIL64 -Parámetro de lista de enteros de PCF de 64 bits	G H			
CMQCFXNx	MQCFIN64 -Parámetro entero de PCF de 64 bits	G H			
CMQCHARVx	MQCHARV-Serie de longitud variable	G H			
CMQCIHx	MQCIH-Cabecera de puente CICS	G H			
CMQCNOx	MQCNO - Opciones de conexión	G H			
CMQCSPx	MQCSP-Parámetros de seguridad	G H			
CMQCXPx	MQCXP-Parámetros de salida de canal	G H R			
CMQDHx	MQDH - Cabecera de distribución	G H R			
CMQDLHx	MQDLH-Cabecera de mensaje no entregado	G H R			
CMQDXPx	MQDXP-Parámetros de salida de conversión de datos	G H R			
CMQEPHx	MQEPH - Cabecera PCF incrustada	G H			
CMQGMOx	MQGMO-Obtener opciones de mensaje	G H R			
CMQIIHx	MQIIH-Cabecera de información de IMS	G H R			
CMQMDx	MQMD - Descriptor de mensaje	G H R			
CMQMD1x	MQMD1 -Descriptor de mensaje versión 1	G H R			
CMQMD2x	MQMD2 -Descriptor de mensaje versión 2	G H			
CMQMDEx	MQMDE-Descriptor de mensaje ampliado	G H R			
CMQODx	MQOD-Descriptor de objeto	G H R			
CMQORx	MQOR-Registro de objeto	G H R			
CMQPMOx	MQPMO-Opciones de transferencia de mensajes	G H R			
CMQPXPx	MQPXP-Parámetros de salida de direccionamiento de publicación/ suscripción	G H			

Tabla 4. Archivos de copia RPG-códigos de retorno, constantes y estructuras (continuación)					
Nombre de archivo	**Descripción**	**IBM i**	**Sistemas UNIX**	**Windows**	**z/OS**
CMQRFHx	MQRFH - Cabecera de reglas y formato	G H			
CMQRFH2x	MQRFH2 – Reglas y formato de la cabecera 2	G H			
CMQRMHx	MQRMH - Cabecera de mensaje de referencia	G H R			
CMQRRx	MQRR-Registro de respuesta	G H R			
CMQTMx	MQTM-Mensaje de desencadenante	G H R			
CMQTMCx	MQTMC-Carácter de mensaje de desencadenante	G H R			
CMQTMC2x	MQTMC2 -Mensaje de desencadenante de 2 caracteres	G H R			
CMQWIHx	MQWIH - Cabecera de información de trabajo	G H			
CMQXQHx	MQXQH - Cabecera de cola de transmisión	G H R			
Clave: • Archivo para enlace estático, inicializado, proporcionado x = G • Archivo para enlace estático, no inicializado, proporcionado x = H • Archivo para enlace dinámico, inicializado, proporcionado, x = R					

Archivos de módulo Visual Basic

Los archivos de cabecera (o formulario) se proporcionan para ayudarle a escribir programas de aplicación de Visual Basic que utilizan MQI. Estos archivos de cabecera se proporcionan sólo en versiones de 32 bits y se resumen en la tabla:

Tabla 5. Archivos de módulo de Visual Basic: declaraciones de llamada, tipos de datos, códigos de retorno, constantes y estructuras					
Nombre de archivo	**Descripción**	**IBM i**	**Sistemas UNIX and Linux**	**Windows**	**z/OS**
Declaraciones de llamada, tipos de datos, códigos de retorno, constantes y estructuras					
CMQB	Definiciones MQI			B	
CMQBB	Definiciones de MQAI			B	
CMQCFB	PCF, definiciones			B	
CMQXB	Definiciones de canal y salida			B	
Clave: B= Archivo proporcionado					

MQ_ * (longitudes de serie)

Tabla 6. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQ_ABEND_CODE_LENGTH	4	X'00000004'
MQ_ACCOUNTING_TOKEN_LENGTH	32	X'00000020'
MQ_APPL_IDENTITY_DATA_LENGTH	32	X'00000020'
MQ_APPL_NAME_LENGTH	28	X'0000001C'
MQ_APPL_ORIGIN_DATA_LENGTH	4	X'00000004'
MQ_APPL_TAG_LENGTH	28	X'0000001C'
MQ_LONGITUD_SUFIJO	2	X'00000002'
MQ_ATTENTION_ID_LENGTH	4	X'00000004'
MQ_AUTH_INFO_CONN_NAME_LENGTH	264	X'00000108'
MQ_AUTH_INFO_DESC_LENGTH	64	X'00000040'
MQ_AUTH_INFO_NAME_LENGTH	48	X'00000030'
MQ_AUTH_INFO_OCSP_URL_LENGTH	256	X'00000100'
MQ_AUTHENTICATOR_LENGTH	8	X'00000008'
MQ_AUTO_REORG_CATALOG_LENGTH	44	X'0000002C'
MQ_AUTO_REORG_TIME_LENGTH	4	X'00000004'
MQ_BATCH_INTERFACE_ID_LENGTH	8	X'00000008'
MQ_BRIDGE_NAME_LENGTH	24	X'00000018'
MQ_CANCEL_CODE_LENGTH	4	X'00000004'
MQ_CF_STRUC_DESC_LENGTH	64	X'00000040'
MQ_CF_STRUC_NAME_LENGTH	12	X'0000000C'
MQ_CHANNEL_DATE_LENGTH	12	X'0000000C'
MQ_CHANNEL_DESC_LENGTH	64	X'00000040'
MQ_CHANNEL_NAME_LENGTH	20	X'00000014'
MQ_CHANNEL_TIME_LENGTH	8	X'00000008'
MQ_CHINIT_SERVICE_PARM_LENGTH	32	X'00000020'
MQ_CICS_FILE_NAME_LENGTH	8	X'00000008'
MQ_LONGITUD_ID_CLIENTE	23	X'00000017'
MQ_CLUSTER_NAME_LENGTH	48	X'00000030'
MQ_CONN_NAME_LENGTH	264	X'00000108'
LONGITUD_CÓDIGO_CONEXIÓN	128	X'00000080'
MQ_CONNECTION_ID_LENGTH	24	X'00000018'
MQ_CORREL_ID_LENGTH	24	X'00000018'
MQ_CREATION_DATE_LENGTH	12	X'0000000C'
MQ_CREATION_TIME_LENGTH	8	X'00000008'
MQ_DATE_LENGTH	12	X'0000000C'
MQ_NOMBRE_DISTINGUIDO_LONGITUD	1024	X'00000400'
MQ_DNS_GROUP_NAME_LENGTH	18	X'00000012'
MQ_EXIT_DATA_LENGTH	32	X'00000020'

Tabla 6. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQ_EXIT_INFO_NAME_LENGTH	48	X'00000030'
MQ_EXIT_NAME_LENGTH	(value differs by platform or version)	
MQ_EXIT_PD_AREA_LENGTH	48	X'00000030'
MQ_EXIT_USER_AREA_LENGTH	16	X'00000010'
MQ_FACILITY_LENGTH	8	X'00000008'
MQ_FACILITY_LONGITUD_CLAVE	4	X'00000004'
MQ_FORMAT_LENGTH	8	X'00000008'
LONGITUD_FUNCIÓN_MQ_LONGITUD	4	X'00000004'
MQ_GROUP_ID_LENGTH	24	X'00000018'
MQ_LDAP_PASSWORD_LENGTH	32	X'00000020'
MQ_LISTENER_NAME_LENGTH	48	X'00000030'
MQ_LISTENER_DESC_LENGTH	64	X'00000040'
MQ_LOCAL_ADDRESS_LENGTH	48	X'00000030'
MQ_LTERM_OVERRIDE_LENGTH	8	X'00000008'
MQ_LU_NAME_LENGTH	8	X'00000008'
MQ_LUWID_LENGTH	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'
MQ_MAX_USER_ID_LENGTH	64	X'00000040'
MQ_MCA_NOMBRE_TRABAJO_LONGITUD	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)	
MQ_MFS_MAP_NAME_LENGTH	8	X'00000008'
MQ_MODE_NAME_LENGTH	8	X'00000008'
MQ_MSG_HEADER_LENGTH	4000	X'00000FA0'
MQ_MSG_ID_LENGTH	24	X'00000018'
MQ_MSG_TOKEN_LENGTH	16	X'00000010'
MQ_NAMELIST_DESC_LENGTH	64	X'00000040'
MQ_NAMELIST_NAME_LENGTH	48	X'00000030'
MQ_OBJECT_INSTANCE_ID_LENGTH	24	X'00000018'
MQ_OBJECT_NAME_LENGTH	48	X'00000030'
MQ_PASS_TICKET_APPL_LENGTH	8	X'00000008'
LONGITUD_CONTRASEÑA_MQ	12	X'0000000C'
MQ_PROCESS_APPL_ID_LENGTH	256	X'00000100'
MQ_PROCESS_DESC_LENGTH	64	X'00000040'
MQ_PROCESS_ENV_DATA_LENGTH	128	X'00000080'

Tabla 6. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQ_PROCESS_NAME_LENGTH	48	X'00000030'
MQ_PROCESS_USER_DATA_LENGTH	128	X'00000080'
MQ_NOMBRE_PROGRAMA_LENGTH	20	X'00000014'
MQ_PUT_APPL_NAME_LENGTH	28	X'0000001C'
MQ_PUT_DATE_LENGTH	8	X'00000008'
MQ_PUT_TIME_LENGTH	8	X'00000008'
MQ_Q_DESC_LENGTH	64	X'00000040'
MQ_Q_MGR_DESC_LENGTH	64	X'00000040'
MQ_Q_MGR_IDENTIFIER_LENGTH	48	X'00000030'
MQ_Q_MGR_NAME_LENGTH	48	X'00000030'
MQ_Q_NAME_LENGTH	48	X'00000030'
MQ_QSG_NAME_LENGTH	4	X'00000004'
MQ_REMOTE_SYS_ID_LENGTH	4	X'00000004'
MQ_SECURITY_ID_LENGTH	40	X'00000028'
MQ_SELECTOR_LENGTH	10240	X'00002800'
MQ_SERVICE_ARGS_LENGTH	255	X'000000FF'
MQ_SERVICE_LONGITUD_MANDATO	255	X'000000FF'
MQ_SERVICE_DESC_LENGTH	64	X'00000040'
MQ_SERVICE_NAME_LENGTH	32	X'00000020'
MQ_SERVICE_PATH_LENGTH	255	X'000000FF'
MQ_SERVICE_STEP_LENGTH	8	X'00000008'
MQ_SHORT_CONN_NAME_LENGTH	20	X'00000014'
MQ_SHORT_DNAME_LENGTH	256	X'00000100'
MQ_SSL_CIPHER_SPEC_LENGTH	32	X'00000020'
MQ_SSL_CRYPTO_HARDWARE_LENGTH	256	X'00000100'
MQ_SSL_HANDSHAKE_STAGE_LENGTH	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'
MQ_START_CODE_LENGTH	4	X'00000004'
MQ_STORAGE_CLASS_DESC_LENGTH	64	X'00000040'
MQ_STORAGE_CLASS_LENGTH	8	X'00000008'
MQ_SUB_IDENTITY_LENGTH	128	X'00000080'
LONGITUD_PUNTO_SUB_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'

Tabla 6. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQ_SUITE_B_NOT_AVAILABLE	0	X'00000000'
MQ_TCP_NAME_LENGTH	8	X'00000008'
MQ_TIME_LENGTH	8	X'00000008'
MQ_TOPIC_DESC_LENGTH	64	X'00000040'
MQ_NOMBRE_TEMA_LONGITUD	48	X'00000030'
MQ_TOPIC_STR_LENGTH	10240	X'00002800'
MQ_TOTAL_EXIT_DATA_LENGTH	999	X'000003E7'
MQ_TOTAL_EXIT_NAME_LENGTH	999	X'000003E7'
MQ_TP_NAME_LENGTH	64	X'00000040'
MQ_TPIPE_NAME_LENGTH	8	X'00000008'
MQ_TRAN_INSTANCE_ID_LENGTH	16	X'00000010'
MQ_TRANSACTION_ID_LENGTH	4	X'00000004'
MQ_TRIGGER_DATA_LENGTH	64	X'00000040'
MQ_TRIGGER_PROGRAM_NAME_LENGTH	8	X'00000008'
MQ_TRIGGER_TERM_ID_LENGTH	4	X'00000004'
MQ_TRIGGER_TRANS_ID_LENGTH	4	X'00000004'
MQ_USER_ID_LENGTH	12	X'0000000C'
MQ_LONGITUD_VERSIÓN	8	X'00000008'
MQ_XCF_GROUP_NAME_LENGTH	8	X'00000008'
MQ_XCF_MEMBER_NAME_LENGTH	16	X'00000010'

MQ_ * (Formato de mandato, longitudes de serie)

Tabla 7. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQ_ARCHIVE_PFX_LENGTH	36	X'00000024'
MQ_ARCHIVE_UNIT_LENGTH	8	X'00000008'
MQ_LONGITUD_ASID_LONGITUD	4	X'00000004'
MQ_AUTH_PROFILE_NAME_LENGTH	48	X'00000030'
MQ_CF_LEID_LENGTH	12	X'0000000C'
MQ_COMMAND_MQSC_LENGTH	32768	X'00008000'
MQ_DATA_SET_NAME_LENGTH	44	X'0000002C'
MQ_DB2_NAME_LENGTH	4	X'00000004'
MQ_DSG_NAME_LENGTH	8	X'00000008'
MQ_ENTITY_NAME_LENGTH	64	X'00000040'
MQ_ENV_INFO_LENGTH	96	X'00000060'
MQ_IP_ADDRESS_LENGTH	48	X'00000030'
MQ_LOG_CORREL_ID_LENGTH	8	X'00000008'
MQ_LOG_EXTENT_NAME_LENGTH	24	X'00000018'
MQ_LOG_PATH_LENGTH	1024	X'00000400'

Tabla 7. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQ_LRSN_LENGTH	12	X'0000000C'
MQ_NOMBRE_ORIGIN_LENGTH	8	X'00000008'
MQ_PSB_NAME_LENGTH	8	X'00000008'
MQ_PST_ID_LENGTH	8	X'00000008'
MQ_Q_MGR_CPF_LENGTH	4	X'00000004'
MQ_RESPONSE_ID_LENGTH	24	X'00000018'
LONGITUD_RBA_MQ	12	X'0000000C'
MQ_SECURITY_PROFILE_LENGTH	40	X'00000028'
MQ_SERVICE_COMPONENT_LENGTH	48	X'00000030'
MQ_SUB_NAME_LENGTH	10240	X'00002800'
MQ_SYSP_SERVICE_LENGTH	32	X'00000020'
MQ_SYSTEM_NAME_LENGTH	8	X'00000008'
MQ_TASK_NUMBER_LENGTH	8	X'00000008'
MQ_TPIPE_PFX_LENGTH	4	X'00000004'
MQ_UOW_ID_LENGTH	256	X'00000100'
MQ_USER_DATA_LENGTH	10240	X'00002800'
LONGITUD_VOLSER_MQ	6	X'00000006'

MQACH_ * (estructura de cabecera de área de cadena de salida de API)

Tabla 8. Estructuras de constantes	
Nombre	**Estructura**
MQACH_STRUC_ID	"ACH¬"
MQACH_STRUC_ID_ARRAY	'A','C','H','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 9. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQACH_VERSION_1	1	X'00000001'
VERSIÓN_ACTUAL_MQACH_VERSIÓN	1	X'00000001'
MQACH_LENGTH_1	(value differs by platform or version)	
MQACH_CURRENT_LENGTH	(value differs by platform or version)	

MQACT_ * (Señal de contabilidad)

Tabla 10. Nombres y valores de constantes	
Nombre	**Valor**
MQACT_NONE	X'00...00' (32 nulos)
MQACT_NONE_ARRAY	'\0','\0',... (32 nulos)

MQACT_ * (Formato de mandato, Opciones de acción)

Tabla 11. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQACT_FORCE_REMOVE	1	X'00000001'
MQACT_ADVANCE_LOG	2	X'00000002'
MQACT_COLLECT_STATISTICS	3	X'00000003'
MQACT_PUBSUB	4	X'00000004'

MQACTP_ * (Acción)

Tabla 12. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQACTP_NEW	0	X'00000000'
MQACTP_FORWARD	1	X'00000001'
MQACTP_REPLY	2	X'00000002'
MQACTP_REPORT	3	X'00000003'

MQACTT_ * (Tipos de señal de contabilidad)

Tabla 13. Valores de constantes

Nombre	Valor hexadecimal
MQACTT_UNKNOWN	X'00'
MQACTT_CICS_LUOW_ID	X'01'
MQACTT_OS2_DEFAULT	X'04'
MQACTT_DOS_DEFAULT	X'05'
MQACTT_UNIX_NUMERIC_ID	X'06'
MQACTT_OS400_ACCOUNT_TOKEN	X'08'
MQACTT_WINDOWS_DEFAULT	X'09'
MQACTT_NT_SECURITY_ID	X'0B'
MQACTT_USER	X'19'

MQADOPT_ * (Adoptar Nuevas Comprobaciones MCA y Adoptar nuevos tipos MCA)

Adoptar nuevas comprobaciones de MCA

Tabla 14. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQADOPT_CHECK_NONE	0	X'00000000'
MQADOPT_CHECK_ALL	1	X'00000001'
MQADOPT_CHECK_Q_MGR_NAME	2	X'00000002'
MQADOPT_CHECK_NET_ADDR	4	X'00000004'

Adoptar nuevos tipos de MCA

Tabla 15. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQADOPT_TYPE_NO	0	X'00000000'
MQADOPT_TYPE_ALL	1	X'00000001'
MQADOPT_TYPE_SVR	2	X'00000002'
MQADOPT_TYPE_SDR	4	X'00000004'
MQADOPT_TYPE_RCVR	8	X'00000008'
MQADOPT_TYPE_CLUSRCVR	16	X'00000010'

MQAIR_ * (Estructura de registro de información de autenticación)

Tabla 16. Estructuras de constantes	
Nombre	**Estructura**
MQAIR_ID_ESTRUCTURA	"AIR¬"
MQAIR_STRUC_ID_ARRAY	'A','I','R','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 17. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAIR_VERSION_1	1	X'00000001'
MQAIR_VERSION_2	2	X'00000002'
VERSIÓN_ACTUAL_MQAIR_VERSIÓN	2	X'00000002'

MQAIT_ * (Tipo de información de autenticación)

Tabla 18. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAIT_ALL	0	X'00000000'
MQAIT_CRL_LDAP	1	X'00000001'
MQAIT_OCSP	2	X'00000002'

MQAS_ * (Formato de mandato, Valores de estado asíncrono)

Tabla 19. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAS_NONE	0	X'00000000'
MQAS_INICIADO	1	X'00000001'
MQAS_START_WAIT	2	X'00000002'
MQAS_DETENIDO	3	X'00000003'
MQAS_SUSPENDIDO	4	X'00000004'
MQAS_SUSPENDED_TEMPORARY	5	X'00000005'
MQAS_ACTIVO	6	X'00000006'
MQAS_INACTIVE	7	X'00000007'

MQAT_ * (Tipos de aplicación de colocación)

Tabla 20. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQAT_DESCONOCIDO	-1	X'FFFFFFFF'
MQAT_NO_CONTEXT	0	X'00000000'
MQAT_CICS	1	X'00000001'
MQAT_MVS	2	X'00000002'
MQAT_OS390	2	X'00000002'
MQAT_ZOS	2	X'00000002'
MQAT_IMS	3	X'00000003'
MQAT_OS2	4	X'00000004'
MQAT_DOS	5	X'00000005'
MQAT_AIX	6	X'00000006'
MQAT_UNIX	6	X'00000006'
MQAT_QMGR	7	X'00000007'
MQAT_OS400	8	X'00000008'
MQAT_WINDOWS	9	X'00000009'
MQAT_CICS_VSE	10	X'0000000A'
MQAT_WINDOWS_NT	11	X'0000000B'
MQAT_VMS	12	X'0000000C'
MQAT_GUARDIÁN	13	X'0000000D'
MQAT_NSK	13	X'0000000D'
MQAT_VOS	14	X'0000000E'
MQAT_OPEN_TP1	15	X'0000000F'
MQAT_VM	18	X'00000012'
MQAT_IMS_BRIDGE	19	X'00000013'
MQAT_XCF	20	X'00000014'
MQAT_CICS_BRIDGE	21	X'00000015'
MQAT_NOTES_AGENT	22	X'00000016'
MQAT_TPF	23	X'00000017'
USUARIO_MQ	25	X'00000019'
INTERMEDIARIO	26	X'0000001A'
MQAT_QMGR_PUBLISH	26	X'0000001A'
MQAT_JAVA	28	X'0000001C'
MQAT_DQM	29	X'0000001D'
MQAT_CHANNEL_INITIATOR	30	X'0000001E'
MQAT_WLM	31	X'0000001F'
MQAT_LOTE	32	X'00000020'
MQAT_RRS_BATCH	33	X'00000021'
MQAT_SIB	34	X'00000022'
MQAT_DEFAULT	(value differs by platform or version)	

Tabla 20. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAT_USER_FIRST	65536	X'00010000'
MQAT_USER_LAST	999999999	X'3B9AC9FF'

MQAUTH_ * (Formato de mandato, Valores de autorización)

Tabla 21. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAUTH_NONE	0	X'00000000'
MQAUTH_ALT_USER_AUTHORITY	1	X'00000001'
MQAUTH_BROWSE	2	X'00000002'
MQAUTH_CHANGE	3	X'00000003'
MQAUTH_CLEAR	4	X'00000004'
MQAUTH_CONNECT	5	X'00000005'
MQAUTH_CREAR	6	X'00000006'
MQAUTH_DELETE	7	X'00000007'
MQAUTH_DISPLAY	8	X'00000008'
MQAUTH_INPUT	9	X'00000009'
MQAUTH_INQUIRE	10	X'0000000A'
MQAUTH_OUTPUT	11	X'0000000B'
MQAUTH_PASS_ALL_CONTEXT	12	X'0000000C'
CONTRASEÑA_MQAUTH_IDENTITY_CONTEXT	13	X'0000000D'
MQAUTH_SET	14	X'0000000E'
MQAUTH_SET_TODO_CONTEXTO	15	X'0000000F'
Contexto de MQAUTH_SET_IDENTITY_CONTEXT	16	X'00000010'
CONTROL DE MQAUTOR	17	X'00000011'
MQAUTH_CONTROL_EXTENDED	18	X'00000012'
MQAUTH_PUBLISH	19	X'00000013'
MQAUTH_SUBSCRIBE	20	X'00000014'
MQAUTH_RESUME	21	X'00000015'
MQAUTH_SISTEMA	22	X'00000016'

MQAUTHOPT_ * (Formato de mandato, Opciones de autorización)

Tabla 22. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAUTHOPT_ACUMULATIVO	256	X'00000100'
MQAUTHOPT_ENTITY_EXPLICIT	1	X'00000001'
MQAUTHOPT_ENTITY_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_ * (estructura de contexto de salida de API)

Tabla 23. Estructuras de constantes	
Nombre	**Estructura**
MQAXC_STRUC_ID	"AXC¬"
MQAXC_STRUC_ID_ARRAY	'A','X','C','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 24. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAXC_VERSION_1	1	X'00000001'
MQAXC_VERSIÓN_ACTUAL	1	X'00000001'

MQAXP_ * (estructura de parámetros de salida de API)

Tabla 25. Estructuras de constantes	
Nombre	**Estructura**
MQAXP_ID_ESTRUCTURA	"AXP¬"
MQAXP_STRUC_ID_ARRAY	'A','X','P','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 26. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQAXP_VERSION_1	1	X'00000001'
MQAXP_VERSION_2	2	X'00000002'
VERSIÓN_ACTUAL_MQAXP	2	X'00000002'

MQBA_ * (Selectores de atributos de bytes)

Tabla 27. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBA_PRIMERO	6001	X'00001771'
MQBA_LAST	8000	X'00001F40'

MQBACF_ * (Formato de mandato, Tipos de parámetros de bytes)

Tabla 28. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBACF_PRIMERO	7001	X'00001B59'
MQBACF_EVENT_ACCOUNTING_TOKEN	7001	X'00001B59'
MQBACF_EVENT_SECURITY_ID	7002	X'00001B5A'
MQBACF_RESPONSE_SET	7003	X'00001B5B'
MQBACF_ID_RESPUESTA	7004	X'00001B5C'
MQBACF_EXTERNAL_UOW_ID	7005	X'00001B5D'
MQBACF_CONNECTION_ID	7006	X'00001B5E'
MQBACF_GENERIC_CONNECTION_ID	7007	X'00001B5F'

Tabla 28. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBACF_ORIGIN_UOW_ID	7008	X'00001B60'
MQBACF_Q_MGR_UOW_ID	7009	X'00001B61'
MQBACF_ACCOUNTING_TOKEN	7010	X'00001B62'
MQBACF_CORREL_ID	7011	X'00001B63'
MQBACF_GROUP_ID	7012	X'00001B64'
MQBACF_MSG_ID	7013	X'00001B65'
MQBACF_CF_LEID	7014	X'00001B66'
MQBACF_DESTINATION_CORREL_ID	7015	X'00001B67'
MQBACF_SUB_ID	7016	X'00001B68'
MQBACF_LAST_USED	7016	X'00001B68'

MQBL_ * (Longitud de almacenamiento intermedio para mqAddString y mqSetString)

Tabla 29. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBL_NULL_TERMINATED	-1	X'FFFFFFFF'

MQBMHO_ * (Almacenamiento intermedio para opciones y estructura de manejador de mensajes)

Estructura de opciones de almacenamiento intermedio a manejador de mensajes

Tabla 30. Estructuras de constantes	
Nombre	**Estructura**
MQBMHO_STRUC_ID	"BMHO"
MQBMHO_STRUC_ID_ARRAY	'B','M','H','O'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 31. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBMHO_VERSION_1	1	X'00000001'
MQBMHO_CURRENT_VERSION	1	X'00000001'

Opciones de manejador de almacenamiento intermedio a mensaje

Tabla 32. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQBMHO_NONE	0	X'00000000'
MQBMHO_DELETE_PROPERTIEs	1	X'00000001'

MQBND_ * (Enlaces predeterminados)

Tabla 33. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQBND_BIND_ON_OPEN	0	X'00000000'
MQBND_BIND_NOT_FIXED	1	X'00000001'
MQBND_BIND_ON_GROUP	2	X'00000002'

MQBO_ * (Iniciar opciones y estructura)

Estructura de opciones de inicio

Tabla 34. Estructuras de constantes

Nombre	Estructura
MQBO_STRUC_ID	"BO¬¬"
MQBO_STRUC_ID_ARRAY	'B','O','¬','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 35. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQBO_VERSION_1	1	X'00000001'
MQBO_CURRENT_VERSION	1	X'00000001'

Opciones de inicio

Tabla 36. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQBO_NONE	0	X'00000000'

MQBT_ * (Formato de mandato, Tipos de puente)

Tabla 37. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQBT_OTMA	1	X'00000001'

MQCA_ * (Selectores de atributos de caracteres)

Tabla 38. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
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'

Tabla 38. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
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'
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'

Tabla 38. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
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'
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'

Tabla 38. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
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'
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_ * (Formato de mandato, Tipos de parámetros de caracteres)

Tabla 39. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCACF_FIRST	3001	X'00000BB9'
MQCACF_FROM_Q_NAME	3001	X'00000BB9'
MQCACF_TO_Q_NAME	3002	X'00000BBA'
MQCACF_FROM_PROCESS_NAME	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'
MQCACF_Q_NAMES	3011	X'00000BC3'
MQCACF_PROCESS_NAMES	3012	X'00000BC4'
MQCACF_NAMELIST_NAMES	3013	X'00000BC5'
MQCACF_ESCAPE_TEXT	3014	X'00000BC6'
MQCACF_LOCAL_Q_NAMES	3015	X'00000BC7'
MQCACF_MODEL_Q_NAMES	3016	X'00000BC8'
MQCACF_ALIAS_Q_NAMES	3017	X'00000BC9'
MQCACF_REMOTE_Q_NAMES	3018	X'00000BCA'
MQCACF_SENDER_CHANNEL_NAMES	3019	X'00000BCB'

Tabla 39. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQCACF_SERVER_CHANNEL_NAMES	3020	X'00000BCC'
MQCACF_REQUESTER_CHANNEL_NAMES	3021	X'00000BCD'
MQCACF_RECEIVER_CHANNEL_NAMES	3022	X'00000BCE'
MQCACF_OBJECT_Q_MGR_NAME	3023	X'00000BCF'
MQCACF_APPL_NAME	3024	X'00000BD0'
MQCACF_USER_IDENTIFIER	3025	X'00000BD1'
MQCACF_AUX_ERROR_DATA_STR_1	3026	X'00000BD2'
MQCACF_AUX_ERROR_DATA_STR_2	3027	X'00000BD3'
MQCACF_AUX_ERROR_DATA_STR_3	3028	X'00000BD4'
MQCACF_BRIDGE_NAME	3029	X'00000BD5'
MQCACF_STREAM_NAME	3030	X'00000BD6'
MQCACF_TOPIC	3031	X'00000BD7'
MQCACF_PARENT_Q_MGR_NAME	3032	X'00000BD8'
MQCACF_ID_CORREL	3033	X'00000BD9'
MQCACF_PUBLISH_TIMESTAMP	3034	X'00000BDA'
MQCACF_STRING_DATA	3035	X'00000BDB'
MQCACF_SUPPORTED_STREAM_NAME	3036	X'00000BDC'
MQCACF_REG_TOPIC	3037	X'00000BDD'
MQCACF_REG_TIME	3038	X'00000BDE'
MQCACF_REG_USER_ID	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'
MQCACF_REG_CORREL_ID	3044	X'00000BE4'
MQCACF_EVENT_USER_ID	3045	X'00000BE5'
MQCACF_OBJECT_NAME	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_SUBSCRIPTION_NAME	3052	X'00000BEC'
MQCACF_REG_SUB_NAME	3053	X'00000BED'
MQCACF_SUBSCRIPTION_IDENTITY	3054	X'00000BEE'
MQCACF_REG_SUB_IDENTITY	3055	X'00000BEF'
MQCACF_SUBSCRIPTION_USER_DATA	3056	X'00000BF0'
MQCACF_REG_SUB_USER_DATA	3057	X'00000BF1'
MQCACF_APPL_TAG	3058	X'00000BF2'

Tabla 39. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQCACF_DATA_SET_NAME	3059	X'00000BF3'
MQCACF_UOW_START_DATE	3060	X'00000BF4'
MQCACF_UOW_START_TIME	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'
MQCACF_GROUP_ENTITY_NAMES	3066	X'00000BFA'
MQCACF_AUTH_PROFILE_NAME	3067	X'00000BFB'
MQCACF_XX_ENCODE_CASE_ONE nombre_entidad	3068	X'00000BFC'
MQCACF_SERVICE_COMPONENT	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'
VÍA DE ACCESO DE REGISTRO DE MQCACF_PATH	3074	X'00000C02'
MQCACF_COMMAND_MQSC	3075	X'00000C03'
MQCACF_Q_MGR_CPF	3076	X'00000C04'
MQCACF_USAGE_LOG_RBA	3078	X'00000C06'
MQCACF_USAGE_LOG_LRSN	3079	X'00000C07'
MQCACF_COMMAND_SCOPE	3080	X'00000C08'
MQCACF_ASID	3081	X'00000C09'
MQCACF_PSB_NAME	3082	X'00000C0A'
ID de MQCACF_PST_ID	3083	X'00000C0B'
MQCACF_TASK_NUMBER	3084	X'00000C0C'
MQCACF_TRANSACTION_ID	3085	X'00000C0D'
MQCACF_Q_MGR_UOW_ID	3086	X'00000C0E'
MQCACF_ORIGIN_NAME	3088	X'00000C10'
MQCACF_ENV_INFO	3089	X'00000C11'
MQCACF_SECURITY_PROFILE	3090	X'00000C12'
FECHA_CONFIGURACIÓN_MQCACF_DATE	3091	X'00000C13'
MQCACF_CONFIGURATION_TIME	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'
MQCACF_FAIL_DATE	3096	X'00000C18'
MQCACF_FAIL_TIME	3097	X'00000C19'
MQCACF_BACKUP_DATE	3098	X'00000C1A'
MQCACF_BACKUP_TIME	3099	X'00000C1B'

Tabla 39. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQCACF_SYSTEM_NAME	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_FROM_STORAGE_CLASS	3104	X'00000C20'
MQCACF_TO_STORAGE_CLASS	3105	X'00000C21'
MQCACF_STORAGE_CLASS_NAMES	3106	X'00000C22'
MQCACF_DSG_NAME	3108	X'00000C24'
MQCACF_DB2_NAME	3109	X'00000C25'
MQCACF_SYSP_CMD_USER_ID	3110	X'00000C26'
MQCACF_SYSP_OTMA_GROUP	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'
MQCACF_SYSP_LOG_CORREL_ID	3117	X'00000C2D'
MQCACF_SYSP_UNIT_VOLSER	3118	X'00000C2E'
MQCACF_SYSP_Q_MGR_TIME	3119	X'00000C2F'
MQCACF_SYSP_Q_MGR_DATE	3120	X'00000C30'
MQCACF_SYSP_Q_MGR_RBA	3121	X'00000C31'
MQCACF_SYSP_LOG_RBA	3122	X'00000C32'
MQCACF_SYSP_SERVICE	3123	X'00000C33'
MQCACF_FROM_LISTENER_NAME	3124	X'00000C34'
MQCACF_TO_LISTENER_NAME	3125	X'00000C35'
MQCACF_FROM_SERVICE_NAME	3126	X'00000C36'
MQCACF_TO_SERVICE_NAME	3127	X'00000C37'
MQCACF_LAST_PUT_DATE	3128	X'00000C38'
MQCACF_LAST_PUT_TIME	3129	X'00000C39'
MQCACF_LAST_GET_DATE	3130	X'00000C3A'
MQCACF_LAST_GET_TIME	3131	X'00000C3B'
MQCACF_OPERACIÓN_DATE	3132	X'00000C3C'
MQCACF_OPERACIÓN_TIME	3133	X'00000C3D'
MQCACF_ACTIVITY_DESC	3134	X'00000C3E'
MQCACF_APPL_IDENTITY_DATA	3135	X'00000C3F'
MQCACF_APPL_ORIGIN_DATA	3136	X'00000C40'
MQCACF_PUT_DATE	3137	X'00000C41'
MQCACF_PUT_TIME	3138	X'00000C42'
MQCACF_REPLY_TO_Q	3139	X'00000C43'

Tabla 39. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQCACF_REPLY_TO_Q_MGR	3140	X'00000C44'
MQCACF_RESOLVED_Q_NAME	3141	X'00000C45'
MQCACF_STRUC_ID	3142	X'00000C46'
MQCACF_VALOR_VALOR_Nombre	3143	X'00000C47'
MQCACF_SERVICE_START_DATE	3144	X'00000C48'
MQCACF_SERVICE_START_TIME	3145	X'00000C49'
MQCACF_SYSP_OFFLINE_RBA	3146	X'00000C4A'
MQCACF_SYSP_ARCHIVE_PFX2	3147	X'00000C4B'
MQCACF_SYSP_ARCHIVE_UNIT2	3148	X'00000C4C'
MQCACF_TO_TOPIC_NAME	3149	X'00000C4D'
MQCACF_FROM_TOPIC_NAME	3150	X'00000C4E'
MQCACF_TOPIC_NAMES	3151	X'00000C4F'
MQCACF_SUB_NAME	3152	X'00000C50'
MQCACF_DESTINATION_Q_MGR	3153	X'00000C51'
DESTINO_MQCACF_DESTINO	3154	X'00000C52'
MQCACF_SUB_USER_ID	3156	X'00000C54'
MQCACF_SUB_USER_DATA	3159	X'00000C57'
MQCACF_SUB_SELECTOR	3160	X'00000C58'
MQCACF_LAST_PUB_DATE	3161	X'00000C59'
MQCACF_LAST_PUB_TIME	3162	X'00000C5A'
MQCACF_FROM_SUB_NAME	3163	X'00000C5B'
MQCACF_TO_SUB_NAME	3164	X'00000C5C'
MQCACF_LAST_MSG_TIME	3167	X'00000C5F'
MQCACF_LAST_MSG_DATE	3168	X'00000C60'
MQCACF_SUBSCRIPTION_POINT	3169	X'00000C61'
FILTRO MQCACF_FILTER	3170	X'00000C62'
MQCACF_NONE	3171	X'00000C63'
MQCACF_ADMIN_TOPIC_NAMES	3172	X'00000C64'
MQCACF_LAST_USED	3172	X'00000C64'

MQCACH_ * (Formato de mandato, Tipos de parámetros de canal de caracteres)

Tabla 40. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCACH_FIRST	3501	X'00000DAD'
MQCACH_CHANNEL_NAME	3501	X'00000DAD'
MQCACH_DESC	3502	X'00000DAE'
MQCACH_MODE_NAME	3503	X'00000DAF'
MQCACH_TP_NAME	3504	X'00000DB0'
MQCACH_XMIT_Q_NAME	3505	X'00000DB1'

Tabla 40. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQCACH_CONNECTION_NAME	3506	X'00000DB2'
MQCACH_MCA_NAME	3507	X'00000DB3'
MQCACH_SEC_EXIT_NAME	3508	X'00000DB4'
MQCACH_MSG_EXIT_NAME	3509	X'00000DB5'
MQCACH_SEND_EXIT_NAME	3510	X'00000DB6'
MQCACH_RCV_EXIT_NAME	3511	X'00000DB7'
MQCACH_CHANNEL_NAMES	3512	X'00000DB8'
MQCACH_SEC_EXIT_USER_DATA	3513	X'00000DB9'
MQCACH_MSG_EXIT_USER_DATA	3514	X'00000DBA'
MQCACH_SEND_EXIT_USER_DATA	3515	X'00000DBB'
MQCACH_RCV_EXIT_USER_DATA	3516	X'00000DBC'
MQCACH_ID_USUARIO	3517	X'00000DBD'
CONTRASEÑA_MQCACH_PASSWORD	3518	X'00000DBE'
DIRECCIÓN_LOCAL_MQCACH_LOCAL	3520	X'00000DC0'
MQCACH_LOCAL_NAME	3521	X'00000DC1'
MQCACH_LAST_MSG_TIME	3524	X'00000DC4'
MQCACH_LAST_MSG_DATE	3525	X'00000DC5'
MQCACH_MCA_USER_ID	3527	X'00000DC7'
MQCACH_CHANNEL_START_TIME	3528	X'00000DC8'
MQCACH_CHANNEL_START_DATE	3529	X'00000DC9'
MQCACH_MCA_JOB_NAME	3530	X'00000DCA'
MQCACH_LAST_LUWID	3531	X'00000DCB'
MQCACH_CURRENT_LUWID	3532	X'00000DCC'
MQCACH_FORMAT_NAME	3533	X'00000DCD'
MQCACH_MR_EXIT_NAME	3534	X'00000DCE'
MQCACH_MR_EXIT_USER_DATA	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'
MQCACH_SSL_CERT_USER_ID	3549	X'00000DDD'
MQCACH_SSL_CERT_ISSUER_NAME	3550	X'00000DDE'
MQCACH_LU_NAME	3551	X'00000DDF'
DIRECCIÓN_IP_MQCACH_IP	3552	X'00000DE0'
MQCACH_TCP_NAME	3553	X'00000DE1'
MQCACH_LISTENER_NAME	3554	X'00000DE2'
MQCACH_LISTENER_DESC	3555	X'00000DE3'
MQCACH_LISTENER_START_DATE	3556	X'00000DE4'

Tabla 40. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCACH_LISTENER_START_TIME	3557	X'00000DE5'
MQCACH_SSL_KEY_RESET_DATE	3558	X'00000DE6'
MQCACH_SSL_KEY_RESET_TIME	3559	X'00000DE7'
MQCACH_LAST_USED	3559	X'00000DE7'

MQCADSD_ * (Descriptores ADS de cabecera de información CICS)

Tabla 41. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCADSD_NONE	0	X'00000000'
MQCADSD_SEND	1	X'00000001'
MQCADSD_RECV	16	X'00000010'
MQCADSD_MSGFORMAT	256	X'00000100'

MQCAFTY_ * (Valores de afinidad de conexión)

Tabla 42. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCAFTY_NONE	0	X'00000000'
MQCAFTY_PREFERIDO	1	X'00000001'

MQCAMO_ * (Formato de mandato, Tipos de parámetro de supervisión de caracteres)

Tabla 43. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCAMO_PRIMERO	2701	X'00000A8D'
MQCAMO_CLOSE_DATE	2701	X'00000A8D'
MQCAMO_CLOSE_TIME	2702	X'00000A8E'
MQCAMO_CONN_DATE	2703	X'00000A8F'
MQCAMO_CONN_TIME	2704	X'00000A90'
MQCAMO_DISC_DATE	2705	X'00000A91'
MQCAMO_DISC_TIME	2706	X'00000A92'
MQCAMO_END_DATE	2707	X'00000A93'
MQCAMO_END_TIME	2708	X'00000A94'
MQCAMO_OPEN_DATE	2709	X'00000A95'
MQCAMO_OPEN_TIME	2710	X'00000A96'
MQCAMO_START_DATE	2711	X'00000A97'
MQCAMO_START_TIME	2712	X'00000A98'
MQCAMO_LAST_USED	2712	X'00000A98'

MQCBC_ * (estructura de constantes MQCBC)

Tabla 44. Estructuras de constantes	
Nombre	**Estructura**
MQCBC_STRUC_ID	"CBC¬"
MQCBC_STRUC_ID_ARRAY	'C','B','C','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 45. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCBC_VERSION_1	1	X'00000001'
MQCBC_CURRENT_VERSION	1	X'00000001'

MQCBCF_ * (distintivos de constantes MQCBC)

Tabla 46. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCBCF_NONE	0	X'00000000'
MQCBCF_READA_BUFFER_EMPTY	1	X'00000001'

MQCBCT_ * (tipo de devolución de llamada de constantes MQCBC)

Tabla 47. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCBCT_START_CALL	1	X'00000001'
MQCBCT_STOP_CALL	2	X'00000002'
MQCBCT_REGISTER_CALL	3	X'00000003'
MQCBCT_DEREGISTER_CALL	4	X'00000004'
MQCBCT_EVENT_CALL	5	X'00000005'
MQCBCT_MSG_REMOVED	6	X'00000006'
MQCBCT_MSG_NOT_REMOVED	7	X'00000007'

MQCBD_ * (estructura de constantes MQCBD)

Tabla 48. Estructuras de constantes	
Nombre	**Estructura**
MQCBD_ID_STRUCD	"CBD¬"
MQCBD_STRUC_ID_ARRAY	'C','B','D','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 49. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCBD_VERSION_1	1	X'00000001'
MQCBD_CURRENT_VERSION	1	X'00000001'

MQCBDO_ * (opciones de devolución de llamada de constantes MQCBD)

Tabla 50. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCBDO_NONE	0	X'00000000'
MQCBDO_START_CALL	1	X'00000001'
MQCBDO_STOP_CALL	4	X'00000004'
MQCBDO_REGISTER_CALL	256	X'00000100'
MQCBDO_DEREGISTER_CALL	512	X'00000200'
MQCBDO_FAIL_IF_QUIESCING	8192	X'00002000'

MQCBO_ * (Opciones de creación de bolsa para mqCreatebolsa)

Tabla 51. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCBO_NONE	0	X'00000000'
MQCBO_USER_BAG	0	X'00000000'
MQCBO_ADMIN_BAG	1	X'00000001'
MQCBO_COMMAND_BAG	16	X'00000010'
MQCBO_SYSTEM_BAG	32	X'00000020'
MQCBO_GROUP_BAG	64	X'00000040'
MQCBO_LIST_FORM_ALLOWED	2	X'00000002'
MQCBO_LIST_FORM_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida	0	X'00000000'
MQCBO_REORDER_AS_REQUIRED	4	X'00000004'
MQCBO_DO_NOT_REORDER	0	X'00000000'
MQCBO_CHECK_SELECTORS	8	X'00000008'
MQCBO_DO_NOT_CHECK_SELECTORS	0	X'00000000'

MQCBT_ * (constantes de MQCBD Este es el tipo de función de devolución de llamada)

Tabla 52. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCBT_MESSAGE_CONSUMER	1	X'00000001'
MQCBT_EVENT_HANDLER	2	X'00000002'

MQCC_ * (Códigos de finalización)

Tabla 53. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCC_OK	0	X'00000000'
MQCC_WARNING	1	X'00000001'
MQCC_FAILED	2	X'00000002'
MQCC_DESCONOCIDO	-1	X'FFFFFFFF'

MQCCSI_ * (Identificadores de juego de caracteres codificados)

Tabla 54. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCCSI_UNDEFINED	0	X'00000000'
MQCCSI_PREDETERMINADO	0	X'00000000'
MQCCSI_Q_MGR	0	X'00000000'
MQCCSI_INHERIT	-2	X'FFFFFFFE'
MQCCSI_EMBEDDED	-1	X'FFFFFFFF'
MQCCSI_APPL	-3	X'FFFFFFFD'

MQCCT_ * (Opciones de tarea conversacional de cabecera de información CICS)

Tabla 55. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCCT_YES	1	X'00000001'
MQCCT_NO	0	X'00000000'

MQCD_ * (estructura de definición de canal)

Tabla 56. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
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_CURRENT_VERSION	9	X'00000009'
MQCD_LENGTH_4	(value differs by platform or version)	
MQCD_LENGTH_5	(value differs by platform or version)	
MQCD_LENGTH_6	(value differs by platform or version)	
MQCD_LENGTH_7	(value differs by platform or version)	
MQCD_LENGTH_8	(value differs by platform or version)	
MQCD_LENGTH_9	(value differs by platform or version)	
MQCD_LONGITUD_ACTUAL	(value differs by platform or version)	

MQCDC_ * (Conversión de datos de canal)

Tabla 57. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCDC_SENDER_CONVERSION	1	X'00000001'
MQCDC_NO_SENDER_CONVERSION	0	X'00000000'

MQCERT_ * (Tipo de política de validación de certificados)

MQ_CERT_VAL_POLICY_DEFAULT	0	X'00000000'
MQ_CERT_VAL_POLICY_ANY	0	X'00000000'
MQ_CERT_VAL_POLICY_RFC5280	1	X'00000001'

MQCF_ * (Distintivos de capacidad)

Tabla 58. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCF_NONE	0	X'00000000'
MQCF_DIST_LISTS	1	X'00000001'

MQCFAC_ * (Recurso de cabecera de información CICS)

Tabla 59. Nombres y valores de constantes

Nombre	Valor hexadecimal
MQCFAC_NONE	X'00...00' (8 nulos)
MQCFAC_NONE_ARRAY	'\0','\0',... (8 nulos)

MQCFBF_ * (Estructura de parámetros de filtro de serie de bytes de formato de mandato)

Tabla 60. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCFBF_STRUC_LENGTH_FIXED	20	X'00000014'

MQCFBS_ * (Estructura de parámetros de serie de bytes de formato de mandato)

Tabla 61. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCFBS_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFC_ * (Opciones de control de cabecera de formato de mandato)

Tabla 62. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCFC_LAST	1	X'00000001'
MQCFC_NOT_LAST	0	X'00000000'

MQCFGR_ * (Estructura de parámetros de grupo de formato de mandato)

Tabla 63. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFGR_STRUC_LENGTH	16	X'00000010'

MQCFH_ * (estructura de cabecera de formato de mandato)

Tabla 64. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFH_STRUC_LENGTH	36	X'00000024'
MQCFH_VERSION_1	1	X'00000001'
MQCFH_VERSION_2	2	X'00000002'
MQCFH_VERSION_3	3	X'00000003'
MQCFH_CURRENT_VERSION	3	X'00000003'

MQCFIF_ * (Estructura de parámetro de filtro de enteros de formato de mandato)

Tabla 65. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
LONGITUD_ESTRUCTURA_MQCFIF_LENGTH	20	X'00000014'

MQCFIL_ * (Estructura de parámetros de lista de enteros de formato de mandato)

Tabla 66. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFIL_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFIL64_* (Estructura de parámetros de lista de enteros de formato de mandato de 64 bits)

Tabla 67. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFIL64_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFIN_ * (Estructura de parámetro entero de formato de mandato)

Tabla 68. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFIN_STRUC_LENGTH	16	X'00000010'

MQCFIN64_* (Formato de mandato, estructura de parámetro de entero de 64 bits)

Tabla 69. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFIN64_STRUC_LENGTH	24	X'00000018'

MQCFO_ * (Formato de mandato, Renovar opciones de repositorio y Formato de mandato, Eliminar opciones de colas)

Opciones de repositorio de renovación de formato de mandato

Tabla 70. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFO_REFRESH_REPOSITORY_YES	1	X'00000001'
MQCFO_REFRESH_REPOSITORY_NO	0	X'00000000'

Formato de mandato Eliminar opciones de colas

Tabla 71. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFO_REMOVE_QUEUES_SÍ	1	X'00000001'
MQCFO_REMOVE_QUEUES_NO	0	X'00000000'

MQCFOP_ * (Formato de mandato, operadores de filtro)

Tabla 72. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFOP_LESS	1	X'00000001'
MQCFOP_EQUAL	2	X'00000002'
MQCFOP_MAYOR	4	X'00000004'
MQCFOP_NOT_LESS	6	X'00000006'
MQCFOP_NOT_EQUAL	5	X'00000005'
MQCFOP_NO_MAYOR	3	X'00000003'
MQCFOP_LIKE	18	X'00000012'
MQCFOP_NOT_LIKE	21	X'00000015'
MQCFOP_CONTAINS	10	X'0000000A'
MQCFOP_EXCLUDES	13	X'0000000D'
MQCFOP_CONTAINS_GEN	26	X'0000001A'
MQCFOP_EXCLUDES_GEN	29	X'0000001D'

MQCFR_ * (capacidad de recuperación de CF)

Tabla 73. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFR_SÍ	1	X'00000001'
MQCFR_NO	0	X'00000000'

MQCFSF_ * (Estructura de parámetros de filtro de serie de formato de mandato)

Tabla 74. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFSF_STRUC_LENGTH_FIXED	24	X'00000018'

MQCFSL_ * (Estructura de parámetros de lista de series de formato de mandato)

Tabla 75. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCFSL_STRUC_LENGTH_FIXED	24	X'00000018'

MQCFST_ * (Estructura de parámetros de serie de formato de mandato)

Tabla 76. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCFST_STRUC_LENGTH_FIXED	20	X'00000014'

MQCFSTATUS_ * (Formato de mandato, Estado de CF)

Tabla 77. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCFSTATUS_NOT_FOUND	0	X'00000000'
MQCFSTATUS_ACTIVE	1	X'00000001'
MQCFSTATUS_EN_RECOVER	2	X'00000002'
MQCFSTATUS_EN_BACKUP	3	X'00000003'
MQCFSTATUS_FAILED	4	X'00000004'
MQCFSTATUS_NONE	5	X'00000005'
MQCFSTATUS_DESCONOCIDO	6	X'00000006'
MQCFSTATUS_ADMIN_INCOMPLETO	20	X'00000014'
MQCFSTATUS_NEVER_USED	21	X'00000015'
MQCFSTATUS_NO_BACKUP	22	X'00000016'
MQCFSTATUS_NOT_FAILED	23	X'00000017'
MQCFSTATUS_NOT_RECOVERABLE	24	X'00000018'
MQCFSTATUS_XES_ERROR	25	X'00000019'

MQCFT_ * (Formato de mandato, Tipos de estructura)

Tabla 78. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCFT_NONE	0	X'00000000'
MQCFT_COMMAND	1	X'00000001'
MQCFT_RESPONSE	2	X'00000002'
MQCFT_INTEGER	3	X'00000003'
MQCFT_STRING	4	X'00000004'
MQCFT_INTEGER_LIST	5	X'00000005'
MQCFT_STRING_LIST	6	X'00000006'
MQCFT_EVENT	7	X'00000007'
MQCFT_USER	8	X'00000008'
MQCFT_BYTE_STRING	9	X'00000009'

Tabla 78. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFT_TRACE_ROUTE	10	X'0000000A'
MQCFT_REPORT	12	X'0000000C'
MQCFT_INTEGER_FILTER	13	X'0000000D'
MQCFT_STRING_FILTER	14	X'0000000E'
MQCFT_BYTE_STRING_FILTER	15	X'0000000F'
MQCFT_COMMAND_XR	16	X'00000010'
MQCFT_XR_MSG	17	X'00000011'
MQCFT_XR_ITEM	18	X'00000012'
MQCFT_XR_SUMMARY	19	X'00000013'
MQCFT_GROUP	20	X'00000014'
MQCFT_STATISTICS	21	X'00000015'
MQCFT_CONTABILIDAD	22	X'00000016'
MQCFT_INTEGER64	23	X'00000017'
MQCFT_INTEGER64_LIST	25	X'00000019'

MQCFTYPE_ * (Formato de mandato, Tipos CF)

Tabla 79. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCFTYPE_APPL	0	X'00000000'
MQCFTYPE_ADMIN	1	X'00000001'

MQCFUNC_ * (Funciones de cabecera de información CICS)

Tabla 80. Estructuras de constantes	
Nombre	**Estructura**
MQCFUNC_MQCONN	"CONN"
MQCFUNC_MQGET	"GET¬"
MQCFUNC_MQINQ	"INQ¬"
MQCFUNC_MQOPEN	"OPEN"
MQCFUNC_MQPUT	"PUT¬"
MQCFUNC_MQPUT1	"PUT1"
MQCFUNC_NONE	"¬¬¬¬"
MQCFUNC_MQCONN_ARRAY	'C','O','N','N'
MQCFUNC_MQGET_ARRAY	'G','E','T','¬'
MQCFUNC_MQINQ_ARRAY	'I','N','Q','¬'
MQCFUNC_MQOPEN_ARRAY	'O','P','E','N'
MQCFUNC_MQPUT_ARRAY	'P','U','T','¬'
MQCFUNC_MQPUT1_ARRAY	'P','U','T','1'
MQCFUNC_NONE_ARRAY	'¬','¬','¬','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

MQCGWI_ * (Intervalo de espera de obtención de cabecera de información CICS)

Tabla 81. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCGWI_DEFAULT	-2	X'FFFFFFFE'

MQCHAD_ * (Definición automática de canal)

Tabla 82. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCHAD_DISABLED	0	X'00000000'
MQCHAD_ENABLED	1	X'00000001'

MQCHIDS_ * (Formato de mandato, Estado dudoso)

Tabla 83. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCHIDS_NOT_INDOUBT	0	X'00000000'
MQCHIDS_INDOUBT	1	X'00000001'

MQCHLD_ * (Formato de mandato, Disposiciones de canal)

Tabla 84. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCHLD_TODOS	-1	X'FFFFFFFF'
MQCHLD_DEFAULT	1	X'00000001'
MQCHLD_COMPARTIDO	2	X'00000002'
MQCHLD_PRIVATE	4	X'00000004'
MQCHLD_FIXSHARED	5	X'00000005'

MQCHS_* (Formato de mandato, Estado del canal)

Tabla 85. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
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'
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_ * (Formato de mandato, Opciones de reinicio compartido de canal)

Tabla 86. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCHSH_RESTART_NO	0	X'00000000'
MQCHSH_RESTART_YES	1	X'00000001'

MQCHSR_ * (Formato de mandato, Opciones de detención de canal)

Tabla 87. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCHSR_STOP_NO_SOLICITADO	0	X'00000000'
MQCHSR_STOP_SOLICITADO	1	X'00000001'

MQCHSSTATE_ * (Formato de mandato, Subestados de canal)

Tabla 88. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCHSSTATE_OTHER	0	X'00000000'
MQCHSSTATE_END_OF_BATCH	100	X'00000064'
MQCHSSTATE_ENVÍO	200	X'000000C8'
MQCHSSTATE_RECEPCIÓN	300	X'0000012C'
MQCHSSTATE_SERIALIZANDO	400	X'00000190'
MQCHSSTATE_RESYNCHING	500	X'000001F4'
MQCHSSTATE_LATIDO	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_CONECTANDO	1250	X'000004E2'
MQCHSSTATE_SSL_HANDSHAKING	1300	X'00000514'
MQCHSSTATE_NAME_SERVER	1400	X'00000578'
MQCHSSTATE_IN_MQPUT	1500	X'000005DC'
MQCHSSTATE_IN_MQGET	1600	X'00000640'
MQCHSSTATE_IN_MQI_CALL	1700	X'000006A4'
MQCHSSTATE_COMPRIMIENDO	1800	X'00000708'

MQCHT_ * (Tipos de canal)

Tabla 89. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCHT_SENDER	1	X'00000001'
MQCHT_SERVER	2	X'00000002'

Tabla 89. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCHT_RECEIVER	3	X'00000003'
MQCHT_REQUESTER	4	X'00000004'
MQCHT_ALL	5	X'00000005'
MQCHT_CLNTCONN	6	X'00000006'
MQCHT_SVRCONN	7	X'00000007'
MQCHT_CLUSRCVR	8	X'00000008'
MQCHT_CLUSSDR	9	X'00000009'

MQCHTAB_ * (Formato de mandato, Tipos de tabla de canal)

Tabla 90. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCHTAB_Q_MGR	1	X'00000001'
MQCHTAB_CLNTCONN	2	X'00000002'

MQCI_ * (Identificador de correlación)

Tabla 91. Nombres y valores de constantes	
Nombre	**Valor**
MQCI_NONE	X'00...00' (24 nulos)
MQCI_NONE_ARRAY	'\0','\0',... (24 nulos)
MQCI_SESIÓN_NUEVA	X'414D5121...'
MQCI_NEW_SESSION_ARRAY	'\x41','\x4D','\51','\x21',...

MQCIH_ * (Estructura de cabecera de información CICS y distintivos)

Estructura de cabecera de información de CICS

Tabla 92. Estructuras de constantes	
Nombre	**Estructura**
MQCIH_STRUC_ID	"CIH¬"
MQCIH_STRUC_ID_ARRAY	'C','I','H','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 93. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCIH_VERSION_1	1	X'00000001'
MQCIH_VERSION_2	2	X'00000002'
MQCIH_CURRENT_VERSION	2	X'00000002'
MQCIH_LENGTH_1	164	X'000000A4'
MQCIH_LENGTH_2	180	X'000000B4'
MQCIH_CURRENT_LENGTH	180	X'000000B4'

Distintivos de cabecera de información de CICS

Tabla 94. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCIH_NONE	0	X'00000000'
MQCIH_PASS_EXPIRATION	1	X'00000001'
MQCIH_UNLIMITED_EXPIRATION	0	X'00000000'
MQCIH_REPLY_WITHOUT_NULLS	2	X'00000002'
MQCIH_REPLY_WITH_NULLS	0	X'00000000'
MQCIH_SYNC_ON_RETURN	4	X'00000004'
MQCIH_NO_SYNC_ON_RETURN	0	X'00000000'

MQCLCT_ * (Tipos de memoria caché de clúster)

Tabla 95. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCLCT_STATIC	0	X'00000000'
MQCLCT_DYNAMIC	1	X'00000001'

MQCLRS_ * (Formato de mandato Borrar ámbito de serie de tema)

Tabla 96. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCLRS_LOCAL	1	X'00000001'
MQCLRS_GLOBAL	2	X'00000002'

MQCLRT_ * (Formato de mandato, Borrar tipo de serie de tema)

Tabla 97. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCLRT_RETENIDO	1	X'00000001'

MQCLT_ * (Tipos de enlace de cabecera de información CICS)

Tabla 98. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCLT_PROGRAM	1	X'00000001'
TRANSACCIÓN_MQC	2	X'00000002'

MQCLWL_ * (Carga de trabajo de clúster)

Tabla 99. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCLWL_USEQ_LOCAL	0	X'00000000'
MQCLWL_USEQ_ANY	1	X'00000001'
MQCLWL_USEQ_AS_Q_MGR	-3	X'FFFFFFFD'

MQCLXQ_ * (Tipo de cola de transmisión de clúster)

MQCLXQ_* son los valores que puede establecer en el atributo de gestor de colas DEFCLXQ . El atributo DEFCLXQ controla qué cola de transmisión seleccionan de forma predeterminada los canales de clúster emisor para obtener mensajes, para enviar los mensajes a los canales de clúster receptor.

Tabla 100. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCLXQ_SCTQ	0	X'00000000'
MQCLXQ_CHANNEL	1	X'00000001'

Referencia relacionada

"DefClusterXmitQueueTipo (MQLONG)" en la página 797
El atributo DefClusterXmitQueueTipo controla qué cola de transmisión seleccionan de forma predeterminada los canales de clúster emisor para obtener mensajes, para enviar los mensajes a los canales de clúster receptor.

Cambiar gestor de colas

Consultar gestor de colas

Consultar gestor de colas (Respuesta)

"MQINQ-Consultar atributos de objeto" en la página 686
La llamada MQINQ devuelve una matriz de enteros y un conjunto de series de caracteres que contienen los atributos de un objeto.

MQCMD_ * (Códigos de mandato)

Tabla 101. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCMD_NONE	0	X'00000000'
MQCMD_CHANGE_Q_MGR	1	X'00000001'
MQCMD_INQUIRE_Q_MGR	2	X'00000002'
MQCMD_PROCESO DE CAMBIO	3	X'00000003'
MQCMD_PROCESO	4	X'00000004'
MQCMD_XX_ENCODE_CASE_ONE create_proceso	5	X'00000005'
MQCMD_DELETE_PROCESS	6	X'00000006'
MQCMD_INQUIRE_PROCESO	7	X'00000007'
MQCMD_CHANGE_Q	8	X'00000008'
MQCMD_CLEAR_Q	9	X'00000009'
MQCMD_COPY_Q	10	X'0000000A'
MQCMD_CREATE_Q	11	X'0000000B'
MQCMD_DELETE_Q	12	X'0000000C'
MQCMD_INQUIRE_Q	13	X'0000000D'
MQCMD_REFRESH_Q_MGR	16	X'00000010'
MQCMD_RESET_Q_STATS	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_CHANNEL	21	X'00000015'

Tabla 101. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQCMD_COPY_CHANNEL	22	X'00000016'
MQCMD_CREATE_CHANNEL	23	X'00000017'
MQCMD_DELETE_CHANNEL	24	X'00000018'
MQCMD_INQUIRE_CHANNEL	25	X'00000019'
MQCMD_PING_CHANNEL	26	X'0000001A'
MQCMD_RESET_CHANNEL	27	X'0000001B'
MQCMD_START_CHANNEL	28	X'0000001C'
MQCMD_STOP_CHANNEL	29	X'0000001D'
MQCMD_START_CHANNEL_INIT	30	X'0000001E'
MQCMD_START_CHANNEL_LISTENER	31	X'0000001F'
MQCMD_XX_ENCODE_CASE_ONE nombre_cambio	32	X'00000020'
MQCMD_XX_ENCODE_CASE_ONE nombre_cópid_lista	33	X'00000021'
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_CHANNEL	39	X'00000027'
MQCMD_PING_Q_MGR	40	X'00000028'
MQCMD_INQUIRE_Q_STATUS	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'
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_SUSCRIPTOR	65	X'00000041'
MQCMD_REQUEST_UPDATE	66	X'00000042'
MQCMD_BROKER_INTERNAL	67	X'00000043'
MQCMD_ACTIVITY_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'

Tabla 101. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
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_CREATE_AUTH_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_CONNECTION	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_AUTH_REC	90	X'0000005A'
MQCMD_LOGGER_EVENT	91	X'0000005B'
MQCMD_RESET_Q_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'
MQCMD_INQUIRE_LISTENER_STATUS	98	X'00000062'
MQCMD_SUCESO_MANDATO	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'
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'
MQCMD_INQUIRE_ARCHIVE	114	X'00000072'
MQCMD_INQUIRE_CF_STRUC	115	X'00000073'

Tabla 101. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQCMD_INQUIRE_CF_STRUC_STATUS	116	X'00000074'
MQCMD_INQUIRE_CMD_SERVER	117	X'00000075'
MQCMD_INQUIRE_CHANNEL_INIT	118	X'00000076'
MQCMD_INQUIRE_QSG	119	X'00000077'
MQCMD_INQUIRE_LOG	120	X'00000078'
MQCMD_INQUIRE_SECURITY	121	X'00000079'
MQCMD_INQUIRE_STG_CLASS	122	X'0000007A'
MQCMD_INQUIRE_SYSTEM	123	X'0000007B'
MQCMD_INQUIRE_THREAD	124	X'0000007C'
MQCMD_INQUIRE_TRACE	125	X'0000007D'
MQCMD_INQUIRE_USAGE	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_SECURIDAD	133	X'00000085'
MQCMD_SET_ARCHIVE	134	X'00000086'
MQCMD_XX_ENCODE_CASE_CAPS_LOCK_ON registro de sesión	136	X'00000088'
MQCMD_SISTEMA	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'
MQCMD_STOP_TRACE	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'
MQCMD_CHANGE_SERVICE	149	X'00000095'
MQCMD_COPY_SERVICE	150	X'00000096'
MQCMD_CREATE_SERVICE	151	X'00000097'
MQCMD_DELETE_SERVICE	152	X'00000098'
MQCMD_INQUIRE_SERVICE	153	X'00000099'
MQCMD_INQUIRE_SERVICE_STATUS	154	X'0000009A'
MQCMD_START_SERVICE	155	X'0000009B'

Tabla 101. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQCMD_STOP_SERVICE	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_CREATE_LOG	162	X'000000A2'
MQCMD_STATISTICS_MQI	164	X'000000A4'
MQCMD_STATISTICS_Q	165	X'000000A5'
MQCMD_STATISTICS_CHANNEL	166	X'000000A6'
MQCMD_ACCOUNTING_MQI	167	X'000000A7'
MQCMD_ACCOUNTING_Q	168	X'000000A8'
MQCMD_INQUIRE_AUTH_SERVICE	169	X'000000A9'
MQCMD_CHANGE_TOPIC	170	X'000000AA'
TEMA_COPIA_MQCMD	171	X'000000AB'
MQCMD_CREATE_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'
MQCMD_DELETE_SUBSCRIPTION	179	X'000000B3'
MQCMD_COPY_SUBSCRIPTION	181	X'000000B5'
MQCMD_INQUIRE_SUB_STATUS	182	X'000000B6'
MQCMD_INQUIRE_TOPIC_STATUS	183	X'000000B7'
MQCMD_CLEAR_TOPIC_STRING	184	X'000000B8'
MQCMD_INQUIRE_PUBSUB_STATUS	185	X'000000B9'
MQCMD_PURGE_CHANNEL	195	X'000000C3'

MQCMDI_ * (Formato de mandato, Valores de información de mandato)

Tabla 102. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCMDI_CMDSCOPE_ACEPTADO	1	X'00000001'
MQCMDI_CMDSCOPE_GENERATED	2	X'00000002'
MQCMDI_CMDSCOPE_COMPLETED	3	X'00000003'
MQCMDI_QSG_DISP_COMPLETED	4	X'00000004'
MQCMDI_COMMAND_ACEPTADO	5	X'00000005'
MQCMDI_CLUSTER_REQUEST_QUEUED	6	X'00000006'

Tabla 102. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
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_CERO	14	X'0000000E'
MQCMDI_REFRESH_CONFIGURATION	16	X'00000010'
MQCMDI_SEC_SIGNOFF_ERROR	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_* (Niveles de mandatos)

Tabla 103. Nombres y valores de constantes	
Nombre	**Valor**
MQCMDL_LEVEL_1	100
MQCMDL_LEVEL_101	101
MQCMDL_LEVEL_110	110
MQCMDL_LEVEL_114	114
MQCMDL_LEVEL_120	120
MQCMDL_LEVEL_200	200
MQCMDL_LEVEL_201	201
MQCMDL_LEVEL_210	210
MQCMDL_LEVEL_211	211
MQCMDL_LEVEL_220	220
MQCMDL_LEVEL_221	221
MQCMDL_LEVEL_230	230
MQCMDL_LEVEL_320	320
MQCMDL_LEVEL_420	420
MQCMDL_LEVEL_500	500
MQCMDL_LEVEL_510	510
MQCMDL_LEVEL_520	520
MQCMDL_LEVEL_530	530
MQCMDL_LEVEL_531	531
MQCMDL_LEVEL_600	600
MQCMDL_LEVEL_700	700
MQCMDL_LEVEL_701	701
MQCMDL_LEVEL_710	710

Tabla 103. Nombres y valores de constantes (continuación)	
Nombre	**Valor**
MQCMDL_LEVEL_711	711
MQCMDL_LEVEL_750	750

MQCMHO_ * (Crear opciones y estructura de manejadores de mensajes)

Crear estructura de opciones de manejador de mensajes

Tabla 104. Estructuras de constantes	
Nombre	**Estructura**
MQCMHO_STRUC_ID	"CMHO"
MQCMHO_STRUC_ID_ARRAY	'C','M','H','O'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 105. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCMHO_VERSION_1	1	X'00000001'
MQCMHO_CURRENT_VERSION	1	X'00000001'

Crear opciones de manejador de mensajes

Tabla 106. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCMHO_DEFAULT_VALIDATION	0	X'00000000'
MQCMHO_NO_VALIDATION	1	X'00000001'
MQCMHO_VALIDAR	2	X'00000002'
MQCMHO_NONE	0	X'00000000'

MQCNO_ * (Opciones de conexión y estructura)

Estructura de opciones de conexión

Tabla 107. Estructuras de constantes	
Nombre	**Estructura**
MQCNO_STRUC_ID	"CNO¬"
MQCNO_STRUC_ID_ARRAY	'C','N','O','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 108. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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'

Tabla 108. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCNO_CURRENT_VERSION	5	X'00000005'

Opciones de conexión

Tabla 109. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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'
MQCNO_LOCAL_BINDING	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_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_ * (Opciones de cierre)

Tabla 110. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCO_INMEDIATO	0	X'00000000'
MQCO_NONE	0	X'00000000'
MQCO_DELETE	1	X'00000001'

Tabla 110. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCO_DELETE_PURGE	2	X'00000002'
MQCO_KEEP_SUB	4	X'00000004'
MQCO_REMOVE_SUB	8	X'00000008'
MQCO_QUIESCE	32	X'00000020'

MQCODL_ * (Longitud de datos de salida de cabecera de información CICS)

Tabla 111. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCODL_AS_INPUT	-1	X'FFFFFFFF'

MQCOMPRESS_ * (Compresión de canal)

Tabla 112. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCOMPRESS_NO_DISPONIBLE	-1	X'FFFFFFFF'
MQCOMPRESS_NONE	0	X'00000000'
MQCOMPRESS_RLE	1	X'00000001'
MQCOMPRESS_ZLIBFAST	2	X'00000002'
MQCOMPRESS_ZLIBHIGH	4	X'00000004'
MQCOMPRESS_SISTEMA	8	X'00000008'
MQCOMPRESS_ANY	268435455	X'0FFFFFFF'

MQCONNID_ * (Identificador de conexión)

Tabla 113. Nombres y valores de constantes	
Nombre	**Valor**
MQCONNID_NONE	X'00...00' (24 nulos)
MQCONNID_NONE_ARRAY	'\0','\0',... (24 nulos)

MQCOPY_ * (Opciones de copia de propiedad)

Tabla 114. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCOPY_NONE	0	X'00000000'
MQCOPY_ALL	1	X'00000001'
REENVÍO de MQCOPY_FORWARD	2	X'00000002'
MQCOPY_PUBLISH	4	X'00000004'
MQCOPY_REPLY	8	X'00000008'
INFORME de MQCOPY_REPORT	16	X'00000010'
MQCOPY_DEFAULT	22	X'00000016'

MQCQT_ * (Tipos de cola de clúster)

Tabla 115. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCQT_LOCAL_Q	1	X'00000001'
MQCQT_ALIAS_Q	2	X'00000002'
MQCQT_REMOTE_Q	3	X'00000003'
MQCQT_Q_MGR_ALIAS	4	X'00000004'

MQCRC_ * (Códigos de retorno de cabecera de información CICS)

Tabla 116. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCRC_OK	0	X'00000000'
MQCRC_CICS_EXEC_ERROR	1	X'00000001'
MQCRC_MQ_API_ERROR	2	X'00000002'
ERROR DE MQCRC_BRIDGE_	3	X'00000003'
MQCRC_BRIDGE_ABEND	4	X'00000004'
MQCRC_APPLICATION_ABEND	5	X'00000005'
MQCRC_SECURITY_ERROR	6	X'00000006'
MQCRC_PROGRAM_NOT_AVAILABLE	7	X'00000007'
MQCRC_BRIDGE_TIMEOUT	8	X'00000008'
MQCRC_TRANSID_NOT_AVAILABLE	9	X'00000009'

MQCS_ * (estado de consumidor de constantes MQCBC)

Tabla 117. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCS_NONE	0	X'00000000'
MQCS_SUSPENDED_TEMPORARY	1	X'00000001'
MQCS_SUSPENDED_USER_ACTION	2	X'00000002'
MQCS_SUSPENDIDO	3	X'00000003'
MQCS_DETENIDO	4	X'00000004'

MQCSC_ * (Códigos de inicio de cabecera de información CICS)

Tabla 118. Estructuras de constantes

Nombre	Estructura
MQCSC_START	"S¬¬¬"
MQCSC_STARTDATA	"SD¬¬"
MQCSC_TERMINPUT	"TD¬¬"
MQCSC_NONE	"¬¬¬¬"
MQCSC_START_ARRAY	'S','¬','¬','¬'
MQCSC_STARTDATA_ARRAY	'S','D','¬','¬'
MQCSC_TERMINPUT_ARRAY	'T','D','¬','¬'

Tabla 118. Estructuras de constantes (continuación)	
Nombre	**Estructura**
MQCSC_NONE_ARRAY	'¬','¬','¬','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

MQCSP_ * (Estructura de parámetros de seguridad de conexión y tipos de autenticación)

Estructura de parámetros de seguridad de conexión

Tabla 119. Estructuras de constantes	
Nombre	**Estructura**
MQCSP_STRUC_ID	"CSP¬"
MQCSP_STRUC_ID_ARRAY	'C','S','P','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 120. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCSP_VERSION_1	1	X'00000001'
MQCSP_CURRENT_VERSION	1	X'00000001'

Parámetros de seguridad de conexión Tipos de autenticación

Tabla 121. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCSP_AUTH_NONE	0	X'00000000'
MQCSP_AUTH_USER_ID_AND_PWD	1	X'00000001'

MQCSRV_ * (Opciones de servidor de mandatos)

Tabla 122. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQCSRV_CONVERT_NO	0	X'00000000'
MQCSRV_CONVERT_YES	1	X'00000001'
MQCSRV_DLQ_NO	0	X'00000000'
MQCSRV_DLQ_YES	1	X'00000001'

MQCT_ * (Etiqueta de conexión de gestor de colas)

Tabla 123. Nombres y valores de constantes	
Nombre	**Valor**
MQCT_NONE	X'00...00' (128 nulos)
MQCT_NONE_ARRAY	'\0','\0',... (128 nulos)

MQCTES_ * (Estado de finalización de tarea de cabecera de información CICS)

Tabla 124. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCTES_NOSYNC	0	X'00000000'
MQCTES_COMMIT	256	X'00000100'
MQCTES_BACKOUT	4352	X'00001100'
MQCTES_ENDTASK	65536	X'00010000'

MQCTLO_ * (estructura de opciones MQCTL y opciones de control del consumidor)

Estructura de opciones MQCTL

Tabla 125. Estructuras de constantes

Nombre	Estructura
MQCTLO_STRUC_ID	"CTLO"
MQCTLO_STRUC_ID_ARRAY	'C','T','L','O'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 126. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCTLO_VERSION_1	1	X'00000001'
MQCTLO_CURRENT_VERSION	1	X'00000001'

Opciones de control de consumidor de MQCTL

Tabla 127. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCTLO_NONE	0	X'00000000'
MQCTLO_THREAD_AFFINITY	1	X'00000001'
MQCTLO_FAIL_IF_QUIESCING	8192	X'00002000'

MQCUOWC_ * (Controls de unidad de trabajo de cabecera de información CICS)

Tabla 128. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQCUOWC_ONLY	273	X'00000111'
MQCUOWC_CONTINUE	65536	X'00010000'
MQCUOWC_FIRST	17	X'00000011'
MQCUOWC_MIDDLE	16	X'00000010'
MQCUOWC_LAST	272	X'00000110'
MQCUOWC_COMMIT	256	X'00000100'
MQCUOWC_BACKOUT	4352	X'00001100'

MQCXP_ * (Estructura de parámetros de salida de canal)

Tabla 129. Estructuras de constantes	
Nombre	**Estructura**
MQCXP_STRUC_ID	"CXP¬"
MQCXP_STRUC_ID_ARRAY	'C','X','P','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 130. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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_CURRENT_VERSION	8	X'00000008'

MQDC_ * (Clase de destino)

Tabla 131. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDC_GESTIONADO	1	X'00000001'
MQDC_PROPORCIONADO	2	X'00000002'

MQDCC_ * (Opciones de conversión y máscaras y factores)

Opciones de conversión

Tabla 132. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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)	
MQDCC_SOURCE_ENC_NORMAL	16	X'00000010'
MQDCC_SOURCE_ENC_REVERSE	32	X'00000020'
MQDCC_SOURCE_ENC_UNDEFINED	0	X'00000000'
MQDCC_TARGET_ENC_NATIVE	(value differs by platform or version)	
MQDCC_TARGET_ENC_NORMAL	256	X'00000100'
MQDCC_TARGET_ENC_REVERSE	512	X'00000200'
MQDCC_TARGET_ENC_UNDEFINED	0	X'00000000'

Tabla 132. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDCC_NONE	0	X'00000000'

Máscaras y factores de opciones de conversión

Tabla 133. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDCC_SOURCE_ENC_MASK	240	X'000000F0'
MQDCC_TARGET_ENC_MASK	3840	X'00000F00'
MQDCC_SOURCE_ENC_FACTOR	16	X'00000010'
MQDCC_TARGET_ENC_FACTOR	256	X'00000100'

MQDELO_ * (Opciones de supresión de publicación/suscripción)

Tabla 134. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDELO_NONE	0	X'00000000'
MQDELO_LOCAL	4	X'00000004'

MQDH_ * (estructura de cabecera de distribución)

Tabla 135. Estructuras de constantes	
Nombre	**Estructura**
MQDH_STRUC_ID	"DH¬¬"
MQDH_STRUC_ID_ARRAY	'D','H','¬','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 136. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDH_VERSION_1	1	X'00000001'
MQDH_CURRENT_VERSION	1	X'00000001'

MQDHF_ * (distintivos de cabecera de distribución)

Tabla 137. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDHF_NEW_MSG_IDS	1	X'00000001'
MQDHF_NONE	0	X'00000000'

MQDISCONNECT_ * (Formato de mandato, Tipos de desconexión)

Tabla 138. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDISCONNECT_NORMAL	0	X'00000000'
MQDISCONNECT_IMPLICIT	1	X'00000001'
MQDISCONNECT_Q_MGR	2	X'00000002'

MQDL_ * (Listas de distribución)

Tabla 139. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQDL_SUPPORTED	1	X'00000001'
MQDL_NOT_SUPPORTED	0	X'00000000'

MQDLH_ * (estructura de cabecera de mensajes no entregados)

Tabla 140. Estructuras de constantes

Nombre	Estructura
MQDLH_STRUC_ID	"DLH¬"
MQDLH_STRUC_ID_ARRAY	'D','L','H','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 141. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQDLH_VERSION_1	1	X'00000001'
MQDLH_CURRENT_VERSION	1	X'00000001'

MQDLV_ * (Entrega de mensajes permanentes/no persistentes)

Tabla 142. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQDLV_AS_PARENT	0	X'00000000'
MQDLV_ALL	1	X'00000001'
MQDLV_ALL_DUR	2	X'00000002'
MQDLV_ALL_AVAIL	3	X'00000003'

MQDMHO_ * (Suprimir opciones y estructura de manejador de mensajes)

Suprimir estructura de opciones de manejador de mensajes

Tabla 143. Estructuras de constantes

Nombre	Estructura
MQDMHO_STRUC_ID	"DMHO"
MQDMHO_STRUC_ID_ARRAY	'D','M','H','O'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 144. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQDMHO_VERSION_1	1	X'00000001'
MQDMHO_CURRENT_VERSION	1	X'00000001'

Suprimir opciones de manejador de mensajes

Tabla 145. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDMHO_NONE	0	X'00000000'

MQDMPO_ * (Suprimir opciones y estructura de propiedad de mensaje)

Suprimir estructura de opciones de propiedad de mensaje

Tabla 146. Estructuras de constantes	
Nombre	**Estructura**
MQDMPO_STRUC_ID	"DMPO"
MQDMPO_STRUC_ID_ARRAY	'D','M','P','O'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 147. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDMPO_VERSION_1	1	X'00000001'
MQDMPO_CURRENT_VERSION	1	X'00000001'

Suprimir opciones de propiedad de mensaje

Tabla 148. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDMPO_DEL_FIRST	0	X'00000000'
MQDMPO_DEL_PROP_UNDER_CURSOR	1	X'00000001'
MQDMPO_NONE	0	X'00000000'

MQDNSWLM_ * (WLM de DNS)

Tabla 149. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDNSWLM_NO	0	X'00000000'
MQDNSWLM_SÍ	1	X'00000001'

MQDT_ * (Tipos de destino)

Tabla 150. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDT_APPL	1	X'00000001'
MQDT_BROKER	2	X'00000002'

MQDXP_ * (estructura de parámetros de salida de conversión)

Tabla 151. Estructuras de constantes	
Nombre	**Estructura**
MQDXP_STRUC_ID	"DXP¬"

Tabla 151. Estructuras de constantes (continuación)	
Nombre	**Estructura**
MQDXP_STRUC_ID_ARRAY	'D','X','P','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 152. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQDXP_VERSION_1	1	X'00000001'
MQDXP_VERSION_2	2	X'00000002'
MQDXP_CURRENT_VERSION	2	X'00000002'

MQEC_ * (Valores de señal)

Tabla 153. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQEC_MSG_LLEGADO	2	X'00000002'
MQEC_WAIT_INTERVAL_CADUCADO	3	X'00000003'
MQEC_WAIT_CANCELADO	4	X'00000004'
MQEC_Q_MGR_QUIESCING	5	X'00000005'
MQEC_CONNECTION_QUIESCING	6	X'00000006'

MQEI_ * (Caducidad)

Tabla 154. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQEI_UNLIMITED	-1	X'FFFFFFFF'

MQENC_ * (Codificación)

MQENC_ * (Codificación)

Tabla 155. Valores de constantes por plataforma			
Nombre	**Plataforma**	**Valor decimal**	**Valor hexadecimal**
MQENC_NATIVE	IBM i	273	X'00000111 "
	Linux	546	X'00000222 "
	Linux en SPARC	273	X'00000111 "
	Linux en x86	546	X'00000222 "
	Solaris en SPARC	273	X'00000111 "
	sistemas UNIX	273	X'00000111 "
	Windows	546	X'00000222 "
	Micro Focus COBOL en Windows	17	X'00000011 "
	z/OS	785	X'00000311 "

MQENC_ * (Máscaras de codificación)

Tabla 156. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MÁSCARA_ENTERO_MQENC_MÁSCARA	15	X'0000000F'
MÁSCARA_DECIMAL_MQENC_MÁSCARA	240	X'000000F0'
Máscara MQENC_FLOAT_MASK	3840	X'00000F00'
MÁSCARA_RESERVA_MQENC_RESERVADO	-4096	X'FFFFF000'

MQENC_ * (Codificaciones para enteros binarios)

Tabla 157. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQENC_INTEGER_UNDEFINED	0	X'00000000'
MQENC_INTEGER_NORMAL	1	X'00000001'
MQENC_INTEGER_REVERSED	2	X'00000002'

MQENC_ * (Codificaciones para enteros decimales empaquetados)

Tabla 158. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQENC_DECIMAL_UNDEFINED	0	X'00000000'
MQENC_DECIMAL_NORMAL	16	X'00000010'
MQENC_DECIMAL_REVERSED	32	X'00000020'

MQENC_ * (Codificaciones para números de coma flotante)

Tabla 159. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQENC_FLOAT_UNDEFINED	0	X'00000000'
MQENC_FLOAT_IEEE_NORMAL	256	X'00000100'
MQENC_FLOAT_IEEE_REVERSED	512	X'00000200'
MQENC_FLOAT_S390	768	X'00000300'
MQENC_FLOAT_TNS	1024	X'00000400'

MQEPH_ * (Estructura y distintivos de cabecera de formato de mandato incorporado)

Estructura de cabecera de formato de mandato incorporado

Tabla 160. Estructuras de constantes	
Nombre	**Estructura**
MQEPH_STRUC_ID	"EPH¬"
MQEPH_STRUC_ID_ARRAY	'E','P','H','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 161. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQEPH_STRUC_LENGTH_FIXED	68	X'00000044'
MQEPH_VERSION_1	1	X'00000001'
MQEPH_CURRENT_VERSION	1	X'00000001'

Distintivos de cabecera de formato de mandato incorporado

Tabla 162. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQEPH_NONE	0	X'00000000'
MQEPH_CCSID_EMBEDDED	1	X'00000001'

MQET_ * (Formato de mandato, Tipos de escape)

Tabla 163. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQET_MQSC	1	X'00000001'

MQEVO_ * (Formato de mandato, Orígenes de sucesos)

Tabla 164. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQEVO_OTHER	0	X'00000000'
MQEVO_CONSOLE	1	X'00000001'
MQEVO_INIT	2	X'00000002'
MQEVO_MSG	3	X'00000003'
MQEVO_MQSET	4	X'00000004'
MQEVO_INTERNAL	5	X'00000005'

MQEVR_ * (Formato de mandato, Registro de sucesos)

Tabla 165. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQEVR_DISABLED	0	X'00000000'
MQEVR_ENABLED	1	X'00000001'
MQEVR_EXCEPCIÓN	2	X'00000002'
MQEVR_NO_DISPLAY	3	X'00000003'

MQEXPI_ * (Intervalo de exploración de caducidad)

Tabla 166. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQEXPI_OFF	0	X'00000000'

MQFB_ * (Valores de comentarios)

Tabla 167. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQFB_NONE	0	X'00000000'
MQFB_SISTEMA_PRIMERO	1	X'00000001'
MQFB_QUIT	256	X'00000100'
MQFB_EXPIRATION	258	X'00000102'
MQFB_COA	259	X'00000103'
MQFB_COD	260	X'00000104'
MQFB_CHANNEL_COMPLETED	262	X'00000106'
MQFB_CHANNEL_FAIL_RETRY	263	X'00000107'
MQFB_CHANNEL_FAIL	264	X'00000108'
MQFB_APPL_CANNOT_BE_STARTED	265	X'00000109'
MQFB_TM_ERROR	266	X'0000010A'
MQFB_APPL_TYPE_ERROR	267	X'0000010B'
MQFB_STOPPED_BY_MSG_EXIT	268	X'0000010C'
MQFB_ACTIVITY	269	X'0000010D'
MQFB_XMIT_Q_MSG_ERROR	271	X'0000010F'
MQFB_PAN	275	X'00000113'
MQFB_NAN	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'
MQFB_BIND_OPEN_CLUSRCVR_DEL	281	X'00000119'
MQFB_MAX_ACTIVITIES	282	X'0000011A'
MQFB_NO_REENVIADO	283	X'0000011B'
MQFB_NO_ENTREGADO	284	X'0000011C'
MQFB_UNSUPPORTED_FORWARDING	285	X'0000011D'
MQFB_UNSUPPORTED_DELIVERY	286	X'0000011E'
MQFB_DATA_LENGTH_ZERO	291	X'00000123'
MQFB_DATA_LENGTH_NEGATIVO	292	X'00000124'
MQFB_DATA_LENGTH_TOO_BIG	293	X'00000125'
MQFB_BUFFER_OVERFLOW	294	X'00000126'
MQFB_LENGTH_OFF_BY_ONE	295	X'00000127'
MQFB_IIH_ERROR	296	X'00000128'
MQFB_NOT_AUTHORIZED_FOR_IMS	298	X'0000012A'
MQFB_IMS_ERROR	300	X'0000012C'
MQFB_IMS_FIRST	301	X'0000012D'
MQFB_IMS_LAST	399	X'0000018F'
MQFB_CICS_INTERNAL_ERROR	401	X'00000191'
MQFB_CICS_NOT_AUTHORIZED	402	X'00000192'

Tabla 167. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQFB_CICS_BRIDGE_FAILURE	403	X'00000193'
MQFB_CICS_CORREL_ID_ERROR	404	X'00000194'
MQFB_CICS_CCSID_ERROR	405	X'00000195'
MQFB_CICS_ENCODING_ERROR	406	X'00000196'
MQFB_CICS_CIH_ERROR	407	X'00000197'
MQFB_CICS_UOW_ERROR	408	X'00000198'
MQFB_CICS_COMMAREA_ERROR	409	X'00000199'
MQFB_CICS_APPL_NOT_STARTED	410	X'0000019A'
MQFB_CICS_APPL_ABENDED	411	X'0000019B'
MQFB_CICS_DLQ_ERROR	412	X'0000019C'
MQFB_CICS_UOW_BACKED_OUT	413	X'0000019D'
MQFB_PUBLICATIONS_ON_REQUEST	501	X'000001F5'
MQFB_SUBSCRIBER_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'
MQFB_APPL_LAST	999999999	X'3B9AC9FF'

MQFC_ * (Formato de mandato, Forzar opciones)

Tabla 168. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQFC_YES	1	X'00000001'
MQFC_NO	0	X'00000000'

MQFMT_ * (Formatos)

Tabla 169. Nombres y valores de constantes	
Nombre	**Valor**
MQFMT_NONE	"¬¬¬¬¬¬¬¬"
MQFMT_ADMIN	"MQADMIN¬"
MQFMT_CHANNEL_COMPLETED	"MQCHCOM¬"
MQFMT_CICS	"MQCICS¬¬"
MQFMT_COMMAND_1	"MQCMD1¬¬"
MQFMT_COMMAND_2	"MQCMD2¬¬"
MQFMT_DEAD_LETTER_HEADER	"MQDEAD¬¬"
MQFMT_DIST_HEADER	"MQHDIST¬"
MQFMT_EMBEDDED_PCF	"MQHEPCF¬"

Tabla 169. Nombres y valores de constantes (continuación)	
Nombre	**Valor**
MQFMT_EVENT	"MQEVENT¬"
MQFMT_IMS	"MQIMS¬¬¬"
MQFMT_IMS_VAR_STRING	"MQIMSVS¬"
MQFMT_MD_EXTENSION	"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¬¬"
MQFMT_WORK_INFO_HEADER	"MQHWIH¬¬"
MQFMT_XMIT_Q_HEADER	"MQXMIT¬¬"
MQFMT_NONE_ARRAY	'¬','¬','¬','¬','¬','¬','¬','¬'
MQFMT_ADMIN_ARRAY	'M','Q','A','D','M','I','N','¬'
MQFMT_CHANNEL_COMPLETED_ARRAY	'M','Q','C','H','C','O','M','¬'
MQFMT_CICS_ARRAY	'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','¬'
MQFMT_EVENT_ARRAY	'M','Q','E','V','E','N','T','¬'
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','¬','¬','¬'
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: El símbolo ¬ representa un único carácter en blanco.

MQGA_ * (Selectores de atributos de grupo)

Tabla 170. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQGA_PRIMERO	8001	X'00001F41'
MQGA_LAST	9000	X'00002328'

MQGACF_ * (Formato de mandato, Tipos de parámetro de grupo)

Tabla 171. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQGACF_PRIMERO	8001	X'00001F41'
CONTEXTO_MANDATO_MQGACF_MANDATO	8001	X'00001F41'
MQGACF_COMMAND_DATOS	8002	X'00001F42'
MQGACF_TRACE_ROUTE	8003	X'00001F43'
MQGACF_OPERACIÓN	8004	X'00001F44'
MQGACF_ACTIVIDAD	8005	X'00001F45'
MQGACF_EMBEDDED_MQMD	8006	X'00001F46'
MQGACF_MENSAJE	8007	X'00001F47'
MQGACF_MQMD	8008	X'00001F48'
MQGACF_VALOR_NAMING	8009	X'00001F49'
MQGACF_Q_ACCOUNTING_DATA	8010	X'00001F4A'
MQGACF_Q_STATISTICS_DATA	8011	X'00001F4B'
MQGACF_CHL_STATISTICS_DATA	8012	X'00001F4C'
MQGACF_LAST_USED	8012	X'00001F4C'

MQGI_ * (Identificador de grupo)

Tabla 172. Nombres y valores de constantes

Nombre	Valor
MQGI_NONE	X'00...00' (24 nulos)
MQGI_NONE_ARRAY	'\0','\0',... (24 nulos)

MQGMO_ * (Obtener opciones de mensaje y estructura)

Estructura de opciones de obtención de mensajes

Tabla 173. Estructuras de constantes

Nombre	Estructura
MQGMO_STRUC_ID	"GMO¬"
MQGMO_STRUC_ID_ARRAY	'G','M','O','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 174. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQGMO_VERSION_1	1	X'00000001'

Tabla 174. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQGMO_VERSION_2	2	X'00000002'
MQGMO_VERSION_3	3	X'00000003'
MQGMO_VERSION_4	4	X'00000004'
MQGMO_CURRENT_VERSION	4	X'00000004'

Obtener opciones de mensaje

Tabla 175. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQGMO_WAIT	1	X'00000001'
MQGMO_NO_WAIT	0	X'00000000'
MQGMO_SET_SIGNAL	8	X'00000008'
MQGMO_FAIL_IF_QUIESCING	8192	X'00002000'
MQGMO_SYNCPOINT	2	X'00000002'
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_NEXT	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'
MQGMO_LOCK	512	X'00000200'
MQGMO_UNLOCK	1024	X'00000400'
MQGMO_ACCEPT_TRUNCATED_MSG	64	X'00000040'
MQGMO_CONVERT	16384	X'00004000'
MQGMO_LOGICAL_ORDER	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'

Tabla 175. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQGMO_PROPERTIES_AS_Q_DEF	0	X'00000000'
MQGMO_NONE	0	X'00000000'

MQGS_ * (Estado de grupo)

Tabla 176. Nombres y valores de constantes	
Nombre	**Valor**
MQGS_NOT_IN_GROUP	'¬'
MQGS_MSG_IN_GROUP	'G'
MQGS_LAST_MSG_IN_GROUP	'L'

Nota: El símbolo ¬ representa un único carácter en blanco.

MQHA_ * (Selectores de descriptor de contexto)

Tabla 177. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQHA_PRIMERO	4001	X'00000FA1'
MQHA_BAG_HANDLE	4001	X'00000FA1'
MQHA_LAST_USED	4001	X'00000FA1'
MQHA_LAST	6000	X'00001770'

MQHB_ * (Descriptores de paquete)

Tabla 178. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQHB_UNUSABLE_HBAG	-1	X'FFFFFFFF'
MQHB_NONE	-2	X'FFFFFFFE'

MQHC_ * (Descriptores de contexto de conexión)

Tabla 179. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQHC_DEF_HCONN	0	X'00000000'
MQHC_UNUSABLE_HCONN	-1	X'FFFFFFFF'
MQHC_UNASSOCIATED_HCONN	-3	X'FFFFFFFD'

MQHM_ * (Descriptor de mensaje)

Tabla 180. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQHM_UNUSABLE_HMSG	-1	X'FFFFFFFF'
MQHM_NONE	0	X'00000000'

MQHO_ * (manejador de objetos)

Tabla 181. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQHO_UNUSABLE_HOBJ	-1	X'FFFFFFFF'
MQHO_NONE	0	X'00000000'

MQHSTATE_ * (Formato de mandato, Estados de manejador)

Tabla 182. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQHSTATE_INACTIVE	0	X'00000000'
MQHSTATE_ACTIVE	1	X'00000001'

MQIA_ * (Selectores de atributos enteros)

Tabla 183. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
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_TYPE	103	X'00000067'
MQIA_ADOPTNEWMCA_INTERVAL	104	X'00000068'
MQIA_APPL_TYPE	1	X'00000001'
MQIA_ARCHIVE	60	X'0000003C'
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'

Tabla 183. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
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'
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_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'
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'

Tabla 183. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQIA_INHIBIT_EVENT	48	X'00000030'
MQIA_INHIBIT_GET	9	X'00000009'
MQIA_INHIBIT_PUB	181	X'000000B5'
MQIA_INHIBIT_PUT	10	X'0000000A'
MQIA_INHIBIT_SUB	182	X'000000B6'
MQIA_INTRA_GROUP_QUEUING	64	X'00000040'
MQIA_IP_ADDRESS_VERSION	93	X'0000005D'
MQIA_LAST	2000	X'000007D0'
MQIA_LAST_USED	233	X'000000E9'
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'
MQIA_NAME_COUNT	19	X'00000013'
MQIA_NAMELIST_TYPE	72	X'00000048'
MQIA_NPM_CLASS	78	X'0000004E'

Tabla 183. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
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_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_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'
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'

Tabla 183. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
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_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'
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'

Tabla 183. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIA_TIME_SINCE_RESET	35	X'00000023'
MQIA_TOPIC_DEF_PERSISTENCE	185	X'000000B9'
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_ * (Formato de mandato, Tipos de parámetros enteros)

Tabla 184. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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'
MQIACF_FORCE	1005	X'000003ED'
MQIACF_REPLACE	1006	X'000003EE'
MQIACF_PURGE	1007	X'000003EF'
MQIACF_QUIESCE	1008	X'000003F0'
MQIACF_MODE	1008	X'000003F0'
MQIACF_ALL	1009	X'000003F1'
MQIACF_EVENT_APPL_TYPE	1010	X'000003F2'
MQIACF_EVENT_ORIGIN	1011	X'000003F3'
MQIACF_PARAMETER_ID	1012	X'000003F4'
MQIACF_ERROR_ID	1013	X'000003F5'
MQIACF_ERROR_IDENTIFIER	1013	X'000003F5'
MQIACF_SELECTOR	1014	X'000003F6'
MQIACF_CHANNEL_ATTRS	1015	X'000003F7'
MQIACF_OBJECT_TYPE	1016	X'000003F8'

Tabla 184. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQIACF_TIPO_ESCAPE_ES	1017	X'000003F9'
MQIACF_ERROR_OFFSET	1018	X'000003FA'
MQIACF_AUTH_INFO_ATTRS	1019	X'000003FB'
MQIACF_REASON_QUALIFIER	1020	X'000003FC'
MQIACF_COMMAND	1021	X'000003FD'
MQIACF_OPEN_OPTIONS	1022	X'000003FE'
MQIACF_OPEN_TYPE	1023	X'000003FF'
MQIACF_PROCESS_ID	1024	X'00000400'
MQIACF_THREAD_ID	1025	X'00000401'
MQIACF_Q_STATUS_ATTRS	1026	X'00000402'
MQIACF_UNCOMMITTED_MSGS	1027	X'00000403'
MQIACF_HANDLE_ESTADO	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'
MQIACF_BRIDGE_TYPE	1073	X'00000431'
MQIACF_CONSULTA	1074	X'00000432'
MQIACF_INTERVALO_ESPERA	1075	X'00000433'
MQIACF_OPCIONES	1076	X'00000434'
MQIACF_BROKER_OPCIONES	1077	X'00000435'
MQIACF_REFRESH_TYPE	1078	X'00000436'
MQIACF_SEQUENCE_NUMBER	1079	X'00000437'
MQIACF_DATOS	1080	X'00000438'
MQIACF_REGISTRATION_OPTIONS	1081	X'00000439'
MQIACF_PUBCIONTION_OPCIONES	1082	X'0000043A'
MQIACF_CLUSTER_INFO	1083	X'0000043B'
MQIACF_Q_MGR_DEFINITION_TYPE	1084	X'0000043C'
MQIACF_Q_MGR_TYPE	1085	X'0000043D'
MQIACF_ACCIÓN	1086	X'0000043E'
MQIACF_SUSPEND	1087	X'0000043F'
MQIACF_BROKER_COUNT	1088	X'00000440'
MQIACF_APPL_COUNT	1089	X'00000441'
MQIACF_ANONOUS_COUNT	1090	X'00000442'
MQIACF_REG_REG_OPCIONES	1091	X'00000443'
MQIACF_DELETE_OPCIONES	1092	X'00000444'
MQIACF_CLUSTER_Q_MGR_ATTRS	1093	X'00000445'
MQIACF_REFRESH_INTERVAL	1094	X'00000446'
MQIACF_REFRESH_REPOSITORY	1095	X'00000447'
MQIACF_REMOVE_QUEUES	1096	X'00000448'

Tabla 184. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQIACF_OPEN_INPUT_TYPE	1098	X'0000044A'
MQIACF_OPEN_OUTPUT	1099	X'0000044B'
MQIACF_OPEN_SET	1100	X'0000044C'
MQIACF_OPEN_INQUIRE	1101	X'0000044D'
MQIACF_OPEN_BROWSE	1102	X'0000044E'
MQIACF_Q_STATUS_TYPE	1103	X'0000044F'
MQIACF_Q_HANDLE	1104	X'00000450'
MQIACF_Q_STATUS	1105	X'00000451'
MQIACF_SECURITY_TYPE	1106	X'00000452'
MQIACF_CONNECTION_ATTRS	1107	X'00000453'
MQIACF_CONNECT_OPTIONS	1108	X'00000454'
MQIACF_CONN_INFO_TYPE	1110	X'00000456'
MQIACF_CONN_INFO_CONN	1111	X'00000457'
MQIACF_CONN_INFO_HANDLE	1112	X'00000458'
MQIACF_CONN_INFO_ALL	1113	X'00000459'
MQIACF_AUTH_PROFILE_ATTRS	1114	X'0000045A'
MQIACF_AUTORIZATION_LIST	1115	X'0000045B'
MQIACF_AUTH_ADD_AUTHS	1116	X'0000045C'
MQIACF_AUTH_REMOVE_AUTHS	1117	X'0000045D'
MQIACF_tipo_entidad	1118	X'0000045E'
MQIACF_COMMAND_INFO	1120	X'00000460'
MQIACF_CMDSCOPE_Q_MGR_COUNT	1121	X'00000461'
MQIACF_Q_MGR_SYSTEM	1122	X'00000462'
MQIACF_Q_MGR_EVENT	1123	X'00000463'
MQIACF_Q_MGR_DQM	1124	X'00000464'
MQIACF_Q_MGR_CLUSTER	1125	X'00000465'
MQIACF_QSG_DISPS	1126	X'00000466'
MQIACF_UOW_ESTADO	1128	X'00000468'
MQIACF_SECURITY_ITEM	1129	X'00000469'
MQIACF_CF_STRUC_STATUS	1130	X'0000046A'
MQIACF_TIPO_UOW	1132	X'0000046C'
MQIACF_CF_STRUC_ATTRS	1133	X'0000046D'
MQIACF_EXCLUDE_INTERVAL	1134	X'0000046E'
MQIACF_CF_STATUS_TYPE	1135	X'0000046F'
MQIACF_CF_STATUS_SUMMARY	1136	X'00000470'
MQIACF_CF_STATUS_CONNECT	1137	X'00000471'
MQIACF_CF_STATUS_BACKUP	1138	X'00000472'
MQIACF_CF_STRUC_TYPE	1139	X'00000473'
MQIACF_CF_STRUC_SIZE_MAX	1140	X'00000474'

Tabla 184. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQIACF_CF_STRUC_SIZE_USED	1141	X'00000475'
MQIACF_CF_STRUC_ENTRIES_MAX	1142	X'00000476'
MQIACF_CF_STRUC_ENTRIES_USED	1143	X'00000477'
MQIACF_CF_STRUC_BACKUP_SIZE	1144	X'00000478'
Tipo_MOVE_MQIACF_MQ	1145	X'00000479'
MQIACF_MOVE_TYPE_MOVE	1146	X'0000047A'
MQIACF_MOVE_TYPE_ADD	1147	X'0000047B'
MQIACF_Q_MGR_NUMBER	1148	X'0000047C'
MQIACF_Q_MGR_ESTADO	1149	X'0000047D'
MQIACF_DB2_CONN_STATUS	1150	X'0000047E'
MQIACF_SECURITY_ATTRS	1151	X'0000047F'
MQIACF_SECURITY_TIMEOUT	1152	X'00000480'
MQIACF_SECURITY_INTERVAL	1153	X'00000481'
MQIACF_SECURITY_SWITCH	1154	X'00000482'
MQIACF_SECURITY_SETTING	1155	X'00000483'
MQIACF_STORAGE_CLASS_ATTRS	1156	X'00000484'
tipo_USAG_MQIACF	1157	X'00000485'
MQIACF_BUFFER_POOL_ID	1158	X'00000486'
MQIACF_USAGE_TOTAL_PAGES	1159	X'00000487'
MQIACF_USAGE_UNUSED_PAGES	1160	X'00000488'
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'
MQIACF_PAGESET_STATUS	1165	X'0000048D'
MQIACF_USAGE_TOTAL_BUFFERS	1166	X'0000048E'
MQIACF_USAGE_DATA_SET_TYPE	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_COUNT	1171	X'00000493'
MQIACF_EXPIRY_Q_COUNT	1172	X'00000494'
MQIACF_CONFIGURATION_OBJECTS	1173	X'00000495'
MQIACF_CONFIGURATION_EVENTS	1174	X'00000496'
MQIACF_SYSP_TYPE	1175	X'00000497'
MQIACF_SYSP_DEALLOC_INTERVAL	1176	X'00000498'
MQIACF_SYSP_MAX_ARCHIVE	1177	X'00000499'
MQIACF_SYSP_MAX_READ_TAPES	1178	X'0000049A'
MQIACF_SYSP_IN_BUFFER_SIZE	1179	X'0000049B'

Tabla 184. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQIACF_SYSP_OUT_BUFFER_SIZE	1180	X'0000049C'
MQIACF_SYSP_OUT_BUFFER_COUNT	1181	X'0000049D'
MQIACF_SYSP_ARCHIVE	1182	X'0000049E'
MQIACF_SYSP_DUAL_ACTIVE	1183	X'0000049F'
MQIACF_SYSP_DUAL_ARCHIVE	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'
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'
MQIACF_SYSP_ROUTING_CODE	1196	X'000004AC'
MQIACF_SYSP_SMF_ACCOUNTING	1197	X'000004AD'
MQIACF_SYSP_SMF_STATS	1198	X'000004AE'
MQIACF_SYSP_SMF_INTERVAL	1199	X'000004AF'
MQIACF_SYSP_TRACE_CLASS	1200	X'000004B0'
MQIACF_SYSP_TRACE_SIZE	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'
MQIACF_SYSP_BLOCK_SIZE	1206	X'000004B6'
MQIACF_SYSP_CATALOG	1207	X'000004B7'
MQIACF_SYSP_COMPACT	1208	X'000004B8'
MQIACF_SYSP_ALLOC_PRIMARY	1209	X'000004B9'
MQIACF_SYSP_ALLOC_SECONDARY	1210	X'000004BA'
MQIACF_SYSP_PROTECT	1211	X'000004BB'
MQIACF_SYSP_QUIESCE_INTERVAL	1212	X'000004BC'
MQIACF_SYSP_TIMESTAMP	1213	X'000004BD'
MQIACF_SYSP_UNIT_ADDRESS	1214	X'000004BE'
MQIACF_SYSP_UNIT_STATUS	1215	X'000004BF'
MQIACF_SYSP_LOG_COPY	1216	X'000004C0'
MQIACF_SYSP_LOG_USED	1217	X'000004C1'
MQIACF_SYSP_LOG_SUSPEND	1218	X'000004C2'

Tabla 184. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQIACF_SYSP_OFFLOAD_STATUS	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_TIEMPO_INDICADOR	1226	X'000004CA'
MQIACF_OLDEST_MSG_AGE	1227	X'000004CB'
MQIACF_AUTO_OPCIONES	1228	X'000004CC'
MQIACF_Q_MGR_STATUS_ATTRS	1229	X'000004CD'
MQIACF_CONNECTION_COUNT	1230	X'000004CE'
MQIACF_Q_MGR_FACILITY	1231	X'000004CF'
MQIACF_CHINIT_STATUS	1232	X'000004D0'
MQIACF_CMD_SERVER_STATUS	1233	X'000004D1'
MQIACF_ROUTE_DETAIL	1234	X'000004D2'
MQIACF_RECORDED_ACTIVITIES	1235	X'000004D3'
MQIACF_MAX_ACTIVITIES	1236	X'000004D4'
MQIACF_DISCONTINUITY_COUNT	1237	X'000004D5'
MQIACF_ROUTE_ACUMULACIÓN_RUTA	1238	X'000004D6'
MQIACF_ROUTE_DELIVERY	1239	X'000004D7'
MQIACF_OPERACIÓN_TYPE	1240	X'000004D8'
MQIACF_BACKOUT_COUNT	1241	X'000004D9'
MQIACF_COMP_CODE	1242	X'000004DA'
MQIACF_ENCODING	1243	X'000004DB'
MQIACF_EXPIRY	1244	X'000004DC'
MQIACF_FEEDBACK	1245	X'000004DD'
MQIACF_MSG_FLAGS	1247	X'000004DF'
MQIACF_MSG_LENGTH	1248	X'000004E0'
MQIACF_MSG_TYPE	1249	X'000004E1'
MQIACF_OFFSET	1250	X'000004E2'
MQIACF_ORIGINAL_LENGTH	1251	X'000004E3'
MQIACF_PERSISTENCE	1252	X'000004E4'
MQIACF_PRIORITY	1253	X'000004E5'
MQIACF_REASON_CODE	1254	X'000004E6'
MQIACF_REPORT	1255	X'000004E7'
MQIACF_VERSION	1256	X'000004E8'
MQIACF_UNRECORDED_ACTIVITIES	1257	X'000004E9'
MQIACF_SUPERVISIÓN	1258	X'000004EA'

Tabla 184. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQIACF_ROUTE_FORWARDING	1259	X'000004EB'
MQIACF_SERVICE_STATUS	1260	X'000004EC'
MQIACF_Q_TYPES	1261	X'000004ED'
MQIACF_USER_ID_SUPPORT	1262	X'000004EE'
MQIACF_INTERFACE_VERSION	1263	X'000004EF'
MQIACF_AUTH_SERVICE_ATTRS	1264	X'000004F0'
MQIACF_USAGE_EXPAND_TYPE	1265	X'000004F1'
MQIACF_SYSP_CLUSTER_CACHE	1266	X'000004F2'
MQIACF_SYSP_DB2_BLOB_TASKS	1267	X'000004F3'
MQIACF_SYSP_WLM_INT_UNITS	1268	X'000004F4'
MQIACF_TOPIC_ATTRS	1269	X'000004F5'
MQIACF_PUBSUB_PROPERTIEs	1271	X'000004F7'
MQIACF_DESTINATION_CLASS	1273	X'000004F9'
MQIACF_DURABLE_SUBSCRIPTION	1274	X'000004FA'
MQIACF_SUBSCRIPTION_SCOPE	1275	X'000004FB'
MQIACF_VARIABLE_USER_ID	1277	X'000004FD'
MQIACF_REQUEST_ONLY	1280	X'00000500'
MQIACF_PUB_PRIORITY	1283	X'00000503'
MQIACF_SUB_ATTRS	1287	X'00000507'
MQIACF_WILDCARD_SCHEMA	1288	X'00000508'
MQIACF_SUB_TYPE	1289	X'00000509'
MQIACF_MESSAGE_COUNT	1290	X'0000050A'
MQIACF_Q_MGR_PUBSUB	1291	X'0000050B'
MQIACF_Q_MGR_VERSION	1292	X'0000050C'
MQIACF_SUB_STATUS_ATTRS	1294	X'0000050E'
MQIACF_TOPIC_STATUS	1295	X'0000050F'
MQIACF_TOPIC_SUB	1296	X'00000510'
MQIACF_TOPIC_PUB	1297	X'00000511'
MQIACF_RETAINED_PUBLICATION	1300	X'00000514'
MQIACF_TOPIC_STATUS_ATTRS	1301	X'00000515'
MQIACF_TOPIC_STATUS_TYPE	1302	X'00000516'
MQIACF_SUB_OPTIONS	1303	X'00000517'
MQIACF_PUBLISH_COUNT	1304	X'00000518'
MQIACF_TIPO_CLEAR	1305	X'00000519'
MQIACF_CLEAR_SCOPE	1306	X'0000051A'
MQIACF_SUB_LEVEL	1307	X'0000051B'
MQIACF_ASYNC_STATE	1308	X'0000051C'
MQIACF_SUB_SUMMARY	1309	X'0000051D'
MQIACF_OBSOLETE_MSGS	1310	X'0000051E'

Tabla 184. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIACF_PUBSUB_STATUS	1311	X'0000051F'
MQIACF_PS_STATUS_TYPE	1314	X'00000522'
MQIACF_PUBSUB_STATUS_ATTRS	1318	X'00000526'
Tipo_selección_MQIACF	1321	X'00000529'
MQIACF_MCAST_REL_INDICATOR	1351	X'00000547'
MQIACF_LAST_USED	1351	X'00000547'

MQIACH_ * (Formato de mandato, Tipos de canal entero)

Tabla 185. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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'
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'

Tabla 185. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
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'
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'

Tabla 185. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
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'
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'

Tabla 185. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
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'
MQIACH_CHANNEL_SUMMARY_ATTRS	1642	X'0000066A'
MQIACH_LAST_USED	1642	X'0000066A'

MQIAMO_ * (Formato de mandato, Tipos de parámetros de supervisión de enteros)

Tabla 186. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIAMO_PRIMERO	701	X'000002BD'
MQIAMO_AVG_BATCH_SIZE	702	X'000002BE'
MQIAMO_AVG_Q_TIME	703	X'000002BF'
MQIAMO_BACKOUTS	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_CLOSES	709	X'000002C5'
MQIAMO_COMMITS	710	X'000002C6'
MQIAMO_COMMITS_FAILED	711	X'000002C7'
MQIAMO_CONNS	712	X'000002C8'
MQIAMO_CONNS_MAX	713	X'000002C9'
MQIAMO_DISCS	714	X'000002CA'

Tabla 186. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQIAMO_DISCS_IMPLICIT	715	X'000002CB'
MQIAMO_DISC_TYPE	716	X'000002CC'
MQIAMO_EXIT_TIME_AVG	717	X'000002CD'
MQIAMO_EXIT_TIME_MAX	718	X'000002CE'
MQIAMO_EXIT_TIME_MIN	719	X'000002CF'
MQIAMO_FULL_BATCHES	720	X'000002D0'
MQIAMO_GENERATED_MSGS	721	X'000002D1'
MQIAMO_GETS	722	X'000002D2'
MQIAMO_GET_MAX_BYTES	723	X'000002D3'
MQIAMO_GET_MIN_BYTES	724	X'000002D4'
MQIAMO_GETS_FAILED	725	X'000002D5'
MQIAMO_INCOMPLETE_BATCHES	726	X'000002D6'
MQIAMO_INQS	727	X'000002D7'
MQIAMO_MSGS	728	X'000002D8'
MQIAMO_NET_TIME_AVG	729	X'000002D9'
MQIAMO_NET_TIME_MAX	730	X'000002DA'
MQIAMO_NET_TIME_MIN	731	X'000002DB'
MQIAMO_OBJECT_COUNT	732	X'000002DC'
MQIAMO_OPENS	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'
MQIAMO_SETS	744	X'000002E8'
MQIAMO_CONNS_FAILED	749	X'000002ED'
MQIAMO_OPENS_FAILED	751	X'000002EF'
MQIAMO_INQS_FAILED	752	X'000002F0'
MQIAMO_SETS_FAILED	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'

Tabla 186. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQIAMO_MSGS_PURGED	760	X'000002F8'
MQIAMO_SUBS_DUR	764	X'000002FC'
MQIAMO_SUBS_NDUR	765	X'000002FD'
MQIAMO_SUBS_FAILED	766	X'000002FE'
MQIAMO_SUBRQS	767	X'000002FF'
MQIAMO_SUBRQS_FAILED	768	X'00000300'
MQIAMO_CBS	769	X'00000301'
MQIAMO_CBS_FAILED	770	X'00000302'
MQIAMO_CTLS	771	X'00000303'
MQIAMO_CTLS_FAILED	772	X'00000304'
MQIAMO_STATS	773	X'00000305'
MQIAMO_STATS_FAILED	774	X'00000306'
MQIAMO_SUB_DUR_HIGHWATER	775	X'00000307'
MQIAMO_SUB_DUR_LOWWATER	776	X'00000308'
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_FAILED	788	X'00000314'
MQIAMO_INTERVALO	789	X'00000315'
MQIAMO_MSGS_SENT	790	X'00000316'
MQIAMO_BYTES_SENT	791	X'00000317'
MQIAMO_REPAIR_BYTES	792	X'00000318'
MQIAMO_FEEDBACK_MODE	793	X'00000319'
MQIAMO_RELIABILITY_TYPE	794	X'0000031A'
MQIAMO_LATE_JOIN_MARK	795	X'0000031B'
MQIAMO_NACKS_RCVD	796	X'0000031C'
MQIAMO_REPAIR_PKTS	797	X'0000031D'
MQIAMO_HISTORY_PKTS	798	X'0000031E'
MQIAMO_PENDING_PKTS	799	X'0000031F'
MQIAMO_PKT_RATE	800	X'00000320'
MQIAMO_MCAST_XMIT_RATE	801	X'00000321'
MQIAMO_MCAST_BATCH_TIME	802	X'00000322'
MQIAMO_MCAST_HEARTBEAT	803	X'00000323'

Tabla 186. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQIAMO_DEST_DATA_PORT	804	X'00000324'
MQIAMO_DEST_REPAIR_PORT	805	X'00000325'
MQIAMO_ACKS_RCVD	806	X'00000326'
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_TOTAL_BYTES_SENT	812	X'0000032C'
MQIAMO_NUM_STREAMS	813	X'0000032D'
Comentarios de MQIAMO_ACK_FEEDBACK	814	X'0000032E'
MQIAMO_NACK_FEEDBACK	815	X'0000032F'
MQIAMO_PKTS_LOST	816	X'00000330'
MQIAMO_MSGS_RCVD	817	X'00000331'
MQIAMO_MSG_BYTES_RCVD	818	X'00000332'
MQIAMO_MSGS_ENTREGADO	819	X'00000333'
MQIAMO_PKTS_PROCESSED	820	X'00000334'
MQIAMO_PKTS_DLVD	821	X'00000335'
MQIAMO_PKTS_CAÍDO	822	X'00000336'
MQIAMO_PKTS_DUPLICADAS	823	X'00000337'
MQIAMO_NACKS_CREATED	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'
MQIAMO_TOTAL_MSGS_PROCESSED	833	X'00000341'
MQIAMO_TOTAL_MSGS_SELECTED	834	X'00000342'
MQIAMO_TOTAL_MSGS_CADUCADO	835	X'00000343'
MQIAMO_TOTAL_MSGS_ENTREGADO	836	X'00000344'
MQIAMO_TOTAL_MSGS_DEVUELTO	837	X'00000345'
MQIAMO_LAST_USED	837	X'00000345'

MQIAMO64_* (Formato de mandato, tipos de parámetros de supervisión de enteros de 64 bits)

Tabla 187. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
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_ * (Selectores enteros del sistema)

Tabla 188. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQIASY_FIRST	-1	X'FFFFFFFF'
MQIASY_CODED_CHAR_SET_ID	-1	X'FFFFFFFF'
MQIASY_TYPE	-2	X'FFFFFFFE'
MQIASY_COMMAND	-3	X'FFFFFFFD'
MQIASY_MSG_SEQ_NUMBER	-4	X'FFFFFFFC'
CONTROL DE MQIASY_R	-5	X'FFFFFFFB'
CÓDIGO_EMPRESA	-6	X'FFFFFFFA'
MQIASY_REASON	-7	X'FFFFFFF9'
MQIASY_BAG_OPTIONS	-8	X'FFFFFFF8'
MQIASY_VERSION	-9	X'FFFFFFF7'
MQIASY_LAST_USED	-9	X'FFFFFFF7'
MQIASY_LAST	-2000	X'FFFFF830'

MQIAUT_ * (Autenticador de cabecera de información deIMS)

Tabla 189. Nombres y valores de constantes

Nombre	Valor
MQIAUT_NONE	"¬¬¬¬¬¬¬¬"
MQIAUT_NONE_ARRAY	'¬','¬','¬','¬','¬','¬','¬','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

MQIAV_ * (Valores de atributo entero)

Tabla 190. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQIAV_NOT_APPLICABLE	-1	X'FFFFFFFF'
MQIAV_UNDEFINED	-2	X'FFFFFFFE'

MQICM_ * (Modalidades de confirmación de cabecera de información deIMS)

Tabla 191. Nombres y valores de constantes

Nombre	Valor
MQICM_COMMIT_THEN_SEND	'0'
MQICM_SEND_THEN_COMMIT	'1'

MQIDO_ * (Formato de mandato, Opciones dudosas)

Tabla 192. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQIDO_COMMIT	1	X'00000001'
MQIDO_BACKOUT	2	X'00000002'

MQIEP_ * (Puntos de entrada de interfaz)

Estructura de parámetros de seguridad de conexión

Tabla 193. Estructuras de constantes

Nombre	Estructura
MQIEP_STRUC_ID	"IEP¬"
MQIEP_STRUC_ID_ARRAY	'I','E','P','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 194. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQIEP_VERSION_1	1	X'00000001'
MQDXP_CURRENT_VERSION	1	X'00000001'

MQIGQ_ * (En cola de grupo interno)

Tabla 195. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQIGQ_INHABILITADO	0	X'00000000'
MQIGQ_HABILITADO	1	X'00000001'

MQIGQPA_ * (Autoridad de colocación en cola de grupo interno)

Tabla 196. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQIGQPA_PREDETERMINADO	1	X'00000001'

Tabla 196. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
CONTEXTO_MQIGQPA	2	X'00000002'
MQIGQPA_ONLY_IGQ	3	X'00000003'
MQIGQPA_ALTERNATE_OR_IGQ	4	X'00000004'

MQIIH_ * (estructura y distintivos de cabecera de información deIMS)

Estructura de cabecera de información de IMS

Tabla 197. Estructuras de constantes	
Nombre	**Estructura**
MQIIH_STRUC_ID	"IIH¬"
MQIIH_STRUC_ID_ARRAY	'I','I','H','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 198. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIIH_VERSION_1	1	X'00000001'
MQIIH_CURRENT_VERSION	1	X'00000001'
MQIIH_LENGTH_1	84	X'00000054'

Distintivos de cabecera de información de IMS

Tabla 199. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIIH_NONE	0	X'00000000'
MQIIH_PASS_EXPIRATION	1	X'00000001'
MQIIH_UNLIMITED_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_ * (Consultar opciones y estructura de propiedad de mensaje)

Consultar estructura de opciones de propiedad de mensaje

Tabla 200. Estructuras de constantes	
Nombre	**Estructura**
MQIM_ID_ESTRUCTURA	"IMPO"
MQIM_STRUC_ID_ARRAY	'I','M','P','O'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 201. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIMPO_VERSION_1	1	X'00000001'

Tabla 201. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIM_VERSIÓN_ACTUAL	1	X'00000001'

Consultar opciones de propiedad de mensaje

Tabla 202. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIM_CONVERT_TYPE	2	X'00000002'
MQIM_QUERY_LENGTH	4	X'00000004'
MQIM_INQ_PRIMERO	0	X'00000000'
MQIM_INQ_NEXT	8	X'00000008'
MQIM_INQ_PROP_UNDER_CURSOR	16	X'00000010'
MQIM_CONVERT_VALUE	32	X'00000020'
MQIMPO_NONE	0	X'00000000'

MQINBD_ * (Formato de mandato, Disposiciones de entrada)

Tabla 203. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQINBD_Q_MGR	0	X'00000000'
MQINBD_XX_ENCODE_CASE_ONE grupo	3	X'00000003'

MQIND_ * (Valores de índice especiales)

Tabla 204. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIND_NONE	-1	X'FFFFFFFF'
MQIND_TODOS	-2	X'FFFFFFFE'

MQIPADDR_ * (Versiones de dirección IP)

Tabla 205. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQIPADDR_IPV4	0	X'00000000'
MQIPADDR_IPV6	1	X'00000001'

MQISS_ * (Ámbitos de seguridad de cabecera de información deIMS)

Tabla 206. Nombres y valores de constantes	
Nombre	**Valor**
MQISS_CHECK	'C'
MQISS_FULL	'F'

MQIT_ * (Tipos de índice)

Tabla 207. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQIT_NONE	0	X'00000000'
ID_MSG_MQIT	1	X'00000001'
ID_CORREL_MQIT	2	X'00000002'
MQIT_MSG_TOKEN	4	X'00000004'
ID_grupo_MQIT	5	X'00000005'

MQITEM_ * (Tipo de elemento para mqInquireItemInfo)

Tabla 208. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQITEM_INTEGER	1	X'00000001'
MQITEM_STRING	2	X'00000002'
MQITEM_BAG	3	X'00000003'
MQITEM_BYTE_STRING	4	X'00000004'
MQITEM_FILTRO	5	X'00000005'
MQITEM_STRING_FILTER	6	X'00000006'
MQITEM_INTEGER64	7	X'00000007'
MQITEM_BYTE_STRING_FILTER	8	X'00000008'

MQITII_ * (Identificador de instancia de transacción de cabecera de informaciónIMS)

Tabla 209. Nombres y valores de constantes

Nombre	Valor
MQITII_NONE	X'00...00' (16 nulos)
MQITII_NONE_ARRAY	'\0','\0',... (16 nulos)

MQITS_ * (Estados de transacción de cabecera de informaciónIMS)

Tabla 210. Nombres y valores de constantes

Nombre	Valor
MQITS_EN_CONVERSACIÓN	'C'
MQITS_NO_EN_CONVERSACIÓN	'¬'
MQITS_ARCHITECTED	'A'

Nota: El símbolo ¬ representa un único carácter en blanco.

MQKAI_ * (IntervaloKeepAlive)

Tabla 211. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQKAI_AUTO	-1	X'FFFFFFFF'

MQMASTER_ * (Administración maestra)

Tabla 212. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMASTER_NO	0	X'00000000'
MQMASTER_YES	1	X'00000001'

MQMCAS_ * (Formato de mandato, Estado del agente de canal de mensajes)

Tabla 213. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMCAS_STOPPED	0	X'00000000'
MQMCAS_RUNNING	3	X'00000003'

MQMCAT_ * (Tipos de MCA)

Tabla 214. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMCAT_PROCESO	1	X'00000001'
HEBRA MQMCAT_THREAD	2	X'00000002'

MQMCD_ * (Información de etiqueta de opciones de publicación/suscripción)

Opciones de publicación/suscripción Etiqueta Descriptor de contenido de mensaje (mcd) Etiquetas

Tabla 215. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
VERSIÓN_CARPETA_MQMCD_MQM	1	X'00000001'

Nombres de etiqueta de opciones de publicación/suscripción

Tabla 216. Nombres y valores de constantes	
Nombre	**Valor**
MQMCD_MSG_DOMAIN	"Msd"
MQMCD_MSG_SET	"Set"
MQMCD_MSG_TYPE	"Type"
MQMCD_MSG_FORMAT	"Fmt"

Nombres de etiqueta XML de etiquetas de opciones de publicación/suscripción

Tabla 217. Nombres y valores de constantes	
Nombre	**Valor**
MQMCD_MSG_DOMAIN_B	"<Msd>"
MQMCD_MSG_DOMAIN_E	"</Msd>"
MQMCD_MSG_SET_B	"<Set>"
MQMCD_MSG_SET_E	"</Set>"
MQMCD_MSG_TYPE_B	"<Type>"

Tabla 217. Nombres y valores de constantes (continuación)	
Nombre	**Valor**
MQMCD_MSG_TYPE_E	"</Type>"
MQMCD_MSG_FORMAT_B	"<Fmt>"
MQMCD_MSG_FORMAT_E	"</Fmt>"

Valores de etiqueta de opciones de publicación/suscripción

Tabla 218. Nombres y valores de constantes	
Nombre	**Valor**
MQMCD_DOMAIN_NONE	"none"
MQMCD_DOMAIN_NEON	"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"
MQMCD_DOMAIN_JMS_STREAM	"jms_stream"
MQMCD_DOMAIN_JMS_BYTES	"jms_bytes"

MQMD_ * (Estructura del descriptor de mensaje)

Tabla 219. Estructuras de constantes	
Nombre	**Estructura**
MQMD_STRUC_ID	"MD¬¬"
MQMD_STRUC_ID_ARRAY	'M','D','¬','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 220. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMD_VERSION_1	1	X'00000001'
MQMD_VERSION_2	2	X'00000002'
MQMD_CURRENT_VERSION	2	X'00000002'

MQMDE_ * (Estructura de extensión de descriptor de mensaje)

Tabla 221. Estructuras de constantes	
Nombre	**Estructura**
MQMDE_STRUC_ID	"MDE¬"
MQMDE_STRUC_ID_ARRAY	'M','D','E','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 222. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMDE_VERSION_2	2	X'00000002'

Tabla 222. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMDE_CURRENT_VERSION	2	X'00000002'
MQMDE_LENGTH_2	72	X'00000048'

MQMDEF_ * (Distintivos de extensión de descriptor de mensaje)

Tabla 223. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMDEF_NONE	0	X'00000000'

MQMDS_ * (secuencia de entrega de mensajes)

Tabla 224. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMDS_PRIORITY	0	X'00000000'
MQMDS_FIFO	1	X'00000001'

MQMF_ * (distintivos de mensaje)

Tabla 225. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMF_SEGMENTATION_INHIBIDO	0	X'00000000'
MQMF_SEGMENTATION_ALLOWED	1	X'00000001'
MQMF_MSG_IN_GROUP	8	X'00000008'
MQMF_LAST_MSG_IN_GROUP	16	X'00000010'
SEGMENTO MQMF_SEGMENT	2	X'00000002'
MQMF_LAST_SEGMENT	4	X'00000004'
MQMF_NONE	0	X'00000000'

MQMHBO_ * (Descriptor de mensaje para opciones y estructura de almacenamiento intermedio)

Estructura de manejadores de mensajes para opciones de almacenamiento intermedio

Tabla 226. Estructuras de constantes	
Nombre	**Estructura**
MQMHBO_STRUC_ID	"MHBO"
MQMHBO_STRUC_ID_ARRAY	'M','H','B','O'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 227. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMHBO_VERSION_1	1	X'00000001'
MQMHBO_CURRENT_VERSION	1	X'00000001'

Opciones de manejador de mensajes a almacenamiento intermedio

Tabla 228. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQMHBO_PROPERTIES_IN_MQRFH2	1	X'00000001'
MQMHBO_DELETE_PROPERTIES	2	X'00000002'
MQMHBO_NONE	0	X'00000000'

MQMI_ * (Identificador de mensaje)

Tabla 229. Nombres y valores de constantes

Nombre	Valor
MQMI_NONE	X'00...00' (24 nulos)
MQMI_NOE_ARRAY	'\0','\0',... (24 nulos)

MQMMBI_ * (Marca de mensaje-Intervalo de examen)

Tabla 230. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQMMBI_UNLIMITED	-1	X'FFFFFFFF'

MQMO_ * (Opciones de coincidencia)

Tabla 231. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQMO_MATCH_MSG_ID	1	X'00000001'
MQMO_MATCH_CORREL_ID	2	X'00000002'
MQMO_MATCH_GROUP_ID	4	X'00000004'
MQMO_MATCH_MSG_SEQ_NUMBER	8	X'00000008'
MQMO_MATCH_OFFSET	16	X'00000010'
MQMO_MATCH_MSG_TOKEN	32	X'00000020'
MQMO_NONE	0	X'00000000'

MQMODE_ * (Formato de mandato, Opciones de modalidad)

Tabla 232. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQMODE_FORCE	0	X'00000000'
MQMODE_QUIESCE	1	X'00000001'
MQMODE_TERMINAR	2	X'00000002'

MQMON_ * (Valores de supervisión)

Tabla 233. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQMON_NO_DISPONIBLE	-1	X'FFFFFFFF'
MQMON_NONE	-1	X'FFFFFFFF'

Tabla 233. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMON_Q_MGR	-3	X'FFFFFFFD'
MQMON_OFF	0	X'00000000'
MQMON_ON	1	X'00000001'
MQMON_INHABILITADO	0	X'00000000'
MQMON_HABILITADO	1	X'00000001'
MQMON_LOW	17	X'00000011'
MQMON_MEDIO	33	X'00000021'
MQMON_HIGH	65	X'00000041'

MQMT_ * (Tipos de mensaje)

Tabla 234. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQMT_SYSTEM_FIRST	1	X'00000001'
MQMT_REQUEST	1	X'00000001'
MQMT_REPLY	2	X'00000002'
MQMT_DATAGRAM	8	X'00000008'
MQMT_REPORT	4	X'00000004'
MQMT_MQE_FIELDS_FROM_MQE	112	X'00000070'
MQMT_MQE_FIELDS	113	X'00000071'
MQMT_SYSTEM_LAST	65535	X'0000FFFF'
MQMT_APPL_PRIMERO	65536	X'00010000'
MQMT_APPL_LAST	999999999	X'3B9AC9FF'

MQMTOK_ * (Señal de mensaje)

Tabla 235. Nombres y valores de constantes	
Nombre	**Valor**
MQMTOK_NONE	X'00...00' (16 nulos)
MQMTOK_NONE_ARRAY	'\0','\0',... (16 nulos)

MQNC_ * (Recuento de nombres)

Tabla 236. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQNC_MAX_NAMELIST_NAME_COUNT	256	X'00000100'

MQNPM_ * (Clase de mensaje no persistente)

Tabla 237. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQNPM_CLASS_NORMAL	0	X'00000000'
MQNPM_CLASS_HIGH	10	X'0000000A'

MQNPMS_ * (NonPersistent-XX_ENCODE_CASE_One Velocidades de mensajes)

Tabla 238. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQNPMS_NORMAL	1	X'00000001'
MQNPMS_FAST	2	X'00000002'

MQNT_ * (Tipos de lista de nombres)

Tabla 239. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQNT_NONE	0	X'00000000'
MQNT_Q	1	X'00000001'
MQNT_CLUSTER	2	X'00000002'
MQNT_AUTH_INFO	4	X'00000004'
MQNT_ALL	1001	X'000003E9'

MQNVS_ * (Nombres para serie de nombre/valor)

Tabla 240. Nombres y valores de constantes

Nombre	Valor
MQNVS_TIPO_APL	"OPT_APP_GRP¬"
MQNVS_TIPO_MSGS	"OPT_MSG_TYPE¬"

Nota: El símbolo ¬ representa un único carácter en blanco.

MQOA_ * (Límites para selectores para atributos de objeto)

Tabla 241. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQOA_PRIMERO	1	X'00000001'
MQOA_LAST	9000	X'00002328'

MQOD_ * (estructura de descriptor de objeto)

Tabla 242. Estructuras de constantes

Nombre	Estructura
MQOD_ID_STRUCD	"OD¬¬"
MQOD_STRUC_ID_ARRAY	'O','D','¬','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 243. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQOD_VERSION_1	1	X'00000001'
MQOD_VERSION_2	2	X'00000002'
MQOD_VERSION_3	3	X'00000003'

Tabla 243. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOD_VERSION_4	4	X'00000004'
MQOD_CURRENT_VERSION	4	X'00000004'
MQOD_LONGITUD_ACTUAL	(value differs by platform or version)	

MQOII_ * (Identificador de instancia de objeto)

Tabla 244. Nombres y valores de constantes	
Nombre	**Valor**
MQOII_NONE	X'00...00' (24 nulos)
MQOII_NONE_ARRAY	'\0','\0',... (24 nulos)

MQOL_ * (Longitud original)

Tabla 245. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOL_UNDEFINED	-1	X'FFFFFFFF'

MQOM_ * (Opciones de mensajes de Db2 obsoletas en el grupo de consultas)

Tabla 246. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOM_NO	0	X'00000000'
MQOM_SÍ	1	X'00000001'

MQOO_ * (Opciones de apertura)

Tabla 247. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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_BROWSE	8	X'00000008'
MQOO_OUTPUT	16	X'00000010'
MQOO_INQUIRE	32	X'00000020'
MQOO_SET	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'

Tabla 247. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOO_BIND_ON_OPEN	16384	X'00004000'
MQOO_BIND_NOT_FIXED	32768	X'00008000'
MQOO_CO_OP	131072	X'00020000'
MQOO_RESOLVE_LOCAL_TOPIC	262144	X'00040000'
MQOO_NO_READ_AHEAD	524288	X'00080000'
MQOO_READ_AHEAD	1048576	X'00100000'
MQOO_BIND_ON_GROUP	4194304	X'00400000'

MQOO_ * (sólo se utiliza lo siguiente en C++)

Tabla 248. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOO_RESOLVE_NAMES	65536	X'00010000'
MQOO_RESOLVE_LOCAL_Q	262144	X'00040000'

MQOP_ * (Códigos de operación para MQCTL y MQCB)

Códigos de operación para MQCTL

Tabla 249. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOP_START	1	X'00000001'
MQOP_START_WAIT	2	X'00000002'
MQOP_STOP	4	X'00000004'

Códigos de operación para MQCB

Tabla 250. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOP_REGISTRO	256	X'00000100'
MQOP_DEREGISTER	512	X'00000200'

Códigos de operación para MQCTL y MQCB

Tabla 251. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOP_SUSPENDER	65536	X'00010000'
MQOP_RESUME	131072	X'00020000'

MQOPEN_ * (Valores relacionados con la estructura MQOPEN_PRIV)

Tabla 252. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOPEN_PRIV_VERSION_1	1	X'00000001'
MQOPEN_PRIV_CURRENT_VERSION	1	X'00000001'

MQOPER_ * (Operaciones de actividad)

Tabla 253. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQOPER_SYSTEM_FIRST	0	X'00000000'
MQOPER_DESCONOCIDO	0	X'00000000'
MQOPER_EXAMINAR	1	X'00000001'
MQOPER_DISCARD	2	X'00000002'
MQOPER_GET	3	X'00000003'
MQOPER_PUT	4	X'00000004'
MQOPER_PUT_REPLY	5	X'00000005'
MQOPER_PUT_REPORT	6	X'00000006'
MQOPER_RECEIVE	7	X'00000007'
MQOPER_SEND	8	X'00000008'
MQOPER_TRANSFORM	9	X'00000009'
MQOPER_PUBLISH	10	X'0000000A'
MQOPER_EXCLUDED_PUBLISH	11	X'0000000B'
MQOPER_DESCARDED_PUBLISH	12	X'0000000C'
MQOPER_SYSTEM_LAST	65535	X'0000FFFF'
MQOPER_APPL_PRIMERO	65536	X'00010000'
MQOPER_APPL_LAST	999999999	X'3B9AC9FF'

MQOT_ * (Tipos de objeto y tipos de objeto ampliados)

Tipos de objeto

Tabla 254. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQOT_NONE	0	X'00000000'
MQOT_Q	1	X'00000001'
MQOT_NAMELIST	2	X'00000002'
MQOT_PROCESS	3	X'00000003'
Clase de almacenamiento MQOT_STORAGE_CLASS	4	X'00000004'
MQOT_Q_MGR	5	X'00000005'
MQOT_CHANNEL	6	X'00000006'
MQOT_AUTH_INFO	7	X'00000007'
MQOT_TOPIC	8	X'00000008'
MQOT_CF_STRUC	10	X'0000000A'
MQOT_ESCUCHA	11	X'0000000B'
SERVICIO MQOT_SERVICE	12	X'0000000C'
MQOT_RESERVED_1	999	X'000003E7'

Tipos de objetos ampliados

Tabla 255. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQOT_ALL	1001	X'000003E9'
MQOT_ALIAS_Q	1002	X'000003EA'
MQOT_MODEL_Q	1003	X'000003EB'
MQOT_LOCAL_Q	1004	X'000003EC'
MQOT_REMOTE_Q	1005	X'000003ED'
MQOT_SENDER_CHANNEL	1007	X'000003EF'
MQOT_SERVER_CHANNEL	1008	X'000003F0'
MQOT_CANAL_SOLICITANTE	1009	X'000003F1'
MQOT_RECEIVER_CHANNEL	1010	X'000003F2'
MQOT_CANAL_ACTUAL	1011	X'000003F3'
MQOT_SAVED_CHANNEL	1012	X'000003F4'
MQOT_SVRCONN_CHANNEL	1013	X'000003F5'
MQOT_CLNTCONN_CHANNEL	1014	X'000003F6'
MQOT_SHORT_CHANNEL	1015	X'000003F7'

MQPA_ * (Autorización de colocación)

Tabla 256. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPA_PREDETERMINADO	1	X'00000001'
CONTEXTO_MQPA	2	X'00000002'
MQPA_ONLY_MCA	3	X'00000003'
MQPA_ALTERNATE_OR_MCA	4	X'00000004'

MQPD_ * (Descriptor de propiedades, soporte y contexto)

Estructura del descriptor de propiedades

Tabla 257. Estructuras de constantes	
Nombre	**Estructura**
MQPD_STRUC_ID	"PD¬¬"
MQPD_STRUC_ID_ARRAY	'P','D','¬','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 258. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPD_VERSION_1	1	X'00000001'
MQPD_CURRENT_VERSION	1	X'00000001'

Nota: El símbolo ¬ representa un único carácter en blanco.

Opciones de descriptor de propiedad

Tabla 259. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPD_NONE	0	X'00000000'

Opciones de soporte de propiedad

Tabla 260. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPD_SUPPORT_OPTIONAL	1	X'00000001'
MQPD_SUPPORT_REQUIRED	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'

Contexto de propiedad

Tabla 261. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPD_NO_CONTEXT	0	X'00000000'
CONTEXTO_USUARIO_MQPDD	1	X'00000001'

MQPER_ * (Valores de persistencia)

Tabla 262. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPER_PERSISTENCE_AS_PARENT	-1	X'FFFFFFFF'
MQPER_NOT_PERSISTENT	0	X'00000000'
MQPER_PERSISTENT	1	X'00000001'
MQPER_PERSISTENCE_AS_Q_DEF	2	X'00000002'
MQPER_PERSISTENCE_AS_TOPIC_DEF	2	X'00000002'

MQPL_ * (Plataformas)

MQPL_MVS	1	X'00000001'
MQPL_OS390	1	X'00000001'
MQPL_ZOS	1	X'00000001'
MQPL_OS2	2	X'00000002'
MQPL_AIX	3	X'00000003'
MQPL_UNIX	3	X'00000003'
MQPL_OS400	4	X'00000004'
MQPL_WINDOWS	5	X'00000005'
MQPL_WINDOWS_NT	11	X'0000000B'

MQPL_VMS	12	X'0000000C'
MQPL_NSK	13	X'0000000D'
MQPL_NSS	13	X'0000000D'
MQPL_OPEN_TP1	15	X'0000000F'
MQPL_VM	18	X'00000012'
MQPL_TPF	23	X'00000017'
MQPL_VSE	27	X'0000001B'
MQPL_NATIVE	1	X'00000001'

MQPMO_ * (Colocar opciones de mensaje y estructura para máscara de publicación)

Estructura de opciones de colocación de mensaje

Tabla 263. Estructuras de constantes

Nombre	Estructura
MQPMO_STRUC_ID	"PMO¬"
MQPMO_STRUC_ID_ARRAY	'P','M','O','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 264. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPMO_VERSION_1	1	X'00000001'
MQPMO_VERSION_2	2	X'00000002'
MQPMO_VERSION_3	3	X'00000003'
MQPMO_CURRENT_VERSION	3	X'00000003'
MQPMO_LONGITUD_ACTUAL	(value differs by platform or version)	(value differs by platform or version)

Opciones de colocación de mensaje

Tabla 265. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPMO_SYNCPOINT	2	X'00000002'
MQPMO_NO_SYNCPOINT	4	X'00000004'
MQPMO_DEFAULT_CONTEXT	32	X'00000020'
MQPMO_NEW_MSG_ID	64	X'00000040'
MQPMO_NEW_CORREL_ID	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'

Tabla 265. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPMO_FAIL_IF_QUIESCING	8192	X'00002000'
MQPMO_NO_CONTEXT	16384	X'00004000'
MQPMO_LOGICAL_ORDER	32768	X'00008000'
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'

Opciones de colocación de mensaje para máscara de publicación

Tabla 266. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPMO_PUB_OPTIONS_MASK	2097152	X'00200000'

MQPMRF_ * (transferir campos de registro de mensajes)

Tabla 267. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPMRF_MSG_ID	1	X'00000001'
ID MQPMRF_CORREL_ID	2	X'00000002'
MQPMRF_ID_grupo	4	X'00000004'
MQPMRF_FEEDBACK	8	X'00000008'
MQPMRF_ACCOUNTING_TOKEN	16	X'00000010'
MQPMRF_NONE	0	X'00000000'

MQPO_ * (Formato de mandato, Opciones de depuración)

Tabla 268. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPO_SÍ	1	X'00000001'
MQPO_NO	0	X'00000000'

MQPRI_ * (Prioridad)

Tabla 269. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPRI_PRIORITY_AS_Q_DEF	-1	X'FFFFFFFF'
MQPRI_PRIORITY_AS_PARENT	-2	X'FFFFFFFE'
MQPRI_PRIORITY_AS_PUBLISHED	-3	X'FFFFFFFD'
MQPRI_PRIORITY_AS_TOPIC_DEF	-1	X'FFFFFFFF'

MQPROP_ * (Valores de control de propiedades de cola y canal y longitud máxima de propiedades)

Valores de control de propiedades de cola y canal

Tabla 270. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
COMPATIBILIDAD de MQPROP_COMPATIBILITY	0	X'00000000'
MQPROP_NONE	1	X'00000001'
MQPROP_ALL	2	X'00000002'
MQPROP_FORCE_MQRFH2	3	X'00000003'

Longitud máxima propiedades

Tabla 271. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPROP_UNRESTRICTED_LENGTH	-1	X'FFFFFFFF'

MQPRT_ * (Colocar valores de respuesta)

Tabla 272. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPRT_RESPONSE_AS_PARENT	0	X'00000000'
MQPRT_SYNC_RESPONSE	1	X'00000001'
MQPRT_ASYNC_RESPONSE	2	X'00000002'

MQPS_ * (Publicación/suscripción)

Formato de mandato Estado de publicación/suscripción

Tabla 273. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPS_STATUS_INACTIVE	0	X'00000000'
MQPS_STATUS_STARTING	1	X'00000001'
MQPS_STATUS_DETENIENDO	2	X'00000002'
MQPS_STATUS_ACTIVE	3	X'00000003'
MQPS_STATUS_COMPAT	4	X'00000004'
MQPS_STATUS_ERROR	5	X'00000005'

Tabla 273. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPS_STATUS_RECHAZADO	6	X'00000006'

Etiquetas de publicación/suscripción como series

Tabla 274. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPS_COMMAND	"MQPSCommand"	
Código_COMP_MQPS	"MQPSCompCode"	
ID MQPS_CORREL_ID	"MQPSCorrelId"	
MQPS_DELETE_OPCIONES	"MQPSDelOpts"	
ID_ERROR_MQPS	"MQPSErrorId"	
MQPS_ERROR_POS	"MQPSErrorPos"	
MQPS_INTEGER_DATA	"MQPSIntData"	
MQPS_PARAMETER_ID	"MQPSParmId"	
Opciones de MQPS_PUBLICATION_OPTIONS	"MQPSPubOpts"	
MQPS_PUBLISH_TIMESTAMP	"MQPSPubTime"	
MQPS_Q_MGR_NAME	"MQPSQMgrName"	
MQPS_Q_NAME	"MQPSQName"	
RAZÓN_MQPS	"MQPSReason"	
TEXTO_MOTIVO_MQPS	"MQPSReasonText"	
MQPS_REGISTRATION_OPTIONS	"MQPSRegOpts"	
NÚMERO_SECUENCIA_MQPS	"MQPSSeqNum"	
MQPS_STREAM_NAME	"MQPSStreamName"	
MQPS_STRING_DATA	"MQPSStringData"	
MQPS_SUBSCRIPTION_IDENTITY	"MQPSSubIdentity"	
MQPS_SUBSCRIPTION_NAME	"MQPSSubName"	
MQPS_SUBSCRIPTION_USER_DATA	"MQPSSubUserData"	
MQPS_TOPIC	"MQPSTopic"	
ID_USUAR_MQPS	"MQPSUserId"	

Etiquetas de publicación/suscripción como series encerradas en blanco

Tabla 275. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPS_COMMAND_B	"bMQPSCommandb"	
MQPS_COMP_CODE_B	"bMQPSCompCodeb"	
MQPS_ID_CORREL_B	"bMQPSCorrelIdb"	
MQPS_DELETE_OPTIONS_B	"bMQPSDelOptsb"	
ID_ERROR_MQPS_B	"bMQPSErrorIdb"	
MQPS_ERROR_POS_B	"bMQPSErrorPosb"	
MQPS_INTEGER_DATA_B	"bMQPSIntDatab"	

Tabla 275. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPS_ID_PARÁMETRO_B	"bMQPSParmIdb"	
MQPS_PUBCIONTION_OPTIONS_B	"bMQPSPubOptsb"	
MQPS_PUBLISH_TIMESTAMP_B	"bMQPSPubTimeb"	
MQPS_Q_MGR_NAME_B	"bMQPSQMgrNameb"	
MQPS_Q_NAME_B	"bMQPSQNameb"	
MQPS_MOTIVO_B	"bMQPSReasonb"	
MQPS_MOTIVO_TEXTO_B	"bMQPSReasonTextb"	
MQPS_REGISTRATION_OPTIONS_B	"bMQPSRegOptsb"	
MQPS_NÚMERO_SECUENCIA_B	"bMQPSSeqNumb"	
MQPS_NOMBRE_SECUENCIA_B	"bMQPSStreamNameb"	
MQPS_STRING_DATA_B	"bMQPSStringDatab"	
MQPS_SUBSCRIPTION_IDENTITY_B	"bMQPSSubIdentityb"	
MQPS_SUBSCRIPTION_NAME_B	"bMQPSSubNameb"	
MQPS_SUBSCRIPTION_USER_DATA_B	"bMQPSSubUserDatab"	
MQPS_TOPIC_B	"bMQPSTopicb"	
MQPS_ID_USER_B	"bMQPSUserIdb"	

Valores de etiqueta de mandato de publicación/suscripción como series

Tabla 276. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPS_DELETE_PUBLICATION	"DeletePub"	
MQPS_DEREGISTER_PUBLISHER	"DeregPub"	
MQPS_DEREGISTER_SUBSCRIBER	"DeregSub"	
MQPS_PUBLISH	"Publish"	
MQPS_REGISTER_PUBLISHER	"RegPub"	
MQPS_REGISTER_SUBSCRIBER	"RegSub"	
MQPS_REQUEST_UPDATE	"ReqUpdate"	

Valores de etiqueta de mandato de publicación/suscripción como series entre espacios en blanco

Tabla 277. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPS_DELETE_PUBCIONTION_B	"bDeletePubb"	
MQPS_DEREGISTER_PUBLISHER_B	"bDeregPubb"	
MQPS_DEREGISTER_SUBSCRIBER_B	"bDeregSubb"	
MQPS_PUBLISH_B	"bPublishb"	
MQPS_REGISTER_PUBLISHER_B	"bRegPubb"	
MQPS_REGISTER_SUBSCRIBER_B	"bRegSubb"	
MQPS_REQUEST_UPDATE_B	"bReqUpdateb"	

Valores de etiqueta de opciones de publicación/suscripción como series

Tabla 278. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPS_ADD_NAME	"AddName"	
MQPS_ANÓNIMO	"Anon"	
MQPS_CORREL_ID_AS_IDENTITY	"CorrelAsId"	
MQPS_DEREGISTER_ALL	"DeregAll"	
MQPS_DIRECT_REQUESTS	"DirectReq"	
MQPS_DUPLICATES_OK	"DupsOK"	
MQPS_FULL_RESPONSE	"FullResp"	
MQPS_INCLUDE_STREAM_NAME	"InclStreamName"	
MQPS_INFORM_IF_RETAINED	"InformIfRet"	
MQPS_IS_RETAINED_PUBLICATION	"IsRetainedPub"	
MQPS_JOIN_EXCLUSIVE	"JoinExcl"	
MQPS_JOIN_SHARED	"JoinShared"	
MQPS_LEAVE_ONLY	"LeaveOnly"	
MQPS_LOCAL	"Local"	
MQPS_LOCKED	"Locked"	
MQPS_NEW_PUBLICATIONS_ONLY	"NewPubsOnly"	
MQPS_NO_ALTERACIÓN	"NoAlter"	
MQPS_NO_REGISTRO	"NoReg"	
MQPS_NON_PERSISTENT	"NonPers"	
MQPS_NONE	"None"	
MQPS_OTHER_SUBSCRIBERS_ONLY	"OtherSubsOnly"	
MQPS_PERSISTENT	"Pers"	
MQPS_PERSISTENT_AS_PUBLISH	"PersAsPub"	
MQPS_PERSISTENT_AS_Q	"PersAsQueue"	
MQPS_PUBLISH_ON_REQUEST_ONLY	"PubOnReqOnly"	
MQPS_PUBLICACIÓN_RETENCIÓN	"RetainPub"	
MQPS_VARIABLE_USER_ID	"VariableUserId"	

Valores de etiqueta de opciones de publicación/suscripción como series encerradas entre blancos

Tabla 279. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQPS_ADD_NAME_B	"bAddNameb"	
MQPS_ANÓNIMO_B	"bAnonb"	
MQPS_CORREL_ID_AS_IDENTITY_B	"bCorrelAsIdb"	
MQPS_DEREGISTER_ALL_B	"bDeregAllb"	
MQPS_DIRECT_REQUESTS_B	"bDirectReqb"	
MQPS_DUPLICATES_OK_B	"bDupsOKb"	

Tabla 279. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPS_FULL_RESPONSE_B	"bFullRespb"	
MQPS_INCLUDE_STREAM_NAME_B	"bInclStreamNameb"	
MQPS_INFORM_IF_RETAINED_B	"bInformIfRetb"	
MQPS_IS_RETAINED_PUBCIONTION_B	"bIsRetainedPubb"	
MQPS_JOIN_EXCLUSIVE_B	"bJoinExclb"	
MQPS_JOIN_SHARED_B	"bJoinSharedb"	
MQPS_LEAVE_ONLY_B	"bLeaveOnlyb"	
MQPS_LOCAL_B	"bLocalb"	
MQPS_LOCKED_B	"bLockedb"	
MQPS_NEW_PUBLICATIONS_ONLY_B	"bNewPubsOnlyb"	
MQPS_NO_ALTERACIÓN_B	"bNoAlterb"	
MQPS_NO_REGISTRATION_B	"bNoRegb"	
MQPS_NON_PERSISTENT_B	"bNonPersb"	
MQPS_NONE_B	"bNoneb"	
MQPS_OTHER_SUBSCRIBERS_ONLY_B	"bOtherSubsOnlyb"	
MQPS_PERSISTENT_B	"bPersb"	
MQPS_PERSISTENT_AS_PUBLISH_B	"bPersAsPubb"	
MQPS_PERSISTENT_AS_Q_B	"bPersAsQueueb"	
MQPS_PUBLISH_ON_REQUEST_ONLY_B	"bPubOnReqOnlyb"	
MQPS_RETAIN_PUBCIONTION_B	"bRetainPubb"	
MQPS_VARIABLE_USER_ID_B	"bVariableUserIdb"	

MQPSC_ * (Carpeta de mandatos de publicación/suscripción de código de opciones de publicación/suscripción (psc) Etiquetas)

Tabla 280. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPSC_FOLDER_VERSION	1	X'00000001'

MQPSC_ * (Nombres de etiqueta de opciones de publicación/suscripción)

Tabla 281. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPSC_COMMAND	"Command"	
MQPSC_REGISTRATION_OPTION	"RegOpt"	
MQPSC_PUBCIONTION_OPTION	"PubOpt"	
MQPSC_DELETE_OPTION	"DelOpt"	
MQPSC_TOPIC	"Topic"	
MQPSC_SUBSCRIPTION_POINT	"SubPoint"	
FILTRO MQPSC	"Filter"	
MQPSC_Q_MGR_NAME	"QMgrName"	
MQPSC_Q_NAME	"QName"	

Tabla 281. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPSC_PUBLISH_TIMESTAMP	"PubTime"	
NÚMERO_SECUENCIA_MQPSC_NUMBER	"SeqNum"	
MQPSC_SUBSCRIPTION_NAME	"SubName"	
MQPSC_SUSCRIPTION_IDENTITY	"SubIdentity"	
MQPSC_SUBSCRIPTION_USER_DATA	"SubUserData"	
MQPSC_CORREL_ID	"CorrelId"	

MQPSC_ * (Nombres de etiqueta XML de opciones de publicación/ suscripción)

Tabla 282. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPSC_COMMAND_B	"<Command>"	
MQPSC_COMMAND_E	"</Command>"	
MQPSC_REGISTRATION_OPTION_B	"<RegOpt>"	
MQPSC_REGISTRATION_OPTION_E	"</RegOpt>"	
MQPSC_PUBCIONTION_OPTION_B	"<PubOpt>"	
MQPSC_PUBCIONTION_OPTION_E	"</PubOpt>"	
MQPSC_DELETE_OPTION_B	"<DelOpt>"	
MQPSC_DELETE_OPTION_E	"</DelOpt>"	
MQPSC_TOPIC_B	"<Topic>"	
MQPSC_TOPIC_E	"</Topic>"	
MQPSC_SUBSCRIPTION_POINT_B	"<SubPoint>"	
MQPSC_SUBSCRIPTION_POINT_E	"</SubPoint>"	
FILTER_MQPSC_B	"<Filter>"	
FILTER_MQPS_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_NÚMERO_SECUENCIA_B	"<SeqNum>"	
MQPSC_SEQUENCE_NUMBER_E	"</SeqNum>"	
MQPSC_SUBSCRIPTION_NAME_B	"<SubName>"	
MQPSC_SUBSCRIPTION_NAME_E	"</SubName>"	
MQPSC_SUBSCRIPTION_IDENTITY_B	"<SubIdentity>"	
MQPSC_SUBSCRIPTION_IDENTITY_E	"</SubIdentity>"	
MQPSC_SUBSCRIPTION_USER_DATA_B	"<SubUserData>"	
MQPSC_SUBSCRIPTION_USER_DATA_E	"</SubUserData>"	
MQPSC_CORREL_ID_B	"<CorrelId>"	

Tabla 282. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPSC_CORREL_ID_E	"</CorrelId>"	

MQPSC_ * (Valores de etiqueta de opciones de publicación/suscripción como series)

Tabla 283. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPSC_DELETE_PUBCIONCIÓN	"DeletePub"	
MQPSC_DEREGISTER_SUBSCRIBER	"DeregSub"	
MQPSC_PUBLICAR	"Publish"	
MQPSC_REGISTER_SUSCRIPTOR	"RegSub"	
MQPSC_REQUEST_UPDATE	"ReqUpdate"	

MQPSC_ * (Valores de etiqueta de opciones de publicación/suscripción como series)

Tabla 284. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPSC_ADD_NAME	"AddName"	
MQPSC_CORREL_ID_AS_IDENTITY	"CorrelAsId"	
MQPSC_DEREGISTER_ALL	"DeregAll"	
MQPSC_DUPLICATES_Correcto	"DupsOK"	
MQPSC_FULL_RESPONSE	"FullResp"	
MQPSC_INFORM_IF_RETAINED	"InformIfRet"	
MQPSC_IS_RETAINED_PUB	"IsRetainedPub"	
MQPSC_JOIN_SHARED	"JoinShared"	
MQPSC_JOIN_EXCLUSIVE	"JoinExcl"	
MQPSC_LEAVE_ONLY	"LeaveOnly"	
MQPSC_LOCAL	"Local"	
MQPSC_LOCKED	"Locked"	
MQPSC_NEW_PUBS_ONLY	"NewPubsOnly"	
MQPSC_NO_ALTERATION	"NoAlter"	
MQPSC_NO persistente	"NonPers"	
MQPSC_OTHER_SUBS_ONLY	"OtherSubsOnly"	
MQPSC_PERSISTENT	"Pers"	
MQPSC_PERSISTENT_AS_PUBLISH	"PersAsPub"	
MQPSC_PERSISTENT_AS_Q	"PersAsQueue"	
MQPSC_NONE	"None"	
MQPSC_PUB_ON_REQUEST_ONLY	"PubOnReqOnly"	
MQPSC_RETAIN_PUB	"RetainPub"	
MQPSC_VARIABLE_USER_ID	"VariableUserId"	

MQPSCR_ * (Opciones de publicación/suscripción)

Opciones de publicación/suscripción Etiquetas de carpeta de respuesta de publicación/suscripción (pscr)

Tabla 285. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
VERSIÓN_CARPETA_MQPSCRP	1	X'00000001'

Nombres de etiqueta de opciones de publicación/suscripción

Tabla 286. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
COMPLETACIÓN_MQPSCR_COMPLETAR	"Completion"	
MQPSCR_RESPONSE	"Response"	
RAZÓN_MQPSCR_RAZÓN	"Reason"	

Nombres de etiqueta XML de etiquetas de opciones de publicación/suscripción

Tabla 287. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPSCR_COMPLETION_B	"<Completion>"	
MQPSCR_COMPLETION_E	"</Completion>"	
MQPSCR_RESPONSE_B	"<Response>"	
MQPSCR_RESPONSE_E	"</Response>"	
MQPSCR_REASON_B	"<Reason>"	
MQPSCR_REASON_E	"</Reason>"	

Valores de etiqueta de opciones de publicación/suscripción

Tabla 288. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPSCR_Correcto	"ok"	
MQPSCR_AVISO	"warning"	
ERROR DE MQPSRC	"error"	

MQPSM_ * (modalidad Pub/Sub)

Tabla 289. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPSM_INHABILITADO	0	X'00000000'
MQPSM_COMPAT	1	X'00000001'
MQPSM_HABILITADO	2	X'00000002'

MQPSPROP_ * (Propiedades de mensaje de publicación/suscripción)

Tabla 290. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPSPROP_NONE	0	X'00000000'
MQPSPROP_COMPAT	1	X'00000001'
MQPSPROP_RFH2	2	X'00000002'
MQPSPROP_MSGPROP	3	X'00000003'

MQPSST_ * (Formato de mandato, Tipo de estado de publicación/suscripción)

Tabla 291. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPSST_ALL	0	X'00000000'
MQPSST_LOCAL	1	X'00000001'
MQPSST_PARENT	2	X'00000002'
MQPSST_HIJO	3	X'00000003'

MQPUBO_ * (Opciones de publicación/suscripción)

Tabla 292. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPUBO_NONE	0	X'00000000'
MQPUBO_CORREL_ID_AS_IDENTITY	1	X'00000001'
MQPUBO_RETAIN_PUBLICACIÓN	2	X'00000002'
MQPUBO_OTHER_SUBSCRIBERS_ONLY	4	X'00000004'
MQPUBO_NO_REGISTRO	8	X'00000008'
MQPUBO_IS_RETAINED_PUBLICATION	16	X'00000010'

MQPXP_ * (Estructura de parámetros de salida de direccionamiento de publicación/suscripción)

Tabla 293. Estructuras de constantes	
Nombre	**Estructura**
MQPXP_STRUC_ID	"PXP¬"
MQPXP_STRUC_ID_ARRAY	'P','X','P','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 294. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQPXP_VERSION_1	1	X'00000001'
MQPXP_CURRENT_VERSION	1	X'00000001'

MQQA_ * (Atributos de cola)

Inhibir obtención de valores

Tabla 295. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQA_GET_INHIBITED	1	X'00000001'
MQQA_GET_ALLOWED	0	X'00000000'

Inhibir valores de colocación

Tabla 296. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQA_PUT_INHIBITED	1	X'00000001'
MQQA_PUT_ALLOWED	0	X'00000000'

Compartición de cola

Tabla 297. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQA_SHAREABLE	1	X'00000001'
MQQA_NOT_SHAREABLE	0	X'00000000'

Refuerzo de retrotracción

Tabla 298. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQA_BACKOUT_HARDENED	1	X'00000001'
MQQA_BACKOUT_NOT_HARTIZADO	0	X'00000000'

MQQDT_ * (Tipos de definición de cola)

Tabla 299. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQDT_PREDEFINED	1	X'00000001'
MQQDT_PERMANENT_DYNAMIC	2	X'00000002'
MQQDT_TEMPORARY_DYNAMIC	3	X'00000003'
MQQDT_SHARED_DYNAMIC	4	X'00000004'

MQQF_ * (distintivos de cola)

Tabla 300. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQF_LOCAL_Q	1	X'00000001'
MQQF_CLWL_USEQ_ANY	64	X'00000040'
MQQF_CLWL_USEQ_LOCAL	128	X'00000080'

MQQMDT_ * (Formato de mandato, Tipos de definición de gestor de colas)

Tabla 301. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQMDT_EXPLICIT_CLUSTER_SENDER	1	X'00000001'
MQQMDT_AUTO_CLUSTER_SENDER	2	X'00000002'
MQQMDT_AUTO_EXP_CLUSTER_SENDER	4	X'00000004'
MQQMDT_CLUSTER_RECEIVER	3	X'00000003'

MQQMF_ * (distintivos de gestor de colas)

Tabla 302. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
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 de mandato, Recurso de gestor de colas)

Tabla 303. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQMFAC_IMS_BRIDGE	1	X'00000001'
MQQMFAC_DB2	2	X'00000002'

MQQMSTA_ * (Formato de mandato, Estado del gestor de colas)

Tabla 304. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQMSTA_INICIANDO	1	X'00000001'
MQQMSTA_RUNNING	2	X'00000002'
MQQMSTA_QUIESCING	3	X'00000003'

MQQMT_ * (Formato de mandato, Tipos de gestor de colas)

Tabla 305. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQMT_NORMAL	0	X'00000000'
REPOSITORIO_MQQM	1	X'00000001'

MQQO_ * (Formato de mandato, Opciones de inmovilización)

Tabla 306. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQO_SÍ	1	X'00000001'
MQQO_NO	0	X'00000000'

MQQSGD_ * (Disposiciones de grupo de compartición de colas)

Tabla 307. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQSGD_ALL	-1	X'FFFFFFFF'
MQQSGD_Q_MGR	0	X'00000000'
MQQSGD_COPY	1	X'00000001'
MQQSGD_SHARED	2	X'00000002'
MQQSGD_XX_ENCODE_CASE_ONE grupo	3	X'00000003'
MQQSGD_PRIVADO	4	X'00000004'
MQQSGD_LIVE	6	X'00000006'

MQQSGS_ * (Formato de mandato, Estado de QSG)

Tabla 308. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQSGS_DESCONOCIDO	0	X'00000000'
MQQSGS_CREADO	1	X'00000001'
MQQSGS_ACTIVE	2	X'00000002'
MQQSGS_INACTIVE	3	X'00000003'
MQQSGS_FAILED	4	X'00000004'
MQQSGS_PENDIENTE	5	X'00000005'

MQQSIE_ * (Formato de mandato, Servicio de cola-Sucesos de intervalo)

Tabla 309. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQSIE_NONE	0	X'00000000'
MQQSIE_HIGH	1	X'00000001'
MQQSIE_OK	2	X'00000002'

MQQSO_ * (Formato de mandato, Opciones de apertura de estado de cola para SET, BROWSE, INPUT)

Tabla 310. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQSO_NO	0	X'00000000'
MQQSO_SÍ	1	X'00000001'
MQQSO_COMPARTIDO	1	X'00000001'
MQQSO_EXCLUSIVE	2	X'00000002'

MQQSOT_ * (Formato de mandato, Tipos de apertura de estado de cola)

Tabla 311. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQQSOT_ALL	1	X'00000001'

Tabla 311. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQSOT_INPUT	2	X'00000002'
MQQSOT_OUTPUT	3	X'00000003'

MQQSUM_ * (Formato de mandato, Estado de cola, Mensajes no confirmados)

Tabla 312. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQSUM_SÍ	1	X'00000001'
MQQSUM_NO	0	X'00000000'

MQQT_ * (Tipos de cola y tipos de cola ampliados)

Tipos de cola

Tabla 313. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQT_LOCAL	1	X'00000001'
MQQT_MODEL	2	X'00000002'
MQQT_ALIAS	3	X'00000003'
MQQT_REMOTE	6	X'00000006'
MQQT_CLUSTER	7	X'00000007'

Tipos de cola ampliada

Tabla 314. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQQT_ALL	1001	X'000003E9'

MQRC_ * (Códigos de razón)

Tabla 315. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_NONE	0	X'00000000'
MQRC_APPL_PRIMERO	900	X'00000384'
MQRC_APPL_LAST	999	X'000003E7'
MQRC_ALIAS_BASE_Q_TYPE_ERROR	2001	X'000007D1'
MQRC_ALREADY_CONNECTED	2002	X'000007D2'
MQRC_BACKED_OUT	2003	X'000007D3'
MQRC_BUFFER_ERROR	2004	X'000007D4'
MQRC_BUFFER_LENGTH_ERROR	2005	X'000007D5'
MQRC_CHAR_ATTR_LENGTH_ERROR	2006	X'000007D6'
MQRC_CHAR_ATTRS_ERROR	2007	X'000007D7'
MQRC_CHAR_ATTRS_TOO_SHORT	2008	X'000007D8'
MQRC_CONNECTION_BROKEN	2009	X'000007D9'

Tabla 315. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRC_DATA_LENGTH_ERROR	2010	X'000007DA'
MQRC_DYNAMIC_Q_NAME_ERROR	2011	X'000007DB'
ERROR_ENTORNO_MQRC	2012	X'000007DC'
MQRC_EXPIRY_ERROR	2013	X'000007DD'
MQRC_FEEDBACK_ERROR	2014	X'000007DE'
MQRC_GET_INHIBITED	2016	X'000007E0'
MQRC_HANDLE_NOT_AVAILABLE	2017	X'000007E1'
MQRC_HCONN_ERROR	2018	X'000007E2'
MQRC_HOBJ_ERROR	2019	X'000007E3'
MQRC_INHIBIDOR_VALOR_ERROR	2020	X'000007E4'
MQRC_INT_ATTR_COUNT_ERROR	2021	X'000007E5'
MQRC_INT_ATTR_COUNT_TOO_SMALL	2022	X'000007E6'
MQRC_INT_ATTRS_ARRAY_ERROR	2023	X'000007E7'
MQRC_SYNCPOINT_LIMIT_REACHED	2024	X'000007E8'
MQRC_MAX_CONNS_LIMIT_ALCANZADO	2025	X'000007E9'
MQRC_MD_ERROR	2026	X'000007EA'
MQRC_MISSING_REPLY_TO_Q	2027	X'000007EB'
MQRC_MSG_TYPE_ERROR	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'
MQRC_OBJECT_TYPE_ERROR	2043	X'000007FB'
MQRC_OD_ERROR	2044	X'000007FC'
MQRC_OPTION_NOT_VALID_FOR_TYPE	2045	X'000007FD'
MQRC_OPTIONS_ERROR	2046	X'000007FE'
MQRC_PERSISTENCE_ERROR	2047	X'000007FF'
MQRC_PERSISTENT_NOT_ALLOWED	2048	X'00000800'
MQRC_PRIORITY_EXCEEDS_MAXIMUM	2049	X'00000801'
MQRC_PRIORITY_ERROR	2050	X'00000802'
MQRC_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida	2051	X'00000803'

Tabla 315. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRC_Q_DELETED	2052	X'00000804'
MQRC_Q_FULL	2053	X'00000805'
MQRC_Q_NOT_EMPTY	2055	X'00000807'
MQRC_Q_SPACE_NOT_AVAILABLE	2056	X'00000808'
MQRC_Q_TYPE_ERROR	2057	X'00000809'
MQRC_Q_MGR_NAME_ERROR	2058	X'0000080A'
MQRC_Q_MGR_NOT_AVAILABLE	2059	X'0000080B'
MQRC_REPORT_OPTIONS_ERROR	2061	X'0000080D'
MQRC_SECOND_MARK_NOT_ALLOWED	2062	X'0000080E'
MQRC_SECURITY_ERROR	2063	X'0000080F'
MQRC_SELECTOR_COUNT_ERROR	2065	X'00000811'
MQRC_SELECTOR_LIMIT_EXCEEDED	2066	X'00000812'
MQRC_SELECTOR_ERROR	2067	X'00000813'
MQRC_SELECTOR_NOT_FOR_TYPE	2068	X'00000814'
MQRC_SIGNAL_OUTSTANDING	2069	X'00000815'
MQRC_SIGNAL_REQUEST_ACCEPTED	2070	X'00000816'
MQRC_STORAGE_NOT_AVAILABLE	2071	X'00000817'
MQRC_SYNCPOINT_NOT_AVAILABLE	2072	X'00000818'
MQRC_TRIGGER_CONTROL_ERROR	2075	X'0000081B'
MQRC_TRIGGER_DEPTH_ERROR	2076	X'0000081C'
MQRC_TRIGGER_MSG_PRIORITY_ERR	2077	X'0000081D'
MQRC_TRIGGER_TYPE_ERROR	2078	X'0000081E'
MQRC_TRUNCATED_MSG_ACCEPTED	2079	X'0000081F'
MQRC_TRUNCATED_MSG_FAILED	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'
MQRC_WAIT_INTERVAL_ERROR	2090	X'0000082A'
MQRC_XMIT_Q_TYPE_ERROR	2091	X'0000082B'
MQRC_XMIT_Q_USAGE_ERROR	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'
MQRC_CONTEXT_HANDLE_ERROR	2097	X'00000831'
MQRC_CONTEXT_NOT_AVAILABLE	2098	X'00000832'
MQRC_SIGNAL1_ERROR	2099	X'00000833'
MQRC_OBJECT_ALREADY_EXISTS	2100	X'00000834'

Tabla 315. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRC_OBJECT_DAMAGED	2101	X'00000835'
MQRC_RESOURCE_PROBLEM	2102	X'00000836'
MQRC_ANOTHER_Q_MGR_CONNECTED	2103	X'00000837'
MQRC_UNKNOWN_REPORT_OPTION	2104	X'00000838'
MQRC_STORAGE_CLASS_ERROR	2105	X'00000839'
MQRC_COD_NOT_VALID_FOR_XCF_Q	2106	X'0000083A'
MQRC_XWAIT_CANCELADO	2107	X'0000083B'
MQRC_XWAIT_ERROR	2108	X'0000083C'
MQRC_SUPPRESSED_BY_EXIT	2109	X'0000083D'
MQRC_FORMAT_ERROR	2110	X'0000083E'
MQRC_SOURCE_CCSID_ERROR	2111	X'0000083F'
MQRC_SOURCE_INTEGER_ENC_ERROR	2112	X'00000840'
MQRC_SOURCE_DECIMAL_ENC_ERROR	2113	X'00000841'
MQRC_SOURCE_FLOAT_ENC_ERROR	2114	X'00000842'
MQRC_TARGET_CCSID_ERROR	2115	X'00000843'
MQRC_TARGET_INTEGER_ENC_ERROR	2116	X'00000844'
MQRC_TARGET_DECIMAL_ENC_ERROR	2117	X'00000845'
MQRC_TARGET_FLOAT_ENC_ERROR	2118	X'00000846'
MQRC_NOT_CONVERTED	2119	X'00000847'
MQRC_CONVERTED_MSG_TOO_BIG	2120	X'00000848'
MQRC_TRUNCADO	2120	X'00000848'
MQRC_NO_EXTERNAL_PARTICIPANTES	2121	X'00000849'
MQRC_PARTICIPANT_NOT_AVAILABLE	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'
MQRC_ADAPTER_STORAGE_INSUFICIENTE	2127	X'0000084F'
MQRC_UOW_IN_PROGRESS	2128	X'00000850'
MQRC_ADAPTER_CONN_LOAD_ERROR	2129	X'00000851'
MQRC_ADAPTER_SERV_LOAD_ERROR	2130	X'00000852'
MQRC_ADAPTER_DEFS_ERROR	2131	X'00000853'
MQRC_ADAPTER_DEFS_LOAD_ERROR	2132	X'00000854'
nMQRC_ADAPTER_CONV_LOAD_ERROR	2133	X'00000855'
MQRC_BO_ERROR	2134	X'00000856'
MQRC_DH_ERROR	2135	X'00000857'
MQRC_MULTIPLE_RAZONES	2136	X'00000858'
MQRC_OPEN_FAILED	2137	X'00000859'
MQRC_ADAPTER_DISC_LOAD_ERROR	2138	X'0000085A'

Tabla 315. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRC_CNO_ERROR	2139	X'0000085B'
MQRC_CICS_WAIT_FAILED	2140	X'0000085C'
MQRC_DLH_ERROR	2141	X'0000085D'
MQRC_HEADER_ERROR	2142	X'0000085E'
MQRC_SOURCE_LENGTH_ERROR	2143	X'0000085F'
MQRC_TARGET_LENGTH_ERROR	2144	X'00000860'
MQRC_SOURCE_BUFFER_ERROR	2145	X'00000861'
MQRC_TARGET_BUFFER_ERROR	2146	X'00000862'
MQRC_IIH_ERROR	2148	X'00000864'
MQRC_PCF_ERROR	2149	X'00000865'
MQRC_DBCS_ERROR	2150	X'00000866'
MQRC_OBJECT_NAME_ERROR	2152	X'00000868'
MQRC_OBJECT_Q_MGR_NAME_ERROR	2153	X'00000869'
MQRC_RECS_PRESENT_ERROR	2154	X'0000086A'
MQRC_OBJECT_RECORDS_ERROR	2155	X'0000086B'
MQRC_RESPONSE_RECORDS_ERROR	2156	X'0000086C'
MQRC_ASID_MISMATCH	2157	X'0000086D'
MQRC_PMO_RECORD_FLAGS_ERROR	2158	X'0000086E'
MQRC_PUT_MSG_RECORDS_ERROR	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'
MQRC_PMO_ERROR	2173	X'0000087D'
MQRC_API_EXIT_NOT_FOUND	2182	X'00000886'
MQRC_API_EXIT_LOAD_ERROR	2183	X'00000887'
MQRC_REMOTE_Q_NAME_ERROR	2184	X'00000888'
MQRC_INCONSISTENT_PERSISTENCE	2185	X'00000889'
MQRC_GMO_ERROR	2186	X'0000088A'
MQRC_CICS_BRIDGE_RESTRICTION	2187	X'0000088B'
MQRC_STOPPED_BY_CLUSTER_EXIT	2188	X'0000088C'
MQRC_CLUSTER_RESOLUTION_ERROR	2189	X'0000088D'
MQRC_CONVERTED_STRING_TOO_BIG	2190	X'0000088E'
ERROR MQRC_TMC	2191	X'0000088F'
MQRC_PAGESET_FULL	2192	X'00000890'
MQRC_STORAGE_MEDIUM_FULL	2192	X'00000890'
MQRC_PAGESET_ERROR	2193	X'00000891'
MQRC_NAME_NOT_VALID_FOR_TYPE	2194	X'00000892'
MQRC_UNEXPECTED_ERROR	2195	X'00000893'

Tabla 315. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRC_UNKNOWN_XMIT_Q	2196	X'00000894'
MQRC_UNKNOWN_DEF_XMIT_Q	2197	X'00000895'
MQRC_DEF_XMIT_Q_TYPE_ERROR	2198	X'00000896'
MQRC_DEF_XMIT_Q_USAGE_ERROR	2199	X'00000897'
MQRC_MSG_MARKED_BROWSE_CO_OP	2200	X'00000898'
MQRC_NAME_IN_USE	2201	X'00000899'
MQRC_CONNECTION_QUIESCING	2202	X'0000089A'
MQRC_CONNECTION_STOPPING	2203	X'0000089B'
MQRC_ADAPTER_NOT_AVAILABLE	2204	X'0000089C'
MQRC_MSG_ID_ERROR	2206	X'0000089E'
MQRC_CORREL_ID_ERROR	2207	X'0000089F'
MQRC_FILE_SYSTEM_ERROR	2208	X'000008A0'
MQRC_NO_MSG_LOCKED	2209	X'000008A1'
MQRC_SOAP_DOTNET_ERROR	2210	X'000008A2'
MQRC_SOAP_AXIS_ERROR	2211	X'000008A3'
MQRC_SOAP_URL_ERROR	2212	X'000008A4'
MQRC_FILE_NOT_AUDITED	2216	X'000008A8'
MQRC_CONNECTION_NOT_AUTHORIZED	2217	X'000008A9'
MQRC_MSG_TOO_BIG_FOR_CHANNEL	2218	X'000008AA'
MQRC_CALL_IN_PROGRESS	2219	X'000008AB'
MQRC_RMH_ERROR	2220	X'000008AC'
MQRC_Q_MGR_ACTIVE	2222	X'000008AE'
MQRC_Q_MGR_NOT_ACTIVE	2223	X'000008AF'
MQRC_Q_DEPTH_HIGH	2224	X'000008B0'
MQRC_Q_DEPTH_LOW	2225	X'000008B1'
MQRC_Q_SERVICE_INTERVAL_HIGH	2226	X'000008B2'
MQRC_Q_SERVICE_INTERVAL_OK	2227	X'000008B3'
MQRC_RFH_HEADER_FIELD_ERROR	2228	X'000008B4'
MQRC_RAS_PROPERTY_ERROR	2229	X'000008B5'
MQRC_UNIT_OF_WORK_NOT_STARTED	2232	X'000008B8'
MQRC_CHANNEL_AUTO_DEF_OK	2233	X'000008B9'
MQRC_CHANNEL_AUTO_DEF_ERROR	2234	X'000008BA'
MQRC_CFH_ERROR	2235	X'000008BB'
MQRC_CFIL_ERROR	2236	X'000008BC'
MQRC_CFIN_ERROR	2237	X'000008BD'
MQRC_CFSL_ERROR	2238	X'000008BE'
MQRC_CFST_ERROR	2239	X'000008BF'
MQRC_INCOMPLETE_GROUP	2241	X'000008C1'
MQRC_INCOMPLETE_MSG	2242	X'000008C2'

Tabla 315. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRC_INCONSISTENT_CCSIDS	2243	X'000008C3'
MQRC_INCONSISTENT_ENCODINGS	2244	X'000008C4'
MQRC_INCONSISTENT_UOW	2245	X'000008C5'
MQRC_INVALID_MSG_UNDER_CURSOR	2246	X'000008C6'
MQRC_MATCH_OPTIONS_ERROR	2247	X'000008C7'
MQRC_MDE_ERROR	2248	X'000008C8'
MQRC_MSG_FLAGS_ERROR	2249	X'000008C9'
MQRC_MSG_SEQ_NUMBER_ERROR	2250	X'000008CA'
ERROR DE MQRC_OFFSET_	2251	X'000008CB'
MQRC_ORIGINAL_LENGTH_ERROR	2252	X'000008CC'
MQRC_SEGMENT_LENGTH_ZERO	2253	X'000008CD'
MQRC_UOW_NOT_AVAILABLE	2255	X'000008CF'
MQRC_WRONG_GMO_VERSION	2256	X'000008D0'
MQRC_WRONG_MD_VERSION	2257	X'000008D1'
MQRC_GROUP_ID_ERROR	2258	X'000008D2'
MQRC_INCONSISTENT_BROWSE	2259	X'000008D3'
MQRC_XQH_ERROR	2260	X'000008D4'
MQRC_SRC_ENV_ERROR	2261	X'000008D5'
MQRC_SRC_NAME_ERROR	2262	X'000008D6'
MQRC_DEST_ENV_ERROR	2263	X'000008D7'
MQRC_DEST_NAME_ERROR	2264	X'000008D8'
ERROR MQRC_TM	2265	X'000008D9'
MQRC_CLUSTER_EXIT_ERROR	2266	X'000008DA'
MQRC_CLUSTER_EXIT_LOAD_ERROR	2267	X'000008DB'
MQRC_CLUSTER_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida	2268	X'000008DC'
MQRC_CLUSTER_RESOURCE_ERROR	2269	X'000008DD'
MQRC_NO_DESTINATIONS_AVAILABLE	2270	X'000008DE'
MQRC_CONN_TAG_IN_USE	2271	X'000008DF'
MQRC_PARTIALLY_CONVERTED	2272	X'000008E0'
MQRC_CONNECTION_ERROR	2273	X'000008E1'
MQRC_OPTION_ENVIRONMENT_ERROR	2274	X'000008E2'
MQRC_CD_ERROR	2277	X'000008E5'
MQRC_CLIENT_CONN_ERROR	2278	X'000008E6'
MQRC_CHANNEL_STOPPED_BY_USER	2279	X'000008E7'
MQRC_HCONFIG_ERROR	2280	X'000008E8'
MQRC_FUNCTION_ERROR	2281	X'000008E9'
MQRC_CHANNEL_STARTED	2282	X'000008EA'
MQRC_CHANNEL_STOPPED	2283	X'000008EB'
MQRC_CHANNEL_CONV_ERROR	2284	X'000008EC'

Tabla 315. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRC_SERVICE_NOT_AVAILABLE	2285	X'000008ED'
MQRC_INITIALIZATION_FAILED	2286	X'000008EE'
MQRC_TERMINATION_FAILED	2287	X'000008EF'
MQRC_UNKNOWN_Q_NAME	2288	X'000008F0'
MQRC_SERVICE_ERROR	2289	X'000008F1'
MQRC_Q_ALREADY_EXISTS	2290	X'000008F2'
MQRC_USER_ID_NOT_AVAILABLE	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_CANCELADO	2297	X'000008F9'
MQRC_FUNCTION_NOT_SUPPORTED	2298	X'000008FA'
MQRC_SELECTOR_TYPE_ERROR	2299	X'000008FB'
MQRC_COMMAND_TYPE_ERROR	2300	X'000008FC'
MQRC_MULTIPLE_INSTANCE_ERROR	2301	X'000008FD'
MQRC_SYSTEM_ITEM_NOT_ALTERABLE	2302	X'000008FE'
MQRC_BAG_CONVERSION_ERROR	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'
MQRC_STRING_ERROR	2307	X'00000903'
MQRC_ENCODING_NOT_SUPPORTED	2308	X'00000904'
MQRC_SELECTOR_NOT_PRESENT	2309	X'00000905'
MQRC_OUT_SELECTOR_ERROR	2310	X'00000906'
MQRC_STRING_TRUNCATED	2311	X'00000907'
MQRC_SELECTOR_WRONG_TYPE	2312	X'00000908'
MQRC_INCONSISTENT_ITEM_TYPE	2313	X'00000909'
MQRC_INDEX_ERROR	2314	X'0000090A'
MQRC_SYSTEM_BAG_NOT_ALTERABLE	2315	X'0000090B'
MQRC_ITEM_COUNT_ERROR	2316	X'0000090C'
MQRC_FORMAT_NOT_SUPPORTED	2317	X'0000090D'
MQRC_SELECTOR_NOT_SUPPORTED	2318	X'0000090E'
MQRC_ITEM_VALOR_ERROR	2319	X'0000090F'
MQRC_HBAG_ERROR	2320	X'00000910'
MQRC_PARAMETER_MISSING	2321	X'00000911'
MQRC_CMD_SERVER_NOT_AVAILABLE	2322	X'00000912'
MQRC_STRING_LENGTH_ERROR	2323	X'00000913'

Tabla 315. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRC_INQUIRY_COMMAND_ERROR	2324	X'00000914'
MQRC_NESTED_BAG_NOT_SUPPORTED	2325	X'00000915'
MQRC_BAG_TIPO_INCORRECTO	2326	X'00000916'
MQRC_ITEM_TYPE_ERROR	2327	X'00000917'
MQRC_SYSTEM_BAG_NOT_DELETABLE	2328	X'00000918'
MQRC_SYSTEM_ITEM_NOT_DELETABLE	2329	X'00000919'
MQRC_CODED_CHAR_SET_ID_ERROR	2330	X'0000091A'
MQRC_MSG_TOKEN_ERROR	2331	X'0000091B'
MQRC_MISSING_WIH	2332	X'0000091C'
MQRC_WIH_ERROR	2333	X'0000091D'
MQRC_RFH_ERROR	2334	X'0000091E'
MQRC_RFH_STRING_ERROR	2335	X'0000091F'
MQRC_RFH_COMMAND_ERROR	2336	X'00000920'
MQRC_RFH_PARM_ERROR	2337	X'00000921'
MQRC_RFH_DUPLICATE_PARM	2338	X'00000922'
MQRC_RFH_PARM_MISSING	2339	X'00000923'
MQRC_CHAR_CONVERSION_ERROR	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_LIBERADO	2344	X'00000928'
MQRC_CF_NOT_AVAILABLE	2345	X'00000929'
MQRC_CF_STRUC_IN_USE	2346	X'0000092A'
MQRC_CF_STRUC_LIST_HDR_IN_USE	2347	X'0000092B'
MQRC_CF_STRUC_AUTH_FAILED	2348	X'0000092C'
MQRC_CF_STRUC_ERROR	2349	X'0000092D'
MQRC_CONN_TAG_NOT_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'
MQRC_UOW_ENLISTMENT_ERROR	2354	X'00000932'
MQRC_UOW_MIX_NOT_SUPPORTED	2355	X'00000933'
MQRC_WXP_ERROR	2356	X'00000934'
MQRC_CURRENT_RECORD_ERROR	2357	X'00000935'
MQRC_NEXT_OFFSET_ERROR	2358	X'00000936'
MQRC_NO_RECORD_AVAILABLE	2359	X'00000937'
MQRC_OBJECT_LEVEL_INCOMPATIBLE	2360	X'00000938'
MQRC_NEXT_RECORD_ERROR	2361	X'00000939'
MQRC_BACKOUT_THRESHOLD_CONTACTADO	2362	X'0000093A'

Tabla 315. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRC_MSG_NOT_MATCHED	2363	X'0000093B'
MQRC_JMS_FORMAT_ERROR	2364	X'0000093C'
MQRC_SEGMENTS_NOT_SUPPORTED	2365	X'0000093D'
MQRC_ERR_CF_LEVEL	2366	X'0000093E'
MQRC_CONFIG_CREATE_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_PARTICIPANT_NOT_DEFINED	2372	X'00000944'
MQRC_CF_STRUC_FAILED	2373	X'00000945'
MQRC_API_EXIT_ERROR	2374	X'00000946'
MQRC_API_EXIT_INIT_ERROR	2375	X'00000947'
MQRC_API_EXIT_TERM_ERROR	2376	X'00000948'
MQRC_EXIT_REASON_ERROR	2377	X'00000949'
MQRC_RESERVED_VALUE_ERROR	2378	X'0000094A'
MQRC_NO_DATA_AVAILABLE	2379	X'0000094B'
MQRC_SCO_ERROR	2380	X'0000094C'
MQRC_KEY_REPOSITORY_ERROR	2381	X'0000094D'
MQRC_CRYPTO_HARDWARE_ERROR	2382	X'0000094E'
MQRC_AUTH_INFO_REC_COUNT_ERROR	2383	X'0000094F'
MQRC_AUTH_INFO_REC_ERROR	2384	X'00000950'
MQRC_AIR_ERROR	2385	X'00000951'
MQRC_AUTH_INFO_TYPE_ERROR	2386	X'00000952'
MQRC_AUTH_INFO_CONN_NAME_ERROR	2387	X'00000953'
MQRC_LDAP_USER_NAME_ERROR	2388	X'00000954'
MQRC_LDAP_USER_NAME_LENGTH_ERR	2389	X'00000955'
MQRC_LDAP_PASSWORD_ERROR	2390	X'00000956'
MQRC_SSL_ALREADY_INITIALIZED	2391	X'00000957'
MQRC_SSL_CONFIG_ERROR	2392	X'00000958'
MQRC_SSL_INITIALIZATION_ERROR	2393	X'00000959'
MQRC_Q_INDEX_TYPE_ERROR	2394	X'0000095A'
MQRC_CFBS_ERROR	2395	X'0000095B'
MQRC_SSL_NOT_ALLOWED	2396	X'0000095C'
MQRC_JSSE_ERROR	2397	X'0000095D'
MQRC_SSL_PEER_NAME_MISMATCH	2398	X'0000095E'
MQRC_SSL_PEER_NAME_ERROR	2399	X'0000095F'
MQRC_UNSUPPORTED_CIPHER_SUITE	2400	X'00000960'
MQRC_SSL_CERTIFICATE_REVOKED	2401	X'00000961'

Tabla 315. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRC_SSL_CERT_STORE_ERROR	2402	X'00000962'
MQRC_CLIENT_EXIT_LOAD_ERROR	2406	X'00000966'
MQRC_CLIENT_EXIT_ERROR	2407	X'00000967'
MQRC_UOW_COMMITTED	2408	X'00000968'
MQRC_SSL_KEY_RESET_ERROR	2409	X'00000969'
MQRC_UNKNOWN_COMPONENT_NAME	2410	X'0000096A'
MQRC_LOGGER_STATUS	2411	X'0000096B'
MQRC_COMMAND_MQSC	2412	X'0000096C'
MQRC_COMMAND_PCF	2413	X'0000096D'
MQRC_CFIF_ERROR	2414	X'0000096E'
MQRC_CFSF_ERROR	2415	X'0000096F'
MQRC_CFGR_ERROR	2416	X'00000970'
MQRC_MSG_NOT_ALLOWED_IN_GROUP	2417	X'00000971'
MQRC_FILTER_OPERATOR_ERROR	2418	X'00000972'
MQRC_NESTED_SELECTOR_ERROR	2419	X'00000973'
MQRC_EPH_ERROR	2420	X'00000974'
MQRC_RFH_FORMAT_ERROR	2421	X'00000975'
MQRC_CFBF_ERROR	2422	X'00000976'
MQRC_CLIENT_CHANNEL_CONFLICTO	2423	X'00000977'
MQRC_SD_ERROR	2424	X'00000978'
MQRC_TOPIC_STRING_ERROR	2425	X'00000979'
MQRC_STS_ERROR	2426	X'0000097A'
MQRC_NO_SUBSCRIPTION	2428	X'0000097C'
MQRC_SUBSCRIPTION_IN_USE	2429	X'0000097D'
MQRC_STAT_TYPE_ERROR	2430	X'0000097E'
MQRC_SUB_USER_DATA_ERROR	2431	X'0000097F'
MQRC_SUB_ALREADY_EXISTS	2432	X'00000980'
MQRC_IDENTITY_MISMATCH	2434	X'00000982'
MQRC_ALTER_SUB_ERROR	2435	X'00000983'
MQRC_DURABILITY_NOT_ALLOWED	2436	X'00000984'
MQRC_NO_RETAINED_MSG	2437	X'00000985'
MQRC_SRO_ERROR	2438	X'00000986'
MQRC_SUB_NAME_ERROR	2440	X'00000988'
MQRC_OBJECT_STRING_ERROR	2441	X'00000989'
MQRC_PROPERTY_NAME_ERROR	2442	X'0000098A'
MQRC_SEGMENTATION_NOT_ALLOWED	2443	X'0000098B'
MQRC_CBD_ERROR	2444	X'0000098C'
MQRC_CTLO_ERROR	2445	X'0000098D'
MQRC_NO_CALLBACKS_ACTIVE	2446	X'0000098E'

Tabla 315. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRC_CALLBACK_NOT_REGISTERED	2448	X'00000990'
MQRC_OPTIONS_CHANGED	2457	X'00000999'
MQRC_READ_AHEAD_MSGS	2458	X'0000099A'
MQRC_SELECTOR_SYNTAX_ERROR	2459	X'0000099B'
MQRC_HMSG_ERROR	2460	X'0000099C'
MQRC_CMHO_ERROR	2461	X'0000099D'
MQRC_DMHO_ERROR	2462	X'0000099E'
MQRC_SMPO_ERROR	2463	X'0000099F'
MQRC_IMPO_ERROR	2464	X'000009A0'
MQRC_PROPERTY_NAME_TOO_BIG	2465	X'000009A1'
MQRC_PROP_VALUE_NOT_CONVERTED	2466	X'000009A2'
MQRC_PROP_TYPE_NOT_SUPPORTED	2467	X'000009A3'
MQRC_PROPERTY_VALUE_TOO_BIG	2469	X'000009A5'
MQRC_PROP_CONV_NOT_SUPPORTED	2470	X'000009A6'
MQRC_PROPERTY_NOT_AVAILABLE	2471	X'000009A7'
MQRC_PROP_NUMBER_FORMAT_ERROR	2472	X'000009A8'
ERROR_TIPO_PROPIEDAD_MQRC	2473	X'000009A9'
MQRC_PROPERTIES_TOO_BIG	2478	X'000009AE'
MQRC_PUT_NO_RETENIDO	2479	X'000009AF'
MQRC_ALIAS_TARGTYPE_CAMBIADO	2480	X'000009B0'
MQRC_DMPO_ERROR	2481	X'000009B1'
MQRC_PD_ERROR	2482	X'000009B2'
MQRC_CALLBACK_TYPE_ERROR	2483	X'000009B3'
MQRC_CBD_OPTIONS_ERROR	2484	X'000009B4'
MQRC_MAX_MSG_LENGTH_ERROR	2485	X'000009B5'
MQRC_CALLBACK_ROUTINE_ERROR	2486	X'000009B6'
MQRC_CALLBACK_LINK_ERROR	2487	X'000009B7'
MQRC_OPERATION_ERROR	2488	X'000009B8'
MQRC_BMHO_ERROR	2489	X'000009B9'
MQRC_UNSUPPORTED_PROPERTY	2490	X'000009BA'
MQRC_PROP_NAME_NOT_CONVERT	2492	X'000009BC'
MQRC_GET_ENABLED	2494	X'000009BE'
MQRC_MODULE_NOT_FOUND	2495	X'000009BF'
MQRC_MODULE_INVALID	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_ACTIVE	2500	X'000009C4'
MQRC_MHBO_ERROR	2501	X'000009C5'

Tabla 315. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRC_PUBLICATION_FAILURE	2502	X'000009C6'
MQRC_SUB_XX_ENCODE_CASE_ONE inhibida	2503	X'000009C7'
MQRC_SELECTOR_ALWAYS_FALSE	2504	X'000009C8'
MQRC_XEPO_ERROR	2507	X'000009CB'
MQRC_DURABILITY_NOT_ALTERABLE	2509	X'000009CD'
MQRC_TOPIC_NOT_ALTERABLE	2510	X'000009CE'
MQRC_SUBLEVEL_NOT_ALTERABLE	2512	X'000009D0'
MQRC_PROPERTY_NAME_LENGTH_ERR	2513	X'000009D1'
MQRC_DUPLICATE_GROUP_SUB	2514	X'000009D2'
MQRC_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'
MQRC_SELECTION_STRING_ERROR	2519	X'000009D7'
MQRC_RES_OBJECT_STRING_ERROR	2520	X'000009D8'
MQRC_CONNECTION_SUSPENDED	2521	X'000009D9'
MQRC_INVALID_DESTINATION	2522	X'000009DA'
MQRC_INVALID_SUBSCRIPTION	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_CONFLICTO	2529	X'000009E1'
MQRC_ASYNC_XA_CONFLICTO	2530	X'000009E2'
MQRC_PUBSUB_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida	2531	X'000009E3'
MQRC_MSG_HANDLE_COPY_FAILURE	2532	X'000009E4'
MQRC_DEST_CLASS_NOT_ALTERABLE	2533	X'000009E5'
MQRC_OPERATION_NOT_ALLOWED	2534	X'000009E6'
MQRC_ACTION_ERROR	2535	X'000009E7'
MQRC_CHANNEL_NOT_AVAILABLE	2537	X'000009E9'
MQRC_HOST_NOT_AVAILABLE	2538	X'000009EA'
MQRC_CHANNEL_CONFIG_ERROR	2539	X'000009EB'
MQRC_UNKNOWN_CHANNEL_NAME	2540	X'000009EC'
MQRC_LOOPING_PUBLICATION	2541	X'000009ED'
MQRC_ALREADY_JOINED	2542	X'000009EE'
MQRC_CHANNEL_SSL_WARNING	2552	X'000009F8'
MQRC_OCSP_URL_ERROR	2553	X'000009F9'
MQRC_CIPHER_SPEC_NOT_SUITE_B	2591	X'00000A1F'

Tabla 315. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRC_SUITE_B_ERROR	2592	X'00000A20'
MQRC_REOPEN_EXCL_INPUT_ERROR	6100	X'000017D4'
MQRC_REOPEN_INQUIRE_ERROR	6101	X'000017D5'
MQRC_REOPEN_SAVED_CONTEXT_ERR	6102	X'000017D6'
MQRC_REOPEN_TEMPORARY_Q_ERROR	6103	X'000017D7'
MQRC_ATTRIBUTE_LOCKED	6104	X'000017D8'
MQRC_CURSOR_NOT_VALID	6105	X'000017D9'
MQRC_ENCODING_ERROR	6106	X'000017DA'
MQRC_STRUC_ID_ERROR	6107	X'000017DB'
MQRC_NULL_POINTER	6108	X'000017DC'
MQRC_NO_CONNECTION_REFERENCE	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_INSUFICIENT_BUFFER	6113	X'000017E1'
MQRC_INSUFICIENT_DATA	6114	X'000017E2'
MQRC_DATA_TRUNCADO	6115	X'000017E3'
MQRC_LONGITUD_CERO	6116	X'000017E4'
MQRC_NEGATIVE_LENGTH	6117	X'000017E5'
DESPLAZAMIENTO_NEGATIVO_MQRC	6118	X'000017E6'
MQRC_INCONSISTENT_FORMAT	6119	X'000017E7'
MQRC_INCONSISTENT_OBJECT_STATE	6120	X'000017E8'
MQRC_CONTEXT_OBJECT_NOT_VALID	6121	X'000017E9'
MQRC_CONTEXT_OPEN_ERROR	6122	X'000017EA'
MQRC_STRUC_LENGTH_ERROR	6123	X'000017EB'
MQRC_NOT_CONNECTED	6124	X'000017EC'
MQRC_NOT_OPEN	6125	X'000017ED'
MQRC_DISTRIBUTION_LIST_EMPTY	6126	X'000017EE'
MQRC_INCONSISTENT_OPEN_OPTIONS	6127	X'000017EF'
VERSIÓN_ERROR_MQRC	6128	X'000017F0'
ERROR_REFERENCIA_MQRC	6129	X'000017F1'

MQRCCF_ * (Códigos de razón de cabecera de formato de mandato)

Tabla 316. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCCF_CFH_TYPE_ERROR	3001	X'00000BB9'
MQRCCF_CFH_LENGTH_ERROR	3002	X'00000BBA'
MQRCCF_CFH_VERSION_ERROR	3003	X'00000BBB'
MQRCCF_CFH_MSG_SEQ_NUMBER_ERR	3004	X'00000BBC'

Tabla 316. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRCCF_CFH_CONTROL_ERROR	3005	X'00000BBD'
MQRCCF_CFH_PARM_COUNT_ERROR	3006	X'00000BBE'
MQRCCF_CFH_COMMAND_ERROR	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'
MQRCCF_FORCE_VALUE_ERROR	3012	X'00000BC4'
MQRCCF_STRUCTURE_TYPE_ERROR	3013	X'00000BC5'
MQRCCF_CFIN_PARM_ID_ERROR	3014	X'00000BC6'
MQRCCF_CFST_PARM_ID_ERROR	3015	X'00000BC7'
MQRCCF_MSG_LENGTH_ERROR	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'
MQRCCF_REPLACE_VALUE_ERROR	3025	X'00000BD1'
MQRCCF_CFIL_DUPLICATE_VALUE	3026	X'00000BD2'
MQRCCF_CFIL_COUNT_ERROR	3027	X'00000BD3'
MQRCCF_CFIL_LENGTH_ERROR	3028	X'00000BD4'
MQRCCF_QUIESCE_VALUE_ERROR	3029	X'00000BD5'
MQRCCF_MODE_VALOR_ERROR	3029	X'00000BD5'
MQRCCF_MSG_SEQ_NUMBER_ERROR	3030	X'00000BD6'
MQRCCF_PING_DATA_COUNT_ERROR	3031	X'00000BD7'
MQRCCF_PING_DATA_COMPARE_ERROR	3032	X'00000BD8'
MQRCCF_CFSL_PARM_ID_ERROR	3033	X'00000BD9'
MQRCCF_CHANNEL_TYPE_ERROR	3034	X'00000BDA'
MQRCCF_PARM_SEQUENCE_ERROR	3035	X'00000BDB'
MQRCCF_XMIT_PROTOCOL_TYPE_ERR	3036	X'00000BDC'
MQRCCF_BATCH_SIZE_ERROR	3037	X'00000BDD'
MQRCCF_DISC_INT_ERROR	3038	X'00000BDE'
MQRCCF_SHORT_RETRY_ERROR	3039	X'00000BDF'
MQRCCF_SHORT_TIMER_ERROR	3040	X'00000BE0'
MQRCCF_LONG_RETRY_ERROR	3041	X'00000BE1'
MQRCCF_LONG_TIMER_ERROR	3042	X'00000BE2'

Tabla 316. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRCCF_SEQ_NUMBER_WRAP_ERROR	3043	X'00000BE3'
MQRCCF_MAX_MSG_LENGTH_ERROR	3044	X'00000BE4'
MQRCCF_PUT_AUTH_ERROR	3045	X'00000BE5'
MQRCCF_PURGE_VALUE_ERROR	3046	X'00000BE6'
MQRCCF_CFIL_PARM_ID_ERROR	3047	X'00000BE7'
MQRCCF_MSG_TRUNCATED	3048	X'00000BE8'
MQRCCF_CCSID_ERROR	3049	X'00000BE9'
MQRCCF_ENCODING_ERROR	3050	X'00000BEA'
MQRCCF_QUEUES_VALUE_ERROR	3051	X'00000BEB'
MQRCCF_DATA_CONV_VALUE_ERROR	3052	X'00000BEC'
MQRCCF_INDOUBT_VALUE_ERROR	3053	X'00000BED'
MQRCCF_ESCAPE_TYPE_ERROR	3054	X'00000BEE'
MQRCCF_REPOS_VALUE_ERROR	3055	X'00000BEF'
MQRCCF_CHANNEL_TABLE_ERROR	3062	X'00000BF6'
MQRCCF_MCA_TYPE_ERROR	3063	X'00000BF7'
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'
MQRCCF_CFSL_COUNT_ERROR	3068	X'00000BFC'
MQRCCF_CFSL_STRING_LENGTH_ERR	3069	X'00000BFD'
MQRCCF_BROKER_DELETED	3070	X'00000BFE'
MQRCCF_STREAM_ERROR	3071	X'00000BFF'
MQRCCF_TOPIC_ERROR	3072	X'00000C00'
MQRCCF_NO_REGISTRADO	3073	X'00000C01'
MQRCCF_Q_MGR_NAME_ERROR	3074	X'00000C02'
MQRCCF_INCORRECT_STREAM	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_ID_CORREL_ERROR	3080	X'00000C08'
MQRCCF_NO_AUTORIZADO	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'
MQRCCF_Q_MGR_CCSID_ERROR	3086	X'00000C0E'
MQRCCF_DEL_OPTIONS_ERROR	3087	X'00000C0F'

Tabla 316. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRCCF_CLUSTER_NAME_CONFLICTO	3088	X'00000C10'
MQRCCF_REPOS_NAME_CONFLICTO	3089	X'00000C11'
MQRCCF_CLUSTER_Q_USAGE_ERROR	3090	X'00000C12'
MQRCCF_ACTION_VALUE_ERROR	3091	X'00000C13'
MQRCCF_COMMS_LIBRARY_ERROR	3092	X'00000C14'
MQRCCF_NETBIOS_NAME_ERROR	3093	X'00000C15'
MQRCCF_BROKER_COMMAND_FAILED	3094	X'00000C16'
MQRCCF_CFST_CONFLICTING_PARM	3095	X'00000C17'
MQRCCF_PATH_NOT_VALID	3096	X'00000C18'
MQRCCF_PARM_SYNTAX_ERROR	3097	X'00000C19'
MQRCCF_PWD_LENGTH_ERROR	3098	X'00000C1A'
MQRCCF_FILTER_ERROR	3150	X'00000C4E'
MQRCCF_USUARIO_INCORRECTO	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_NOT_AVAILABLE	3162	X'00000C5A'
MQRCCF_DISC_RETRY_ERROR	3163	X'00000C5B'
MQRCCF_ALLOC_RETRY_ERROR	3164	X'00000C5C'
MQRCCF_ALLOC_SLOW_TIMER_ERROR	3165	X'00000C5D'
MQRCCF_ALLOC_FAST_TIMER_ERROR	3166	X'00000C5E'
MQRCCF_NÚMERO_PUERTO_ERROR	3167	X'00000C5F'
MQRCCF_CHL_SYSTEM_NOT_ACTIVE	3168	X'00000C60'
MQRCCF_ENTITY_NAME_MISSING	3169	X'00000C61'
MQRCCF_PROFILE_NAME_ERROR	3170	X'00000C62'
MQRCCF_AUTH_VALUE_ERROR	3171	X'00000C63'
MQRCCF_AUTH_VALUE_MISSING	3172	X'00000C64'
MQRCCF_OBJECT_TYPE_MISSING	3173	X'00000C65'
MQRCCF_CONNECTION_ID_ERROR	3174	X'00000C66'
MQRCCF_LOG_TYPE_ERROR	3175	X'00000C67'
MQRCCF_PROGRAM_NOT_AVAILABLE	3176	X'00000C68'
MQRCCF_PROGRAM_AUTH_FAILED	3177	X'00000C69'
MQRCCF_NONE_FOUND	3200	X'00000C80'
MQRCCF_SECURITY_SWITCH_OFF	3201	X'00000C81'

Tabla 316. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRCCF_SECURITY_REFRESH_FAILED	3202	X'00000C82'
MQRCCF_PARM_CONFLICT	3203	X'00000C83'
MQRCCF_COMMAND_INHIBIDO	3204	X'00000C84'
MQRCCF_OBJECT_BEING_DELETED	3205	X'00000C85'
MQRCCF_STORAGE_CLASS_IN_USE	3207	X'00000C87'
MQRCCF_NOMBRE_OBJETO_RESTRINGIDO	3208	X'00000C88'
MQRCCF_OBJECT_LIMIT_SUPERADO	3209	X'00000C89'
MQRCCF_OBJECT_OPEN_FORCE	3210	X'00000C8A'
MQRCCF_DISPOSITION_CONFLICTO	3211	X'00000C8B'
MQRCCF_Q_MGR_NOT_IN_QSG	3212	X'00000C8C'
MQRCCF_ATTR_VALUE_FIXED	3213	X'00000C8D'
MQRCCF_NAMELIST_ERROR	3215	X'00000C8F'
MQRCCF_NO_CHANNEL_INITIATOR	3217	X'00000C91'
MQRCCF_CHANNEL_INITIATOR_ERROR	3218	X'00000C92'
MQRCCF_COMMAND_LEVEL_CONFLICTO	3222	X'00000C96'
MQRCCF_Q_ATTR_CONFLICTO	3223	X'00000C97'
MQRCCF_EVENTS_DISABLED	3224	X'00000C98'
MQRCCF_COMMAND_SCOPE_ERROR	3225	X'00000C99'
MQRCCF_COMMAND_REPLY_ERROR	3226	X'00000C9A'
MQRCCF_FUNCTION_RESTRICTED	3227	X'00000C9B'
MQRCCF_PARM_MISSING	3228	X'00000C9C'
MQRCCF_PARM_VALUE_ERROR	3229	X'00000C9D'
MQRCCF_COMMAND_LENGTH_ERROR	3230	X'00000C9E'
MQRCCF_COMMAND_ORIGIN_ERROR	3231	X'00000C9F'
MQRCCF_LISTENER_CONFLICTO	3232	X'00000CA0'
MQRCCF_LISTENER_STARTED	3233	X'00000CA1'
MQRCCF_LISTENER_STOPPED	3234	X'00000CA2'
MQRCCF_CHANNEL_ERROR	3235	X'00000CA3'
MQRCCF_CF_STRUC_ERROR	3236	X'00000CA4'
MQRCCF_UNKNOWN_USER_ID	3237	X'00000CA5'
MQRCCF_UNEXPECTED_ERROR	3238	X'00000CA6'
MQRCCF_NO_XCF_PARTNER	3239	X'00000CA7'
MQRCCF_CFGR_PARM_ID_ERROR	3240	X'00000CA8'
MQRCCF_CFIF_LENGTH_ERROR	3241	X'00000CA9'
MQRCCF_CFIF_OPERATOR_ERROR	3242	X'00000CAA'
MQRCCF_CFIF_PARM_ID_ERROR	3243	X'00000CAB'
MQRCCF_CFSF_FILTER_VAL_LEN_ERR	3244	X'00000CAC'
MQRCCF_CFSF_LENGTH_ERROR	3245	X'00000CAD'
MQRCCF_CFSF_OPERATOR_ERROR	3246	X'00000CAE'

Tabla 316. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRCCF_CFSF_PARM_ID_ERROR	3247	X'00000CAF'
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_SERV_STATUS_NOT_FOUND	3252	X'00000CB4'
MQRCCF_SERVICE_STOPPED	3253	X'00000CB5'
MQRCCF_CFBS_DUPLICATE_PARM	3254	X'00000CB6'
MQRCCF_CFBS_LENGTH_ERROR	3255	X'00000CB7'
MQRCCF_CFBS_PARM_ID_ERROR	3256	X'00000CB8'
MQRCCF_CFBS_STRING_LENGTH_ERR	3257	X'00000CB9'
MQRCCF_CFGR_LENGTH_ERROR	3258	X'00000CBA'
MQRCCF_CFGR_PARM_COUNT_ERROR	3259	X'00000CBB'
MQRCCF_CONN_NOT_STOPPED	3260	X'00000CBC'
MQRCCF_SERVICE_REQUEST_PENDING	3261	X'00000CBD'
MQRCCF_NO_START_CMD	3262	X'00000CBE'
MQRCCF_NO_STOP_CMD	3263	X'00000CBF'
MQRCCF_CFBF_LENGTH_ERROR	3264	X'00000CC0'
MQRCCF_CFBF_PARM_ID_ERROR	3265	X'00000CC1'
MQRCCF_CFBF_OPERATOR_ERROR	3266	X'00000CC2'
MQRCCF_CFBF_FILTER_VAL_LEN_ERR	3267	X'00000CC3'
MQRCCF_LISTENER_STILL_ACTIVE	3268	X'00000CC4'
MQRCCF_DEF_XMIT_Q_CLUS_ERROR	3269	X'0000CC5'
MQRCCF_TOPICSTR_ALREADY_EXISTS	3300	X'00000CE4'
MQRCCF_SHARING_CONVS_ERROR	3301	X'00000CE5'
MQRCCF_SHARING_CONVS_TYPE	3302	X'00000CE6'
MQRCCF_SECURITY_CASE_CONFLICTO	3303	X'00000CE7'
MQRCCF_TOPIC_TYPE_ERROR	3305	X'00000CE9'
MQRCCF_MAX_INSTANCES_ERROR	3306	X'00000CEA'
MQRCCF_MAX_INSTS_PER_CLNT_ERR	3307	X'00000CEB'
MQRCCF_TOPIC_STRING_NOT_FOUND	3308	X'00000CEC'
MQRCCF_SUBSCRIPTION_POINT_ERR	3309	X'00000CED'
MQRCCF_SUB_ALREADY_EXISTS	3311	X'00000CEF'
MQRCCF_UNKNOWN_OBJECT_NAME	3312	X'00000CF0'
MQRCCF_REMOTE_Q_NAME_ERROR	3313	X'00000CF1'
MQRCCF_DURABILITY_NOT_ALLOWED	3314	X'00000CF2'
MQRCCF_HOBJ_ERROR	3315	X'00000CF3'
MQRCCF_DEST_NAME_ERROR	3316	X'00000CF4'
MQRCCF_INVALID_DESTINATION	3317	X'00000CF5'

Tabla 316. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRCCF_PUBSUB_XX_ENCODE_CASE_ONE inhibida	3318	X'00000CF6'
MQRCCF_CHLAUTH_TYPE_ERROR	3326	X'00000CFE'
MQRCCF_CHLAUTH_ACTION_ERROR	3327	X'00000CFF'
MQRCCF_CHLAUTH_USERSRC_ERROR	3335	X'00000D07'
MQRCCF_TIPO_AUTORIZACIÓN_ERRÓNEO	3336	X'00000D08'
MQRCCF_CHLAUTH_ALREADY_EXISTS	3337	X'00000D09'
MQRCCF_CHLAUTH_NOT_FOUND	3338	X'00000D0A'
MQRCCF_EQUIVOC_CHLAUTH_ACTION	3339	X'00000D0B'
MQRCCF_INJU_CHLAUTH_USERSRC	3340	X'00000D0C'
MQRCCF_CHLAUTH_WARN_ERROR	3341	X'00000D0D'
MQRCCF_ERRÓNEA_CHLAUTH_MATCH	3342	X'00000D0E'
MQRCCF_IPADDR_RANGE_CONFLICTO	3343	X'00000D0F'
MQRCCF_CHLAUTH_MAX_SUPERADO	3344	X'00000D10'
MQRCCF_IPADDR_ERROR	3345	X'00000D11'
MQRCCF_IPADDR_RANGE_ERROR	3346	X'00000D12'
MQRCCF_PROFILE_NAME_MISSING	3347	X'00000D13'
MQRCCF_CHLAUTH_CLNTUSER_ERROR	3348	X'00000D14'
MQRCCF_CHLAUTH_NAME_ERROR	3349	X'00000D15'
MQRCCF_SUITE_B_ERROR	3353	X'00000D19'
MQRCCF_PSCLUS_DISABLED_TOPDEF	3359	X'00000D1F'
MQRCCF_PSCLUS_TOPIC_EXISTS	3360	X'00000D20'
MQRCCF_OBJECT_ALREADY_EXISTS	4001	X'00000FA1'
MQRCCF_OBJECT_TIPO_INCORRECTO	4002	X'00000FA2'
MQRCCF_LIKE_OBJECT_TIPO_INCORRECTO	4003	X'00000FA3'
MQRCCF_OBJECT_OPEN	4004	X'00000FA4'
MQRCCF_ATTR_VALUE_ERROR	4005	X'00000FA5'
MQRCCF_UNKNOWN_Q_MGR	4006	X'00000FA6'
MQRCCF_Q_TIPO_INCORRECTO	4007	X'00000FA7'
MQRCCF_OBJECT_NAME_ERROR	4008	X'00000FA8'
MQRCCF_ALLOCATE_FAILED	4009	X'00000FA9'
MQRCCF_HOST_NOT_AVAILABLE	4010	X'00000FAA'
MQRCCF_CONFIGURATION_ERROR	4011	X'00000FAB'
MQRCCF_CONNECTION_RECHAZADO	4012	X'00000FAC'
MQRCCF_ENTRY_ERROR	4013	X'00000FAD'
MQRCCF_SEND_FAILED	4014	X'00000FAE'
MQRCCF_RECEIVED_DATA_ERROR	4015	X'00000FAF'
MQRCCF_RECEIVE_FAILED	4016	X'00000FB0'
MQRCCF_CONNECTION_CLOSED	4017	X'00000FB1'
MQRCCF_NO_STORAGE	4018	X'00000FB2'

Tabla 316. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
MQRCCF_NO_COMMS_MANAGER	4019	X'00000FB3'
MQRCCF_LISTENER_NOT_STARTED	4020	X'00000FB4'
MQRCCF_BIND_FAILED	4024	X'00000FB8'
MQRCCF_CHANNEL_INDOUBT	4025	X'00000FB9'
MQRCCF_MQCONN_FAILED	4026	X'00000FBA'
MQRCCF_MQOPEN_FAILED	4027	X'00000FBB'
MQRCCF_MQGET_FAILED	4028	X'00000FBC'
MQRCCF_MQPUT_FAILED	4029	X'00000FBD'
MQRCCF_ERROR	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_TERMINANDO	4035	X'00000FC3'
MQRCCF_MQINQ_FAILED	4036	X'00000FC4'
MQRCCF_NOT_XMIT_Q	4037	X'00000FC5'
MQRCCF_CHANNEL_DISABLED	4038	X'00000FC6'
MQRCCF_USER_EXIT_NOT_AVAILABLE	4039	X'00000FC7'
MQRCCF_COMMIT_FAILED	4040	X'00000FC8'
MQRCCF_TIPO_CANAL_INCORRECTO	4041	X'00000FC9'
MQRCCF_CHANNEL_ALREADY_EXISTS	4042	X'00000FCA'
MQRCCF_DATA_TOO_LARGE	4043	X'00000FCB'
MQRCCF_CHANNEL_NAME_ERROR	4044	X'00000FCC'
MQRCCF_XMIT_Q_NAME_ERROR	4045	X'00000FCD'
MQRCCF_MCA_NAME_ERROR	4047	X'00000FCF'
MQRCCF_SEND_EXIT_NAME_ERROR	4048	X'00000FD0'
MQRCCF_SEC_EXIT_NAME_ERROR	4049	X'00000FD1'
MQRCCF_MSG_EXIT_NAME_ERROR	4050	X'00000FD2'
MQRCCF_RCV_EXIT_NAME_ERROR	4051	X'00000FD3'
MQRCCF_XMIT_Q_NAME_INJUSTA_TYPE	4052	X'00000FD4'
MQRCCF_MCA_NAME_INJUSTA_TYPE	4053	X'00000FD5'
MQRCCF_DISC_INT_TIPO_INCORRECTO	4054	X'00000FD6'
MQRCCF_SHORT_RETRY_INJU_TYPE	4055	X'00000FD7'
MQRCCF_SHORT_TIMER_INJU_TYPE	4056	X'00000FD8'
MQRCCF_LONG_RETRY_INJU_TYPE	4057	X'00000FD9'
MQRCCF_LONG_TIMER_INJUSTA_TYPE	4058	X'00000FDA'
MQRCCF_PUT_AUTH_TIPO_ERRÓNEO	4059	X'00000FDB'
MQRCCF_KEEP_ALIVE_INT_ERROR	4060	X'00000FDC'
MQRCCF_MISSING_CONN_NAME	4061	X'00000FDD'

Tabla 316. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCCF_CONN_NAME_ERROR	4062	X'00000FDE'
MQRCCF_MQSET_FAILED	4063	X'00000FDF'
MQRCCF_CHANNEL_NOT_ACTIVE	4064	X'00000FE0'
MQRCCF_TERMINATED_BY_SEC_EXIT	4065	X'00000FE1'
MQRCCF_DYNAMIC_Q_SCOPE_ERROR	4067	X'00000FE3'
MQRCCF_CELL_DIR_NO_DISPONIBLE	4068	X'00000FE4'
MQRCCF_MR_COUNT_ERROR	4069	X'00000FE5'
MQRCCF_MR_COUNT_WRONG_TYPE	4070	X'00000FE6'
MQRCCF_MR_EXIT_NAME_ERROR	4071	X'00000FE7'
MQRCCF_MR_EXIT_NAME_INJUSTA_TYPE	4072	X'00000FE8'
MQRCCF_MR_INTERVAL_ERROR	4073	X'00000FE9'
MQRCCF_MR_INTERVAL_TIPO_INCORRECTO	4074	X'00000FEA'
MQRCCF_NPM_SPEED_ERROR	4075	X'00000FEB'
MQRCCF_NPM_SPEED_WRONG_TYPE	4076	X'00000FEC'
MQRCCF_HB_INTERVAL_ERROR	4077	X'00000FED'
MQRCCF_HB_INTERVAL_TIPO_ERRÓNEO	4078	X'00000FEE'
MQRCCF_CHAD_ERROR	4079	X'00000FEF'
MQRCCF_CHAD_TIPO_INCORRECTO	4080	X'00000FF0'
MQRCCF_CHAD_EVENT_ERROR	4081	X'00000FF1'
MQRCCF_CHAD_EVENT_INJU_TYPE	4082	X'00000FF2'
MQRCCF_CHAD_EXIT_ERROR	4083	X'00000FF3'
MQRCCF_CHAD_EXIT_TIPO_INCORRECTO	4084	X'00000FF4'
MQRCCF_SUPPRESSED_BY_EXIT	4085	X'00000FF5'
MQRCCF_BATCH_INT_ERROR	4086	X'00000FF6'
MQRCCF_BATCH_INT_TIPO_INCORRECTO	4087	X'00000FF7'
MQRCCF_NET_PRIORITY_ERROR	4088	X'00000FF8'
MQRCCF_NET_PRIORITY_TIPO_INCORRECTO	4089	X'00000FF9'
MQRCCF_CHANNEL_CLOSED	4090	X'00000FFA'
MQRCCF_Q_STATUS_NOT_FOUND	4091	X'00000FFB'
MQRCCF_SSL_CIPHER_SPEC_ERROR	4092	X'00000FFC'
MQRCCF_SSL_PEER_NAME_ERROR	4093	X'00000FFD'
MQRCCF_SSL_CLIENT_AUTH_ERROR	4094	X'00000FFE'
MQRCCF_RETAINED_NOT_SUPPORTED	4095	X'00000FFF'

MQRCN_ * (Constantes de reconexión de cliente)

Tabla 317. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCN_NO	0	X'00000000'
MQRCN_YES	1	X'00000001'

Tabla 317. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCN_Q_MGR	2	X'00000002'
MQRCN_DISABLED	3	X'00000003'

MQRCVTIME_ * (tipos de tiempo de espera de recepción)

Tabla 318. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRCVTIME_MULTIPLY	0	X'00000000'
MQRCVTIME_ADD	1	X'00000001'
MQRCVTIME_EQUAL	2	X'00000002'

MQREADA_ * (Leer valores de cabecera)

Tabla 319. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQREADA_NO	0	X'00000000'
MQREADA_SÍ	1	X'00000001'
MQREADA_DISABLED	2	X'00000002'
MQREADA_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida	3	X'00000003'
MQREADA_BACKLOG	4	X'00000004'

MQRECORDING_ * (Opciones de grabación)

Tabla 320. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRECORDING_INHABILITADO	0	X'00000000'
MQRECORDING_Q	1	X'00000001'
MQRECORDING_MSG	2	X'00000002'

MQREGO_ * (Opciones de registro de publicación/suscripción)

Tabla 321. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQREGO_NONE	0	X'00000000'
MQREGO_CORREL_ID_AS_IDENTITY	1	X'00000001'
MQREGO_ANONYMOUS	2	X'00000002'
MQREGO_LOCAL	4	X'00000004'
MQREGO_DIRECT_REQUESTS	8	X'00000008'
MQREGO_NEW_PUBLICATIONS_ONLY	16	X'00000010'
MQREGO_PUBLISH_ON_REQUEST_ONLY	32	X'00000020'
MQREGO_DEREGISTER_ALL	64	X'00000040'
MQREGO_INCLUDE_STREAM_NAME	128	X'00000080'
MQREGO_INFORM_IF_RETAINED	256	X'00000100'
MQREGO_DUPLICATES_OK	512	X'00000200'

Tabla 321. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQREGO_NON_PERSISTENT	1024	X'00000400'
MQREGO_PERSISTENT	2048	X'00000800'
MQREGO_PERSISTENT_AS_PUBLISH	4096	X'00001000'
MQREGO_PERSISTENT_AS_Q	8192	X'00002000'
MQREGO_ADD_NAME	16384	X'00004000'
MQREGO_NO_ALTERACIÓN	32768	X'00008000'
MQREGO_FULL_RESPONSE	65536	X'00010000'
MQREGO_JOIN_SHARED	131072	X'00020000'
MQREGO_JOIN_EXCLUSIVE	262144	X'00040000'
MQREGO_LEAVE_ONLY	524288	X'00080000'
MQREGO_VARIABLE_USER_ID	1048576	X'00100000'
MQREGO_LOCKED	2097152	X'00200000'

MQRFH_ * (Estructura y distintivos de cabecera de reglas y formateo)

Reglas y estructura de cabecera de formato

Tabla 322. Estructuras de constantes	
Nombre	**Estructura**
MQRFH_STRUC_ID	"RFH¬"
MQRFH_STRUC_ID_ARRAY	'R','F','H','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 323. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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'

Reglas y distintivos de cabecera de formato

Tabla 324. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRFH_NONE	0	X'00000000'
MQRFH_NO_FLAGS	0	X'00000000'

MQRFH2_* (Etiqueta de opciones de publicación/suscripción RFH2 Etiquetas de carpeta de nivel superior)

Tabla 325. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRFH2_NAME_VALUE_VERSION	1	X'00000001'

MQRFH2_* (Nombres de etiqueta de opciones de publicación/suscripción)

Tabla 326. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRFH2_PUBSUB_CMD_FOLDER	"psc"	
MQRFH2_PUBSUB_RESP_FOLDER	"pscr"	
MQRFH2_MSG_CONTENT_FOLDER	"mcd"	
MQRFH2_USER_FOLDER	"usr"	

MQRFH2_* (Nombres de etiqueta XML de opciones de publicación/ suscripción)

Tabla 327. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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_ * (Longitud devuelta)

Tabla 328. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRL_UNDEFINED	-1	X'FFFFFFFF'

MQRMH_ * (estructura de cabecera de mensaje de referencia)

Tabla 329. Estructuras de constantes	
Nombre	**Estructura**
MQRMH_STRUC_ID	"RMH¬"
MQRMH_STRUC_ID_ARRAY	'R','M','H','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 330. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRMH_VERSION_1	1	X'00000001'
MQRMH_CURRENT_VERSION	1	X'00000001'

MQRMHF_ * (Distintivos de cabecera de mensaje de referencia)

Tabla 331. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRMHF_LAST	1	X'00000001'

Tabla 331. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRMHF_NOT_LAST	0	X'00000000'

MQRO_ * (Opciones de informe)

Tabla 332. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRO_EXCEPTION	16777216	X'01000000'
MQRO_EXCEPTION_WITH_DATA	50331648	X'03000000'
MQRO_EXCEPTION_WITH_FULL_DATA	117440512	X'07000000'
MQRO_EXPIRATION	2097152	X'00200000'
MQRO_EXPIRATION_WITH_DATA	6291456	X'00600000'
MQRO_EXPIRATION_WITH_FULL_DATA	14680064	X'00E00000'
MQRO_COA	256	X'00000100'
MQRO_COA_WITH_DATA	768	X'00000300'
MQRO_COA_WITH_FULL_DATA	1792	X'00000700'
MQRO_COD	2048	X'00000800'
MQRO_COD_WITH_DATA	6144	X'00001800'
MQRO_COD_WITH_FULL_DATA	14336	X'00003800'
MQRO_PAN	1	X'00000001'
MQRO_NAN	2	X'00000002'
MQRO_ACTIVITY	4	X'00000004'
MQRO_NEW_MSG_ID	0	X'00000000'
MQRO_PASS_MSG_ID	128	X'00000080'
MQRO_COPY_MSG_ID_TO_CORREL_ID	0	X'00000000'
MQRO_PASS_CORREL_ID	64	X'00000040'
MQRO_DEAD_LETTER_Q	0	X'00000000'
MQRO_DISCARD_MSG	134217728	X'08000000'
MQRO_PASS_DISCARD_AND_EXPIRY	16384	X'00004000'
MQRO_NONE	0	X'00000000'

MQRO_ * (máscaras de opciones de informe)

Tabla 333. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQRO_REJECT_UNSUP_MASK	270270464	X'101C0000'
MQRO_ACCEPT_UNSUP_MASK	-270532353	X'EFE000FF'
MQRO_ACCEPT_UNSUP_IF_XMIT_MASK	261888	X'0003FF00'

MQROUTE_ * (ruta de rastreo)

Máximo de actividades de ruta de rastreo (MQIACF_MAX_ACTIVITIES)

Tabla 334. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQROUTE_UNLIMITED_ACTIVITIES	0	X'00000000'

Detalle de ruta de rastreo (MQIACF_ROUTE_DETAIL)

Tabla 335. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQROUTE_DETAIL_LOW	2	X'00000002'
MQROUTE_DETAIL_MEDIUM	8	X'00000008'
MQROUTE_DETAIL_HIGH	32	X'00000020'

Reenvío de ruta de rastreo (MQIACF_ROUTE_FORWARDING)

Tabla 336. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQROUTE_FORWARD_ALL	256	X'00000100'
MQROUTE_FORWARD_IF_SUPPORTED	512	X'00000200'
MQROUTE_FORWARD_REJ_UNSUP_MASK	-65536	X'FFFF0000'

Entrega de ruta de rastreo (MQIACF_ROUTE_DELIVERY)

Tabla 337. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQROUTE_DELIVER_YES	4096	X'00001000'
MQROUTE_DELIVER_NO	8192	X'00002000'
MQROUTE_DELIVER_REJ_UNSUP_MASK	-65536	X'FFFF0000'

Acumulación de ruta de rastreo (MQIACF_ROUTE_ACUMULACIÓN)

Tabla 338. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQROUTE_ACCUMULATE_NONE	65539	X'00010003'
MQROUTE_ACCUMULATE_IN_MSG	65540	X'00010004'
MQROUTE_ACCUMULATE_AND_REPLY	65541	X'00010005'

MQRP_ * (Formato de mandato, Opciones de sustitución)

Tabla 339. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQRP_SÍ	1	X'00000001'
MQRP_NO	0	X'00000000'

MQRQ_ * (Formato de mandato, calificadores de razón)

Tabla 340. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQRQ_CONN_NOT_AUTHORIZED	1	X'00000001'
MQRQ_OPEN_NOT_AUTHORIZED	2	X'00000002'
MQRQ_CLOSE_NOT_AUTHORIZED	3	X'00000003'
MQRQ_CMD_NOT_AUTHORIZED	4	X'00000004'
MQRQ_Q_MGR_DETENIENDO	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_STOPPED_OK	11	X'0000000B'
MQRQ_BRIDGE_STOPPED_ERROR	12	X'0000000C'
MQRQ_SSL_HANDSHAKE_ERROR	13	X'0000000D'
MQRQ_SSL_CIPHER_SPEC_ERROR	14	X'0000000E'
MQRQ_SSL_CLIENT_AUTH_ERROR	15	X'0000000F'
MQRQ_SSL_PEER_NAME_ERROR	16	X'00000010'
MQRQ_SUB_NOT_AUTHORIZED	17	X'00000011'
MQRQ_SUB_DEST_NOT_AUTHORIZED	18	X'00000012'
MQRQ_SSL_UNKNOWN_REVOCATION	19	X'00000013'

MQRT_ * (Formato de mandato, Tipos de renovación)

Tabla 341. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
CONFIGURACIÓN_MQR	1	X'00000001'
MQRT_CADUCIDAD	2	X'00000002'
MQRT_NSPROC	3	X'00000003'
MQRT_PROXYSUB	4	X'00000004'

MQRU_ * (sólo solicitud)

Tabla 342. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQRU_PUBLISH_ON_REQUEST	1	X'00000001'
MQRU_PUBLISH_ALL	2	X'00000002'

MQSCA_ * (Autenticación de cliente SSL)

Tabla 343. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSCA_REQUIRED	0	X'00000000'

Tabla 343. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSCA_OPTIONAL	1	X'00000001'

MQSCO_ * (opciones de configuración SSL)

Estructura de opciones de configuración SSL

Tabla 344. Estructuras de constantes	
Nombre	**Estructura**
MQSCO_ID_ESTRUCTURA	"SCO¬"
MQSCO_STRUC_ID_ARRAY	'S','C','O','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 345. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSCO_VERSION_1	1	X'00000001'
MQSCO_VERSION_2	2	X'00000002'
MQSCO_VERSION_3	3	X'00000003'
MQSCO_VERSION_4	4	X'00000004'
MQSCO_VERSIÓN_ACTUAL	4	X'00000004'

Nota: El símbolo ¬ representa un único carácter en blanco.

Recuento de restablecimiento de claves de opciones de configuración SSL

Tabla 346. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSCO_RESET_COUNT_DEFAULT	0	X'00000000'

Formato de mandato de ámbito de definición de cola

Tabla 347. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSCO_Q_MGR	1	X'00000001'
MQSCO_CELL	2	X'00000002'

MQSCOPE_ * (ámbito de publicación)

Tabla 348. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSCOPE_TODOS	0	X'00000000'
MQSCOPE_AS_PARENT	1	X'00000001'
MQSCOPE_QMGR	4	X'00000004'

MQSCYC_ * (Caso de seguridad)

Tabla 349. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSCYC_UPPER	0	X'00000000'
MQSCYC_MIXED	1	X'00000001'

MQSD_ * (Estructura de descriptor de objeto)

Tabla 350. Nombres y estructuras de constantes	
Nombre	**Estructura**
MQSD_STRUC_ID	"SD¬¬"
MQSD_STRUC_ID_ARRAY	'S','D','¬','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 351. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSD_VERSION_1	1	X'00000001'
MQSD_CURRENT_VERSION	1	X'00000001'

MQSECITEM_ * (Formato de mandato, Elementos de seguridad)

Tabla 352. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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'
MQSECITEM_MXNLIST	8	X'00000008'
MQSECITEM_MXPROC	9	X'00000009'
MQSECITEM_MXQUEUE	10	X'0000000A'
MQSECITEM_MXTOPIC	11	X'0000000B'

MQSECSW_ * (Formato de mandato, Conmutadores de seguridad y estados de conmutador)

Formato de mandato Conmutadores de seguridad

Tabla 353. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
PROCESO DE MQSECSW	1	X'00000001'
MQSECSW_NAMELIST	2	X'00000002'

Tabla 353. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSECSW_Q	3	X'00000003'
MQSECSW_TOPIC	4	X'00000004'
CONTEXTO_MQSECSW_CONTEXTO	6	X'00000006'
MQSECSW_ALTERNATE_USER	7	X'00000007'
MQSECSW_COMMAND	8	X'00000008'
MQSECSW_CONNECTION	9	X'00000009'
MQSECSW_SUBSYSTEM	10	X'0000000A'
MQSECSW_COMMAND_RESOURCES	11	X'0000000B'
MQSECSW_Q_MGR	15	X'0000000F'
MQSECSW_QSG	16	X'00000010'

Formato de mandato Estado de conmutador de seguridad

Tabla 354. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSECSW_OFF_FOUND	21	X'00000015'
MQSECSW_ON_FOUND	22	X'00000016'
MQSECSW_OFF_NOT_FOUND	23	X'00000017'
MQSECSW_ON_NOT_FOUND	24	X'00000018'
MQSECSW_OFF_ERROR	25	X'00000019'
MQSECSW_ON_OVERRIDDEN	26	X'0000001A'

MQSECTYPE_ * (Formato de mandato, Tipos de seguridad)

Tabla 355. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSECTYPE_AUTHSERV	1	X'00000001'
MQSECTYPE_SSL	2	X'00000002'
MQSECTYPE_CLASSES	3	X'00000003'

MQSEG_ * (Segmentación)

Tabla 356. Nombres y valores de constantes	
Nombre	**Valor**
MQSEG_INHIBIDO	'¬'
MQSEG_PERMITIDO	'A'

Nota: El símbolo ¬ representa un único carácter en blanco.

MQSEL_ * (Valores de selector especiales)

Tabla 357. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSEL_ANY_SELECTOR	-30001	X'FFFF8ACF'

Tabla 357. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
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_ * (Tipos de selector)

Tabla 358. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSELTYPE_NONE	0	X'00000000'
MQSELTYPE_STANDARD	1	X'00000001'
MQSELTYPE_EXTENDED	2	X'00000002'

MQSID_ * (Identificador de seguridad)

Tabla 359. Nombres y valores de constantes	
Nombre	**Valor**
MQSID_NONE	X'00...00' (40 nulos)
MQSID_NONE_ARRAY	'\0','\0',... (40 nulos)

MQSIDT_ * (Tipos de identificador de seguridad)

Tabla 360. Nombres y valores de constantes	
Nombre	**Valor hexadecimal**
MQSIDT_NONE	X'00'
MQSIDT_NT_SECURITY_ID	X'01'
MQSIDT_WAS_SECURITY_ID	X'02'

MQSMPO_ * (Establecer opciones y estructura de propiedad de mensaje)

Establecer estructura de opciones de propiedad de mensaje

Tabla 361. Estructuras de constantes	
Nombre	**Estructura**
MQSMPO_STRUC_ID	"SMPO"
MQSMPO_STRUC_ID_ARRAY	'S','M','P','O'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 362. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSMPO_VERSION_1	1	X'00000001'
MQSMPO_VERSIÓN_ACTUAL	1	X'00000001'

Establecer opciones de propiedad de mensaje

Tabla 363. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
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_ * (Opciones de suscripción)

Tabla 364. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSO_NONE	0	X'00000000'
MQSO_NON_DURABLE	0	X'00000000'
MQSO_READ_AHEAD_AS_Q_DEF	0	X'00000000'
MQSO_ALTER	1	X'00000001'
MQSO_CREATE	2	X'00000002'
MQSO_RESUME	4	X'00000004'
MQSO_DURABLE	8	X'00000008'
SUB_GRUPO_MQSO	16	X'00000010'
MQSO_GESTIONADO	32	X'00000020'
MQSO_SET_IDENTITY_CONTEXT	64	X'00000040'
MQSO_FIXED_USERID	256	X'00000100'
MQSO_ANY_USERID	512	X'00000200'
MQSO_PUBLICATIONS_ON_REQUEST	2048	X'00000800'
MQSO_NEW_PUBLICATIONS_ONLY	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'
MQSO_SET_CORREL_ID	4194304	X'00400000'
MQSO_SCOPE_QMGR	67108864	X'04000000'
MQSO_NO_READ_AHEAD	134217728	X'08000000'
MQSO_READ_AHEAD	268435456	X'10000000'

MQSP_ * (Disponibilidad de punto de sincronización)

Tabla 365. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSP_AVAILABLE	1	X'00000001'
MQSP_NOT_AVAILABLE	0	X'00000000'

MQSQQM_ * (Nombre de gestor de colas compartido)

Tabla 366. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSQQM_USE	0	X'00000000'
MQSQQM_IGNORE	1	X'00000001'

MQSR_ * (Acción)

Tabla 367. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSR_ACTION_PUBLICATION	1	X'00000001'

MQSRO_ * (estructura de opciones de solicitud de suscripción)

Tabla 368. Estructuras de constantes

Nombre	Estructura
MQSRO_STRUC_ID	"SRO¬"
MQSRO_STRUC_ID_ARRAY	'S','R','O','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 369. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSRO_VERSION_1	1	X'00000001'
MQSRO_CURRENT_VERSION	1	X'00000001'
MQSRO_NONE	0	X'00000000'
MQSRO_FAIL_IF_QUIESCING	8192	X'00002000'

MQSS_ * (Estado de segmento)

Tabla 370. Nombres y estructuras de constantes

Nombre	Estructura
MQSS_NO_A_SEGMENTO	'¬'
SEGMENTO_MQSS	'S'
MQSS_LAST_SEGMENT	'L'

Nota: El símbolo ¬ representa un único carácter en blanco.

MQSSL_ * (Requisitos de SSL FIPS)

Tabla 371. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSSL_FIPS_NO	0	X'00000000'
MQSSL_FIPS_YES	1	X'00000001'

MQSTAT_ * (Opciones de estadísticas)

MQSTAT_TYPE_ASYNC_ERROR	0	X'00000000'

MQSTAT_TYPE_RECONNECTION	0	X'00000000'
MQSTAT_TYPE_RECONNECTION_ERROR	0	X'00000000'

MQSTS_ * (Estructura de la estructura de informes de estado)

Tabla 372. Estructuras de constantes

Nombre	Estructura
MQSTS_ID_STRUCT	"STAT"
MQSTS_MATRIZ_ID_ESTRUCTURA	'S','T','A','T'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 373. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSTS_VERSION_1	1	X'00000001'
MQSTS_CURRENT_VERSION	1	X'00000001'

MQSUB_ * (Suscripciones duraderas)

Suscripciones duraderas

Tabla 374. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSUB_DURABLE_AS_PARENT	0	X'00000000'
MQSUB_DURABLE_ALLOWED	1	X'00000001'
MQSUB_DURABLE_INHIBIDO	2	X'00000002'

Suscripciones duraderas

Tabla 375. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSUB_DURABLE_ALL	-1	X'FFFFFFFF'
MQSUB_DURABLE_YES	1	X'00000001'
MQSUB_DURABLE_NO	2	X'00000002'

MQSUBTYPE_ * (Formato de mandato, Tipos de suscripción)

Tabla 376. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSUBTYPE_API	1	X'00000001'
MQSUBTYPE_ADMIN	2	X'00000002'
PROXY MQSUBTYPE_PROXY	3	X'00000003'
MQSUBTYPE_ALL	-1	X'FFFFFFFF'
MQSUBTYPE_USER	-2	X'FFFFFFFE'

MQSUS_ * (Formato de mandato, Estado de suspensión)

Tabla 377. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSUS_SÍ	1	X'00000001'
MQSUS_NO	0	X'00000000'

MQSVC_ * (Servicio)

Tipos de servicio

Tabla 378. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSVC_TYPE_COMMAND	0	X'00000000'
MQSVC_TYPE_SERVER	1	X'00000001'

Controles de servicio

Tabla 379. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSVC_CONTROL_Q_MGR	0	X'00000000'
MQSVC_CONTROL_Q_MGR_START	1	X'00000001'
MQSVC_CONTROL_MANUAL	2	X'00000002'

Estado del servicio

Tabla 380. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSVC_STATUS_STOPPED	0	X'00000000'
MQSVC_STATUS_STARTING	1	X'00000001'
MQSVC_STATUS_RUNNING	2	X'00000002'
MQSVC_STATUS_DETENIÉNDOSE	3	X'00000003'
MQSVC_STATUS_REINTENTANDO	4	X'00000004'

MQSYNCPOINT_ * (Formato de mandato, Valores de punto de sincronismo para migración de Pub/Sub)

Tabla 381. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSYNCPOINT_YES	0	X'00000000'
MQSYNCPOINT_IFPER	1	X'00000001'

MQSYSP_ * (Formato de mandato, Valores de parámetros del sistema)

Tabla 382. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQSYSP_NO	0	X'00000000'

Tabla 382. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQSYSP_SÍ	1	X'00000001'
MQSYSP_AMPLIADO	2	X'00000002'
MQSYSP_TIPO_INICIAL	10	X'0000000A'
MQSYSP_TYPE_SET	11	X'0000000B'
MQSYSP_TYPE_LOG_COPY	12	X'0000000C'
MQSYSP_TYPE_LOG_STATUS	13	X'0000000D'
MQSYSP_TYPE_ARCHIVE_TAPE	14	X'0000000E'
MQSYSP_ALLOC_BLK	20	X'00000014'
MQSYSP_ALLOC_TRK	21	X'00000015'
MQSYSP_ALLOC_CYL	22	X'00000016'
MQSYSP_STATUS_OCUPADO	30	X'0000001E'
MQSYSP_STATUS_PREMOUNT	31	X'0000001F'
MQSYSP_STATUS_AVAILABLE	32	X'00000020'
MQSYSP_STATUS_UNKNOWN	33	X'00000021'
MQSYSP_STATUS_ALLOC_ARCHIVE	34	X'00000022'
MQSYSP_STATUS_COPYING_BSDS	35	X'00000023'
MQSYSP_STATUS_COPYING_LOG	36	X'00000024'

MQTA_ * (Atributos de tema)

Comodines

Tabla 383. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTA_BLOCK	1	X'00000001'
MQTA_PASSTHRU	2	X'00000002'

Suscripciones permitidas

Tabla 384. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTA_SUB_AS_PARENT	0	X'00000000'
MQTA_SUB_XX_ENCODE_CASE_ONE inhibida	1	X'00000001'
MQTA_SUB_ALLOWED	2	X'00000002'

Subpropagación de proxy

Tabla 385. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTA_PROXY_SUB_FORCE	1	X'00000001'
MQTA_PROXY_SUB_FIRSTUSE	2	X'00000002'

Publicaciones permitidas

Tabla 386. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQTA_PUB_AS_PARENT	0	X'00000000'
MQTA_PUB_XX_ENCODE_CASE_ONE inhibida	1	X'00000001'
MQTA_PUB_ALLOWED	2	X'00000002'

MQTC_ * (Controles desencadenantes)

Tabla 387. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQTC_OFF	0	X'00000000'
MQTC_ON	1	X'00000001'

MQTCPKEEP_ * (TCP Keepalive)

Tabla 388. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQTCPKEEP_NO	0	X'00000000'
MQTCPKEEP_YES	1	X'00000001'

MQTCPSTACK_ * (Tipos de pila TCP)

Tabla 389. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQTCPSTACK_SINGLE	0	X'00000000'
MQTCPSTACK_MULTIPLE	1	X'00000001'

MQTIME_ * (Formato de mandato, Unidades de tiempo)

Tabla 390. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQTIME_UNIT_MINS	0	X'00000000'
MQTIME_UNIT_SECS	1	X'00000001'

MQTM_ * (Desencadenar estructura de mensaje)

Tabla 391. Estructuras de constantes

Nombre	Estructura
MQTM_STRUC_ID	"TM¬¬"
MQTM_STRUC_ID_ARRAY	'T','M','¬','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 392. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQTM_VERSION_1	1	X'00000001'
MQTM_CURRENT_VERSION	1	X'00000001'

MQTMC_ * (estructura de formato de caracteres de mensaje desencadenante)

Tabla 393. Estructuras de constantes	
Nombre	**Estructura**
MQTMC_STRUC_ID	"TMC¬"
MQTMC_STRUC_ID_ARRAY	'T','M','C','¬'
MQTMC_VERSION_1	"¬¬¬1"
MQTMC_VERSION_2	"¬¬¬2"
MQTMC_CURRENT_VERSION	"¬¬¬2"
MQTMC_VERSION_1_ARRAY	'¬','¬','¬','1'
MQTMC_VERSION_2_ARRAY	'¬','¬','¬','2'
MQTMC_CURRENT_VERSION_ARRAY	'¬','¬','¬','2'

Nota: El símbolo ¬ representa un único carácter en blanco.

MQTOPT_ * (Tipo de tema)

Tabla 394. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTOPT_LOCAL	0	X'00000000'
MQTOPT_CLUSTER	1	X'00000001'
MQTOPT_ALL	2	X'00000002'

MQTRAXSTR_ * (Inicio automático de rastreo de iniciador de canal)

Tabla 395. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTRAXSTR_NO	0	X'00000000'
MQTRAXSTR_SÍ	1	X'00000001'

MQTSCOPE_ * (Ámbito de suscripción)

Tabla 396. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTSCOPE_QMGR	1	X'00000001'
MQTSCOPE_ALL	2	X'00000002'

MQTT_ * (Tipos de desencadenante)

Tabla 397. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQTT_NONE	0	X'00000000'
MQTT_FIRST	1	X'00000001'
MQTT_EVERY	2	X'00000002'
MQTT_DEPTH	3	X'00000003'

MQTYPE_ * (Tipos de datos de propiedad)

Tabla 398. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQTYPE_AS_SET	0	X'00000000'
MQTYPE_NULL	2	X'00000002'
MQTYPE_BOOLEAN	4	X'00000004'
MQTYPE_BYTE_STRING	8	X'00000008'
MQTYPE_INT8	16	X'00000010'
MQTYPE_INT16	32	X'00000020'
MQTYPE_INT32	64	X'00000040'
MQTYPE_LONG	64	X'00000040'
MQTYPE_INT64	128	X'00000080'
MQTYPE_FLOAT32	256	X'00000100'
MQTYPE_FLOAT64	512	X'00000200'
MQTYPE_STRING	1024	X'00000400'

MQUA_ * (Selectores de atributos de usuario de publicación/suscripción)

Tabla 399. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQUA_PRIMERO	65536	X'00010000'
MQUA_LAST	999999999	X'3B9AC9FF'

MQUIDSUPP_ * (Formato de mandato, Soporte de ID de usuario)

Tabla 400. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQUIDSUPP_NO	0	X'00000000'
MQUIDSUPP_SÍ	1	X'00000001'

MQUNDELIVERED_ * (Formato de mandato, Valores no entregados para la migración de Pub/Sub)

Tabla 401. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQUNDELIVERED_NORMAL	0	X'00000000'
MQUNDELIVERED_SAFE	1	X'00000001'
MQUNDELIVERED_DISCARD	2	X'00000002'
MQUNDELIVERED_KEEP	3	X'00000003'

MQUOWST_ * (Formato de mandato, Estados de UOW)

Tabla 402. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQUOWST_NONE	0	X'00000000'

Tabla 402. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQUOWST_ACTIVE	1	X'00000001'
MQUOWST_PREPARADO	2	X'00000002'
MQUOWST_SIN resolver	3	X'00000003'

MQUOWT_ * (Formato de mandato, Tipos de UOW)

Tabla 403. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQUOWT_Q_MGR	0	X'00000000'
MQUOWT_CICS	1	X'00000001'
MQUOWT_RRS	2	X'00000002'
MQUOWT_IMS	3	X'00000003'
MQUOWT_XA	4	X'00000004'

MQUS_ * (Usos de cola)

Tabla 404. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQUS_NORMAL	0	X'00000000'
MQUS_TRANSMISSION	1	X'00000001'

MQUSAGE_ * (Formato de mandato, Valores de uso de conjunto de páginas y Valores de uso de conjunto de datos)

Formato de mandato Valores de uso de conjunto de páginas

Tabla 405. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQUSAGE_PS_AVAILABLE	0	X'00000000'
MQUSAGE_PS_DEFINED	1	X'00000001'
MQUSAGE_PS_OFFLINE	2	X'00000002'
MQUSAGE_PS_NOT_DEFINED	3	X'00000003'
MQUSAGE_EXPAND_USER	1	X'00000001'
MQUSAGE_EXPAND_SISTEMA	2	X'00000002'
MQUSAGE_EXPAND_NONE	3	X'00000003'

Formato de mandato Valores de uso de conjunto de datos

Tabla 406. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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_ * (Longitud de valor)

Tabla 407. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQVL_NULL_TERMINATED	-1	X'FFFFFFFF'
MQVL_EMPTY_STRING	0	X'00000000'

MQVU_ * (ID de usuario variable)

Tabla 408. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
USUARIO_ARREGLADO_MQVU_USUARIO	1	X'00000001'
MQVU_ANY_USER	2	X'00000002'

MQWDR_ * (Estructura de registro de destino de salida de carga de trabajo de clúster)

Tabla 409. Estructuras de constantes

Nombre	Estructura
MQWDR_STRUC_ID	"WDR¬"
MQWDR_STRUC_ID_ARRAY	'W','D','R','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 410. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQWDR_VERSION_1	1	X'00000001'
MQWDR_VERSION_2	2	X'00000002'
MQWDR_CURRENT_VERSION	2	X'00000002'
MQWDR_LENGTH_1	124	X'0000007C'
MQWDR_LENGTH_2	136	X'00000088'
MQWDR_CURRENT_LENGTH	136	X'00000088'

MQWI_ * (Intervalo de espera)

Tabla 411. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQWI_UNLIMITED	-1	X'FFFFFFFF'

MQWIH_ * (Estructura de cabecera de información de carga de trabajo y distintivos)

Estructura de cabecera de información de carga de trabajo

Tabla 412. Estructuras de constantes

Nombre	Estructura
MQWIH_STRUC_ID	"WIH¬"
MQWIH_STRUC_ID_ARRAY	'W','I','H','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 413. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQWIH_VERSION_1	1	X'00000001'
MQWIH_CURRENT_VERSION	1	X'00000001'
MQWIH_LENGTH_1	120	X'00000078'
MQWIH_CURRENT_LENGTH	120	X'00000078'

Distintivos de cabecera de información de carga de trabajo

Tabla 414. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQWIH_NONE	0	X'00000000'

MQWQR_ * (Estructura de registro de cola de salida de carga de trabajo de clúster)

Tabla 415. Estructuras de constantes	
Nombre	**Estructura**
MQWQR_STRUC_ID	"WQR¬"
MQWQR_STRUC_ID_ARRAY	'W','Q','R','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 416. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQWQR_VERSION_1	1	X'00000001'
MQWQR_VERSION_2	2	X'00000002'
MQWQR_VERSION_3	3	X'00000003'
MQWQR_CURRENT_VERSION	3	X'00000003'
MQWQR_LENGTH_1	200	X'000000C8'
MQWQR_LENGTH_2	208	X'000000D0'
MQWQR_LENGTH_3	212	X'000000D4'
MQWQR_CURRENT_LENGTH	212	X'000000D4'

MQWS_ * (Esquema comodín)

Tabla 417. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQWS_PREDETERMINADO	0	X'00000000'
MQWS_CHAR	1	X'00000001'
MQWS_TOPIC	2	X'00000002'

MQWXP_ * (Estructura de parámetros de salida de carga de trabajo de clúster)

MQWXP_ * (Estructura de parámetros de salida de carga de trabajo de clúster)

Tabla 418. Estructuras de constantes	
Nombre	**Estructura**
MQWXP_STRUC_ID	"WXP¬"
MQWXP_STRUC_ID_ARRAY	'W','X','P','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 419. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQWXP_VERSION_1	1	X'00000001'
MQWXP_VERSION_2	2	X'00000002'
MQWXP_VERSION_3	3	X'00000003'
MQWXP_VERSION_4	4	X'00000004'
MQWXP_CURRENT_VERSION	4	X'00000004'

MQWXP_ * (distintivos de carga de trabajo de clúster)

Tabla 420. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQWXP_PUT_BY_CLUSTER_CHL	2	X'00000002'

Referencia relacionada

Campos en MQWXP-Estructura de parámetros de salida de carga de trabajo de clúster

MQXACT_ * (Tipos de interlocutor de API)

Tabla 421. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXACT_EXTERNAL	1	X'00000001'
MQXACT_INTERNAL	2	X'00000002'

MQXC_ * (mandatos de salida)

Tabla 422. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXC_MQOPEN	1	X'00000001'
MQXC_MQCLOSE	2	X'00000002'
MQXC_MQGET	3	X'00000003'
MQXC_MQPUT	4	X'00000004'
MQXC_MQPUT1	5	X'00000005'
MQXC_MQINQ	6	X'00000006'
MQXC_MQSET	8	X'00000008'
MQXC_MQBACK	9	X'00000009'

Tabla 422. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXC_MQCMIT	10	X'0000000A'

MQXCC_ * (Respuestas de salida)

Tabla 423. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXCC_Correcto	0	X'00000000'
MQXCC_SUPPRESS_FUNCTION	-1	X'FFFFFFFF'
MQXCC_SKIP_FUNCTION	-2	X'FFFFFFFE'
MQXCC_SEND_AND_REQUEST_SEC_MSG	-3	X'FFFFFFFD'
MQXCC_SEND_SEC_MSG	-4	X'FFFFFFFC'
MQXCC_SUPPRESS_EXIT	-5	X'FFFFFFFB'
MQXCC_CLOSE_CHANNEL	-6	X'FFFFFFFA'
MQXCC_REQUEST_ACK	-7	X'FFFFFFF9'
MQXCC_FAILED	-8	X'FFFFFFF8'

MQXDR_ * (Respuesta de salida)

Tabla 424. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXDR_Correcto	0	X'00000000'
MQXDR_CONVERSION_FAILED	1	X'00000001'

MQXE_ * (Entornos)

Tabla 425. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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'

MQXEPO_ * (estructura de opciones de punto de entrada de registro y opciones de salida)

Estructura de opciones de punto de entrada de registro

Tabla 426. Estructuras de constantes	
Nombre	**Estructura**
MQXEPO_ID_ESTRUCTURA	"XEPO"
MQXEPO_STRUC_ID_ARRAY	'X','E','P','O'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 427. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXEPO_VERSION_1	1	X'00000001'
MQXEPO_VERSIÓN_ACTUAL	1	X'00000001'

Opciones de salida

Tabla 428. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXEPO_NONE	0	X'00000000'

MQXF_ * (Identificadores de función de API)

Tabla 429. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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_INICIO	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'

Tabla 429. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
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'

MQXP_ * (estructura de parámetros de salida cruzada de API)

Tabla 430. Estructuras de constantes	
Nombre	**Estructura**
MQXP_STRUC_ID	"XP¬¬"
MQXP_STRUC_ID_ARRAY	'X','P','¬','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 431. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXP_VERSION_1	1	X'00000001'

MQXPDA_ * (Área de determinación de problemas)

Tabla 432. Nombres y valores de constantes	
Nombre	**Valor**
MQXPDA_NONE	X'00...00' (48 nulos)
MQXPDA_NONE_ARRAY	'\0','\0',... (48 nulos)

MQXPT_ * (Tipos de transporte)

Tabla 433. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXPT_TODOS	-1	X'FFFFFFFF'
MQXPT_LOCAL	0	X'00000000'
MQXPT_LU62	1	X'00000001'
MQXPT_TCP	2	X'00000002'
MQXPT_NETBIOS	3	X'00000003'
MQXPT_SPX	4	X'00000004'
MQXPT_DECNET	5	X'00000005'
MQXPT_UDP	6	X'00000006'

MQXQH_ * (Estructura de cabecera de cola de transmisión)

Tabla 434. Estructuras de constantes	
Nombre	**Estructura**
MQXQH_STRUC_ID	"XQH¬"

Tabla 434. Estructuras de constantes (continuación)	
Nombre	**Estructura**
MQXQH_STRUC_ID_ARRAY	'X','Q','H','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 435. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXQH_VERSION_1	1	X'00000001'
MQXQH_CURRENT_VERSION	1	X'00000001'

MQXR_ * (Razones de salida)

Tabla 436. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXR_ANTES	1	X'00000001'
MQXR_AFTER	2	X'00000002'
MQXR_CONNECTION	3	X'00000003'
MQXR_INIT	11	X'0000000B'
MQXR_TERM	12	X'0000000C'
MQXR_MSG	13	X'0000000D'
MQXR_XMIT	14	X'0000000E'
MQXR_SEC_MSG	15	X'0000000F'
MQXR_INIT_SEC	16	X'00000010'
MQXR_REINTENTAR	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'
MQXR_CLWL_REPOS	23	X'00000017'
MQXR_CLWL_REPOS_MOVE	24	X'00000018'
MQXR_END_BATCH	25	X'00000019'
MQXR_ACK_RECEIVED	26	X'0000001A'
MQXR_AUTO_SVRCONN	27	X'0000001B'
MQXR_AUTO_CLUSRCVR	28	X'0000001C'
MQXR_SEC_PARMS	29	X'0000001D'

MQXR2_* (Respuesta de salida 2)

Tabla 437. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXR2_PUT_WITH_DEF_ACTION	0	X'00000000'
MQXR2_PUT_WITH_DEF_USERID	1	X'00000001'
MQXR2_PUT_WITH_MSG_USERID	2	X'00000002'

Tabla 437. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
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_ * (Identificadores de salida)

Tabla 438. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
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_ * (Valor de área de usuario de salida)

Tabla 439. Nombres y valores de constantes	
Nombre	**Valor**
MQXUA_NONE	X'00...00' (16 nulos)
MQXUA_NOE_MATRIZ	'\0','\0',... (16 nulos)

MQXWD_ * (estructura del descriptor de espera de salida)

Tabla 440. Estructuras de constantes	
Nombre	**Estructura**
MQXWD_ID_STRUCD	"XWD¬"
MQXWD_STRUC_ID_ARRAY	'X','W','D','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 441. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQXWD_VERSION_1	1	X'00000001'

MQZAC_ * (Estructura de contexto de aplicación)

Tabla 442. Estructuras de constantes	
Nombre	**Estructura**
MQZAC_STRUC_ID	"ZAC¬"
MQZAC_STRUC_ID_ARRAY	'Z','A','C','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 443. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZAC_VERSION_1	1	X'00000001'
MQZAC_CURRENT_VERSION	1	X'00000001'

MQZAD_ * (estructura de datos de autorización)

Tabla 444. Estructuras de constantes	
Nombre	**Estructura**
MQZAD_STRUC_ID	"ZAD¬"
MQZAD_STRUC_ID_ARRAY	'Z','A','D','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 445. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZAD_VERSION_1	1	X'00000001'
MQZAD_VERSION_2	2	X'00000002'
MQZAD_CURRENT_VERSION	2	X'00000002'

MQZAET_ * (Tipos de entidad de servicios instalables)

Tabla 446. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZAET_NONE	0	X'00000000'
MQZAET_PRINCIPAL	1	X'00000001'
GRUPO_MQZAC	2	X'00000002'
MQZAET_DESCONOCIDO	3	X'00000003'

MQZAO_ * (Autorizaciones de servicios instalables)

Tabla 447. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZAO_CONNECT	1	X'00000001'
MQZAO_BROWSE	2	X'00000002'
MQZAO_INPUT	4	X'00000004'
MQZAO_OUTPUT	8	X'00000008'
MQZAO_INQUIRE	16	X'00000010'
MQZAO_SET	32	X'00000020'

Tabla 447. Valores de constantes (continuación)

Nombre	Valor decimal	Valor hexadecimal
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_PUBLISH	2048	X'00000800'
MQZAO_SUBSCRIBE	4096	X'00001000'
MQZAO_RESUME	8192	X'00002000'
MQZAO_ALL_MQI	16383	X'00003FFF'
MQZAO_CREAR	65536	X'00010000'
MQZAO_DELETE	131072	X'00020000'
MQZAO_DISPLAY	262144	X'00040000'
MQZAO_CHANGE	524288	X'00080000'
MQZAO_CLEAR	1048576	X'00100000'
MQZAO_CONTROL	2097152	X'00200000'
MQZAO_CONTROL_EXTENDED	4194304	X'00400000'
MQZAO_AUTORIZAR	8388608	X'00800000'
MQZAO_ALL_ADMIN	16646144	X'00FE0000'
MQZAO_TODOS	16662527	X'00FE3FFF'
MQZAO_ELIMINAR	16777216	X'01000000'
MQZAO_NONE	0	X'00000000'

MQZAS_ * (versión de interfaz de servicio de servicios instalables)

Tabla 448. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
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_ * (Tipos de autenticación)

Tabla 449. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQZAT_INITIAL_CONTEXT	0	X'00000000'
CONTEXTO_CAMBIO_MQZAT_CONTEXTO	1	X'00000001'

MQZCI_ * (Indicador de continuación de servicios instalables)

Tabla 450. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZCI_DEFAULT	0	X'00000000'
MQZCI_CONTINUE	0	X'00000000'
MQZCI_STOP	1	X'00000001'

MQZED_ * (Estructura de datos de entidad)

Tabla 451. Estructuras de constantes	
Nombre	**Estructura**
MQZED_STRUC_ID	"ZED¬"
MQZED_STRUC_ID_ARRAY	'Z','E','D','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 452. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZED_VERSION_1	1	X'00000001'
MQZED_VERSION_2	2	X'00000002'
MQZED_CURRENT_VERSION	2	X'00000002'

MQZFP_ * (estructura de parámetros libres)

Tabla 453. Estructuras de constantes	
Nombre	**Estructura**
MQZFP_STRUC_ID	"ZFP¬"
MQZFP_STRUC_ID_ARRAY	'Z','F','P','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 454. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZFP_VERSION_1	1	X'00000001'
MQZFP_CURRENT_VERSION	1	X'00000001'

MQZIC_ * (Estructura de contexto de identidad)

Tabla 455. Estructuras de constantes	
Nombre	**Estructura**
MQZIC_STRUC_ID	"ZIC¬"
MQZIC_STRUC_ID_ARRAY	'Z','I','C','¬'

Nota: El símbolo ¬ representa un único carácter en blanco.

Tabla 456. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZIC_VERSION_1	1	X'00000001'

Tabla 456. Valores de constantes (continuación)		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZIC_CURRENT_VERSION	1	X'00000001'

MQZID_ * (ID de función para servicios)

ID de función comunes a todos los servicios

Tabla 457. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZID_INIT	0	X'00000000'
MQZID_TERM	1	X'00000001'

ID de función para el servicio de autorización

Tabla 458. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZID_INIT_AUTHORITY	0	X'00000000'
MQZID_TERM_AUTHORITY	1	X'00000001'
MQZID_CHECK_AUTHORITY	2	X'00000002'
MQZID_COPIA_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_REFRESH_CACHE	8	X'00000008'
MQZID_ENUMERATE_AUTORITY_DATA	9	X'00000009'
MQZID_AUTOENTICATE_USER	10	X'0000000A'
MQZID_FREE_USER	11	X'0000000B'
MQZID_INQUIRE	12	X'0000000C'
MQZID_CHECK_PRIVILEGED	13	X'0000000D'

ID de función para el servicio de nombres

Tabla 459. Valores de constantes		
Nombre	**Valor decimal**	**Valor hexadecimal**
MQZID_INIT_NAME	0	X'00000000'
MQZID_XX_ENCODE_CASE_ONE nombre_term	1	X'00000001'
MQZID_LOOKUP_NAME	2	X'00000002'
MQZID_INSERT_NAME	3	X'00000003'
MQZID_DELETE_NAME	4	X'00000004'

ID de función para el servicio de ID de usuario

Tabla 460. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQZID_INIT_USERID	0	X'00000000'
MQZID_TERM_USERID	1	X'00000001'
MQZID_FIND_USERID	2	X'00000002'

MQZIO_ * (Opciones de inicialización de servicios instalables)

Tabla 461. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQZIO_PRIMARY	0	X'00000000'
MQZIO_SECUNDARIO	1	X'00000001'

MQZNS_ * (Versión de interfaz de servicio de nombres)

Tabla 462. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQZNS_VERSION_1	1	X'00000001'

MQZSE_ * (Indicador de enumeración de inicio de servicios instalables)

Tabla 463. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQZSE_START	1	X'00000001'
MQZSE_CONTINUE	0	X'00000000'

MQZSL_ * (Indicador de selector de servicios instalables)

Tabla 464. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQZSL_NO_DEVUELTO	0	X'00000000'
MQZSL_DEVUELTO	1	X'00000001'

MQZTO_ * (Opciones de terminación de servicios instalables)

Tabla 465. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQZTO_PRIMARY	0	X'00000000'
MQZTO_SECUNDARIO	1	X'00000001'

MQZUS_ * (Versión de interfaz de servicio de ID de usuario)

Tabla 466. Valores de constantes

Nombre	Valor decimal	Valor hexadecimal
MQZUS_VERSION_1	1	X'00000001'

Tipos de datos utilizados en la MQI

Información sobre los tipos de datos que se pueden utilizar en MQI. Descripciones, campos y declaraciones de idioma para los idiomas relevantes con cada tipo de datos.

Introducción de tipos de datos utilizados en la MQI

En esta sección se presentan los tipos de datos utilizados en la MQI y se proporcionan algunas instrucciones sobre cómo utilizarlos en los lenguajes de programación soportados.

Tipos de datos elementales

Esta sección contiene información sobre los tipos de datos utilizados en la MQI (o en las funciones de salida). Estos se describen en detalle, seguidos de ejemplos que muestran cómo declarar los tipos de datos elementales en los lenguajes de programación soportados en los temas siguientes.

Los tipos de datos utilizados en la MQI (o en las funciones de salida) son:

- Tipos de datos elementales, o
- Agregados de tipos de datos elementales (matrices o estructuras)

Los siguientes tipos de datos elementales se utilizan en la MQI (o en las funciones de salida):

Nombre de tipo de datos elemental	**Tipo de datos**	**Descripción**
MQBOOL	Boolean	El tipo de datos MQBOOL representa un valor booleano. El valor 0 representa false. Cualquier otro valor representa true. Un MQBOOL debe estar alineado como para el tipo de datos MQLONG.

Nombre de tipo de datos elemental	Tipo de datos	Descripción
MQBYTE	Byte	El tipo de datos MQBYTE representa un único byte de datos. No se coloca ninguna interpretación particular en el byte; se trata como una serie de bits, y no como un número binario o carácter. No es necesaria ninguna alineación especial. Cuando se envían datos MQBYTE entre gestores de colas que utilizan conjuntos de caracteres o codificaciones diferentes, los datos MQBYTE *no* se convierten de ninguna forma. Los campos *MsgId* y *CorrelId* de la estructura MQMD son los siguientes. A veces se utiliza una matriz de MQBYTE para representar un área de almacenamiento principal que el gestor de colas no conoce. Por ejemplo, el área puede contener datos de mensaje de aplicación o una estructura. La alineación de los límites de esta zona debe ser compatible con la naturaleza de los datos contenidos en ella. En el lenguaje de programación C, se puede utilizar cualquier tipo de datos para los parámetros de función que se muestran como matrices de MQBYTE. Esto se debe a que estos parámetros siempre se pasan por dirección, y en C el parámetro de función se declara como puntero a vacío.

Nombre de tipo de datos elemental	Tipo de datos	Descripción
MQBYTEn	Serie de *n* bytes	Cada tipo de datos MQBYTEn representa una serie de *n* bytes, donde *n* puede tomar cualquiera de los valores siguientes: 8, 16, 24, 32, 40 o 128. Cada byte se describe mediante el tipo de datos MQBYTE. No es necesaria ninguna alineación especial. Si los datos de la serie de bytes son más cortos que la longitud definida de la serie, los datos deben rellenarse con nulos para rellenar la serie. Cuando el gestor de colas devuelve series de bytes a la aplicación (por ejemplo, en la llamada MQGET), el gestor de colas rellena con nulos la longitud definida de la serie. Las constantes con nombre están disponibles para definir las longitudes de los campos de serie de bytes. Estos se listan en “Constantes” en la página 50
MQCHAR	Carácter	El tipo de datos MQCHAR representa un carácter de un solo byte o un byte de un carácter de doble byte o de varios bytes. No es necesaria ninguna alineación especial. Cuando se envían datos MQCHAR entre gestores de colas que utilizan diferentes conjuntos de caracteres o codificaciones, los datos MQCHAR normalmente requieren conversión para que los datos se interpreten correctamente. El gestor de colas lo hace automáticamente para los datos MQCHAR en la estructura MQMD. La conversión de datos MQCHAR en los datos del mensaje de aplicación se controla mediante la opción MQGMO_CONVERT especificada en la llamada MQGET; consulte la descripción de esta opción en “MQGMO-Opciones de obtención de mensajes” en la página 344 para obtener más detalles.

Nombre de tipo de datos elemental	Tipo de datos	Descripción
MQCHARn	Serie de *n* caracteres	Cada tipo de datos MQCHARn representa una serie de *n* caracteres, donde *n* puede tomar cualquiera de los valores siguientes: 4, 8, 12, 20, 28, 32, 48, 64, 128 o 256. Cada carácter se describe mediante el tipo de datos MQCHAR. No es necesaria ninguna alineación especial. Si los datos de la serie son más cortos que la longitud definida de la serie, los datos deben rellenarse con espacios en blanco para rellenar la serie. En algunos casos, se puede utilizar un carácter nulo para finalizar la serie de forma prematura, en lugar de rellenarla con espacios en blanco; el carácter nulo y los caracteres que le siguen se tratan como espacios en blanco, hasta la longitud definida de la serie. Los lugares donde se puede utilizar un nulo se identifican en las descripciones de llamada y tipo de datos. Cuando el gestor de colas devuelve series de caracteres a la aplicación (por ejemplo, en la llamada MQGET), el gestor de colas siempre rellena con blancos la longitud definida de la serie; el gestor de colas no utiliza el carácter nulo para delimitar la serie. Las constantes con nombre están disponibles que definen las longitudes de los campos de serie de caracteres y se listan en “Constantes” en la página 50.

Nombre de tipo de datos elemental	Tipo de datos	Descripción
MQFLOAT32	Número de coma flotante de 32 bits	El tipo de datos MQFLOAT32 es un número de coma flotante de 32 bits representado utilizando el formato de coma flotante IEEE estándar. Un MQFLOAT32 debe estar alineado en un límite de 4 bytes. El uso de MQFLOAT32 en C en z/OS requiere el uso del distintivo de compilador FLOAT (IEEE). El uso de MQFLOAT32 en COBOL está limitado a compiladores que dan soporte a números de coma flotante en formato IEEE. Esto puede requerir el uso del distintivo de compilador FLOAT (NATIVE).
MQFLOAT64	Número de coma flotante de 64 bits	El tipo de datos MQFLOAT64 es un número de coma flotante de 64 bits representado utilizando el formato de coma flotante IEEE estándar. Un MQFLOAT64 debe estar alineado en un límite de 8 bytes. El uso de MQFLOAT64 en C en z/OS requiere el uso del distintivo de compilador FLOAT (IEEE). El uso de MQFLOAT64 en COBOL está limitado a compiladores que dan soporte a números de coma flotante en formato IEEE. Esto puede requerir el uso del distintivo de compilador FLOAT (NATIVE).

Nombre de tipo de datos elemental	Tipo de datos	Descripción
MQHCONFIG	Descriptor de contexto de configuración	El tipo de datos MQHCONFIG representa un descriptor de contexto de configuración, es decir, el componente que se está configurando para un servicio instalable determinado. Un descriptor de contexto de configuración debe estar alineado en su límite natural. Las aplicaciones no deben basarse en el formato de los datos almacenados dentro de este descriptor de contexto. Si es válido, su valor está pensado para ser utilizable en más llamadas MQI, pero no está pensado para tener ningún significado aparte de ese propósito.
MQHCONN	Descriptor de contexto de conexión	El tipo de datos MQHCONN representa un descriptor de conexión, es decir, la conexión con un gestor de colas determinado. Un descriptor de conexión debe estar alineado en un límite de 4 bytes. Las aplicaciones no deben basarse en el formato de los datos almacenados dentro de este descriptor de contexto. Si es válido, su valor está pensado para ser utilizable en más llamadas MQI, pero no está pensado para tener ningún significado aparte de ese propósito.

Nombre de tipo de datos elemental	Tipo de datos	Descripción
MQHMSG	Descriptor de contexto de mensaje	El tipo de datos MQHMSG representa un descriptor de mensaje que proporciona acceso a un mensaje. Un descriptor de mensaje debe estar alineado en un límite de 8 bytes. Las aplicaciones no deben basarse en el formato de los datos almacenados dentro de este descriptor de contexto. Si es válido, su valor está pensado para ser utilizable en más llamadas MQI, pero no está pensado para tener ningún significado aparte de ese propósito.
MQHOBJ	Descriptor de contexto del objeto	El tipo de datos MQHOBJ representa un descriptor de contexto de objeto que proporciona acceso a un objeto. Un descriptor de contexto de objeto debe estar alineado en un límite de 4 bytes. Las aplicaciones no deben basarse en el formato de los datos almacenados dentro de este descriptor de contexto. Si es válido, su valor está pensado para ser utilizable en más llamadas MQI, pero no está pensado para tener ningún significado aparte de ese propósito.
MQINT8	Entero con signo de 8 bits	El tipo de datos MQINT8 es un entero con signo de 8 bits que puede tomar cualquier valor en el rango de -128 a +127, a menos que el contexto restrinja lo contrario.
MQINT16	Entero con signo de 16 bits	El tipo de datos MQINT16 es un entero con signo de 16 bits que puede tomar cualquier valor en el rango de -32 768 a +32 767, a menos que el contexto restrinja lo contrario. Un MQINT16 debe estar alineado en un límite de 2 bytes.

Nombre de tipo de datos elemental	Tipo de datos	Descripción
MQINT32	Entero con signo de 32 bits	El tipo de datos MQINT32 es un entero binario con signo de 32 bits que puede tomar cualquier valor en el rango de -2 147 483 648 a + 2 147 483 647, a menos que el contexto restrinja lo contrario. Consulte la definición de MQLONG.
MQINT64	Entero con signo de 64 bits	El tipo de datos MQINT64 es un entero con signo de 64 bits que puede tomar cualquier valor en el rango de -9 223 372 036 854 775 808 a + 9 223 372 036 854 775 807, a menos que el contexto restrinja lo contrario. Para COBOL, el rango válido está limitado a -999 999 999 999 999 999 a +999 999 999 999 999 999. Un entero de 64 bits debe estar alineado en un límite de 8 bytes.
MQLONG	Entero con signo de 32 bits	El tipo de datos MQLONG es un entero binario con signo de 32 bits que puede tomar cualquier valor en el rango de -2 147 483 648 a + 2 147 483 647, a menos que el contexto restrinja lo contrario. Para COBOL, el rango válido está limitado a -999 999 999 a +999 999 999. Un MQLONG debe estar alineado en un límite de 4 bytes.
IDMQP	Identificador proceso	El identificador de proceso de WebSphere MQ . Es el mismo identificador que se utiliza en los volcados MQ trace y FFST™ , pero puede ser diferente del identificador de proceso del sistema operativo.

Nombre de tipo de datos elemental	Tipo de datos	Descripción
MQPTR	Puntero	El tipo de datos MQPTR es la dirección de los datos de cualquier tipo. Un puntero debe estar alineado en su límite natural; es un límite de 16 bytes en IBM i y un límite de 8 bytes en otras plataformas. Algunos lenguajes de programación dan soporte a punteros escritos; la MQI también los utiliza en algunos casos (por ejemplo, PMQCHAR y PMQLONG en el lenguaje de programación C).
MQTID	Identificador de hebra	Identificador de hebra de WebSphere MQ . Es el mismo identificador que se utiliza en los volcados MQ trace y FFST™ , pero puede ser diferente del identificador de hebra del sistema operativo.
MQUINT8	Entero sin signo de 8 bits	El tipo de datos MQUINT8 es un entero sin signo de 8 bits que puede tomar cualquier valor en el rango de 0 a +255, a menos que el contexto restrinja lo contrario.
MQUINT16	Entero sin signo de 16 bits	El tipo de datos MQUINT16 es un entero sin signo de 16 bits que puede tomar cualquier valor en el rango de 0 a +65 535, a menos que el contexto restrinja lo contrario. Un MQUINT16 debe estar alineado en un límite de 2 bytes.
MQUINT32	Entero de 32 bits sin signo	El tipo de datos MQUINT32 es un entero binario sin signo de 32 bits. Consulte la definición de MQULONG.

Nombre de tipo de datos elemental	Tipo de datos	Descripción
MQUINT64	Entero de 64 bits sin signo	El tipo de datos MQINT64 es un entero sin signo de 64 bits que puede tomar cualquier valor en el rango de 0 a +18 446 744 073 709 551 615, a menos que el contexto restrinja lo contrario. Para COBOL, el rango válido está limitado a 0 a +999 999 999 999 999 999. Un entero de 64 bits debe estar alineado en un límite de 8 bytes.
MQULONG	Entero de 32 bits sin signo	El tipo de datos MQULONG es un entero binario sin signo de 32 bits que puede tomar cualquier valor en el rango de 0 a + 4 294 967 294, a menos que el contexto restrinja lo contrario. Para COBOL, el rango válido está limitado a 0 a +999 999 999. Un MQULONG debe estar alineado en un límite de 4 bytes.
PMQACH	Puntero	Puntero a una estructura de datos de tipo MQACH
PMQAIR	Puntero	Puntero a una estructura de datos de tipo MQAIR
PMQAXC	Puntero	Puntero a una estructura de datos de tipo MQAXC
PMQAXP	Puntero	Puntero a una estructura de datos de tipo MQAXP
PMQBMHO	Puntero	Puntero a una estructura de datos de tipo MQBMHO
PMQBO	Puntero	Puntero a una estructura de datos de tipo MQBO
PMQBOOL	Puntero	Puntero a datos de tipo MQBOOL
PMQBYTE	Puntero	Puntero a datos de tipo MQBYTE
PMQBYTEn	Puntero	Puntero a datos de tipo MQBYTEn, donde n puede ser 8, 16, 24, 32, 40, 128
PMQCBC	Puntero	Puntero a una estructura de datos de tipo MQCBC
PMQCBD	Puntero	Puntero a una estructura de datos de tipo MQCBD
PMQCHAR	Puntero	Puntero a datos de tipo MQCHAR

Nombre de tipo de datos elemental	Tipo de datos	Descripción
PMQCHARN	Puntero	Puntero a un tipo de datos de MQCHARN, donde n puede ser 4, 8, 12, 20, 28, 32, 48, 64, 128, 256, 264
PMQCHARV	Puntero	Puntero a una estructura de datos de tipo MQCHARV
PMQCIH	Puntero	Puntero a una estructura de datos de tipo MQCIH
PMQCMHO	Puntero	Puntero a una estructura de datos de tipo MQCMHO
PMQCNO	Puntero	Puntero a una estructura de datos de tipo MQCNO
PMQCSP	Puntero	Puntero a una estructura de datos de tipo MQCSP
PMQCTLO	Puntero	Puntero a una estructura de datos de tipo MQCTLO
PMQDH	Puntero	Puntero a una estructura de datos de tipo MQDH
PMQDHO	Puntero	Puntero a una estructura de datos de tipo MQDHO
PMQDLH	Puntero	Puntero a una estructura de datos de tipo MQDLH
PMQDMHO	Puntero	Puntero a una estructura de datos de tipo MQDMHO
PMQDMPO	Puntero	Puntero a una estructura de datos de tipo MQDMPO
PMQEPH	Puntero	Puntero a una estructura de datos de tipo MQEPH
PMQFLOAT32	Puntero	Puntero a una estructura de datos de tipo MQFLOAT32
PMQFLOAT64	Puntero	Puntero a una estructura de datos de tipo MQFLOAT64
PMQFUNC	Puntero	Puntero a una función
PMQGMO	Puntero	Puntero a una estructura de datos de tipo MQGMO
PMQHCONFIG	Puntero	Puntero a datos de tipo MQHCONFIG
PMQHCONN	Puntero	Puntero a datos de tipo MQHCONN
PMQHMSG	Puntero	Puntero a datos de tipo MQHMSG
PMQHOBJ	Puntero	Puntero a datos de tipo MQHOBJ

Nombre de tipo de datos elemental	Tipo de datos	Descripción
PMQIIH	Puntero	Puntero a una estructura de datos de tipo MQIIH
PMQIMPO	Puntero	Puntero a una estructura de datos de tipo MQIMPO
PMQINT8	Puntero	Puntero a datos de tipo MQINT8
PMQINT16	Puntero	Puntero a datos de tipo MQINT16
PMQINT32	Puntero	Puntero a datos de tipo MQINT32
PMQINT64	Puntero	Puntero a datos de tipo MQINT64
PMQLONG	Puntero	Puntero a datos de tipo MQLONG
PMQMD	Puntero	Puntero a estructura de tipo MQMD
PMQMDE	Puntero	Puntero a una estructura de datos de tipo MQMDE
PMQMD1	Puntero	Puntero a una estructura de datos de tipo MQMD1
PMQMD2	Puntero	Puntero a una estructura de datos de tipo MQMD2
PMQMHBO	Puntero	Puntero a una estructura de datos de tipo MQMHBO
PMQOD	Puntero	Puntero a una estructura de datos de tipo MQOD
PMQOR	Puntero	Puntero a una estructura de datos de tipo MQOR
PMQPD	Puntero	Puntero a una estructura de datos de tipo MQPD
PMQPID	Puntero	Puntero a un identificador de proceso
PMQMD	Puntero	Puntero a una estructura de datos de tipo MQMD
PMQPMO	Puntero	Puntero a una estructura de datos de tipo MQPMO
PMQPTR	Puntero	Puntero a datos de tipo MQPTR
PMQRFH	Puntero	Puntero a una estructura de datos de tipo MQRFH
PMQRFH2	Puntero	Puntero a una estructura de datos de tipo MQRFH2
PMQRMH	Puntero	Puntero a una estructura de datos de tipo MQRMH
PMQRR	Puntero	Puntero a una estructura de datos de tipo MQRR

Nombre de tipo de datos elemental	Tipo de datos	Descripción
PMQSCO	Puntero	Puntero a una estructura de datos de tipo MQSCO
PMQD	Puntero	Puntero a una estructura de datos de tipo MQSD
PMQSMPO	Puntero	Puntero a una estructura de datos de tipo MQSMPO
PMQSRO	Puntero	Puntero a una estructura de datos de tipo MQSRO
PSSTS	Puntero	Puntero a una estructura de datos de tipo MQSTS
PMQTID	Puntero	Puntero a un ID de hebra
PMQTM	Puntero	Puntero a una estructura de datos de tipo MQTM
PMQTMC2	Puntero	Puntero a una estructura de datos de tipo MQTMC2
PMQUINT8	Puntero	Puntero a un tipo de datos de MQUINT8
PMQUINT16	Puntero	Puntero a un tipo de datos de MQUINT16
PMQUINT32	Puntero	Puntero a un tipo de datos de MQUINT32
PMQUINT64	Puntero	Puntero a un tipo de datos de MQUINT64
PMQULONG	Puntero	Puntero a un tipo de datos de MQULONG
PMQVOID	Puntero	
PMQWIH	Puntero	Puntero a una estructura de datos de tipo MQWIH
PMQXQH	Puntero	Puntero a una estructura de datos de tipo MQXQH

Declaraciones C

Tipo de datos	Representación
MQBOOL	`typedef MQLONG MQBOOL;`
MQBYTE	`typedef unsigned char MQBYTE;`
MQBYTE8	`typedef MQBYTE MQBYTE8[8];`
MQBYTE16	`typedef MQBYTE MQBYTE16[16];`

Tipo de datos	Representación
MQBYTE24	`typedef MQBYTE MQBYTE24[24];`
MQBYTE32	`typedef MQBYTE MQBYTE32[32];`
MQBYTE40	`typedef MQBYTE MQBYTE40[40];`
MQCHAR	`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;`
MQFLOAT64	`typedef double MQFLOAT64;`
MQHCONFIG	`typedef void MQPOINTER MQHCONFIG;`
MQHCONN	`typedef MQLONG MQHCONN;`
MQHOBJ	`typedef MQLONG MQHOBJ;`
MQINT8	`typedef signed char MQINT8;`
MQINT16	`typedef short MQINT16;`

Tipo de datos	Representación

MQINT64	En sistemas UNIX de 64 bits:
	`typedef long;`
	En AIX, Solaris y HP-UXde 32 bits:
	`typedef int64_t;`
	En IBM i, Linuxy z/OS:
	`typedef long long;`
	En Windows:
	`typedef _int64;`

MQLONG	En IBM i:
	`typedef long MQLONG;`
	Otras plataformas:
	`if defined(MQ_64_BIT)`
	`typedef int MQLONG;`
	`else`
	`typedef long MQLONG;`

IDMQP	`typedef MQLONG MQPID;`
MQPTR	`typedef void MQPOINTER MQPTR;`
MQTID	`typedef MQLONG MQTID;`
MQUINT8	`typedef unsigned char MQUINT8;`
MQUINT16	`typedef unsigned short MQUINT16;`

MQUINT64	En sistemas UNIX de 64 bits:
	`typedef unsigned long;`
	En AIX, Solaris y HP-UXde 32 bits:
	`typedef uint64_t;`
	En IBM i, Linuxy z/OS:
	`typedef unsigned long long;`
	En Windows:
	`typedef unsigned _int64;`

Tipo de datos	Representación

MQULONG	En IBM i:
	`typedef unsigned long MQULONG;`
	Otras plataformas:
	`if defined(MQ_64_BIT)`
	`  typedef unsigned int MQULONG;`
	`else`
	`  typedef unsigned long MQULONG;`

PMQBO	`typedef MQBO MQPOINTER PMQBO;`
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];`
PMQCHAR	`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];`
PMQCHAR64	`typedef MQCHAR64[64] MQPOINTER PMQCHAR64[64];`

Tipo de datos	Representación
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;`
PMQCNO	`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;`
PMQHOBJ	`typedef MQHOBJ MQPOINTER PMQHOBJ;`
PMQIIH	`typedef MQIIH MQPOINTER PMQIIH;`
PMQINT8	`typedef MQINT8 MQPOINTER PMQINT8;`
PMQINT16	`typedef MQINT16 MQPOINTER PMQINT16;`
PMQLONG	`typedef MQLONG MQPOINTER PMQLONG;`
PMQMD	`typedef MQMD MQPOINTER PMQMD;`
PMQMD1	`typedef MQMD1[1] MQPOINTER PMQMD1[1];`
PMQMDE	`typedef MQMDE MQPOINTER PMQMDE;`
PMQOD	`typedef MQOD MQPOINTER PMQOD;`

Tipo de datos	Representación
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;`
PMQTM	`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;`
PMQVOID	`typedef void MQPOINTER PMQVOID;`
PMQWIH	`typedef MQWIH MQPOINTER PMQWIH;`
PMQXQH	`typedef MQXQH MQPOINTER PMQXQH;`
PMQBO	`typedef PMQBO MQPOINTER PPMQBO;`
PMQBYTE	`typedef PMQBYTE MQPOINTER PPMQBYTE;`
PMQCHAR	`typedef PMQCHAR MQPOINTER PPMQCHAR;`
PMQCNO	`typedef PMQCNO MQPOINTER PPMQCNO;`
PMQGMO	`typedef PMQGMO MQPOINTER PPMQGMO;`
PMQHCONN	`typedef PMQHCONN MQPOINTER PPMQHCONN;`
PMQHOBJ	`typedef PMQHOBJ MQPOINTER PPMQHOBJ;`
PMQLONG	`typedef PMQLONG MQPOINTER PPMQLONG;`

Tipo de datos	Representación
PPMQMD	`typedef PMQMD MQPOINTER PPMQMD;`
PMQOD	`typedef PMQOD MQPOINTER PPMQOD;`
PMQPMO	`typedef PMQPMO MQPOINTER PPMQPMO;`
PMQULONG	`typedef PMQULONG MQPOINTER PPMQULONG;`
PMQVOID	`typedef PMQVOID MQPOINTER PPMQVOID;`
Donde `defined(MQ_64_BIT)` significa una plataforma de 64 bits.	

Consulte “Tipos de datos” en la página 245 para obtener una descripción de la variable de macro MQPOINTER.

Declaraciones COBOL

Tipo de datos	Representación
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)`
MQCHAR	`PIC X`
MQCHAR4	`PIC X(4)`
MQCHAR8	`PIC X(8)`
MQCHAR12	`PIC X(12)`
MQCHAR20	`PIC X(20)`
MQCHAR28	`PIC X(28)`

Tipo de datos	Representación
MQCHAR32	`PIC X(32)`
MQCHAR48	`PIC X(48)`
MQCHAR64	`PIC X(64)`
MQCHAR128	`PIC X(128)`
MQCHAR256	`PIC X(256)`
MQFLOAT32	`USAGE COMP-1`
MQFLOAT64	`USAGE COMP-2`
MQHCONN	`PIC S9(9) BINARY`
MQHOBJ	`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`
MQUINT64	`PIC 9(18) BINARY`
MQULONG	`PIC 9(9) BINARY`

Declaraciones PL/I
PL/I está soportado en z/OS.

Tipo de datos	Representación
MQBOOL	`fixed bin(31)`

Tipo de datos	Representación
MQBYTE	`char(1)`
MQBYTE8	`char(8)`
MQBYTE16	`char(16)`
MQBYTE24	`char(24)`
MQBYTE32	`char(32)`
MQBYTE40	`char(40)`
MQCHAR	`char(1)`
MQCHAR4	`char(4)`
MQCHAR8	`char(8)`
MQCHAR12	`char(12)`
MQCHAR20	`char(20)`
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)`
MQHOBJ	`fixed bin(31)`

Tipo de datos	Representación
MQINT8	`fixed bin(7)`
MQINT16	`fixed bin(15)`
MQINT64	`fixed bin(63)`
MQLONG	`fixed bin(31)`
MQPTR	`pointer`
MQUINT8	`fixed bin(8)`
MQUINT16	`fixed bin(16)`
MQUINT64	`fixed bin(64)`
MQULONG	`fixed bin(32)`

Declaraciones de ensamblador de System/390
System/390 assembler solo está soportado en z/OS .

Tipo de datos	Representación
MQBOOL	`DS F`
MQBYTE	`DS XL1`
MQBYTE8	`DS XL8`
MQBYTE16	`DS XL16`
MQBYTE24	`DS XL24`
MQBYTE32	`DS XL32`
MQBYTE40	`DS XL40`
MQCHAR	`DS CL1`
MQCHAR4	`DS CL4`
MQCHAR8	`DS CL8`

Tipo de datos	Representación
MQCHAR12	`DS CL12`
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`
MQHOBJ	`DS F`
MQINT8	`DS XL1`
MQINT16	`DS H`
MQINT64	`DS D`
MQLONG	`DS F`
MQPTR	`DS F`
MQUINT8	`DS XL1`
MQUINT16	`DS H`
MQUINT64	`DS D`
MQULONG	`DS F`

Tipos de datos de estructura-introducción

En esta sección se presentan los tipos de datos de estructura utilizados en la MQI. Los propios tipos de datos de estructura se describen en secciones posteriores.

Resumen

Las tablas siguientes resumen los tipos de datos de estructura utilizados en la MQI.

Tabla 467. Tipos de datos de estructura utilizados en llamadas MQI (o funciones de salida):

Estructura	**Descripción**	**Llamadas donde se utilizan**
MQACH	Cabecera de cadena de salida de API	
MQAIR	Registro de información de autenticación	MQCONNX
MQAXC	Contexto de salida de API	
MQAXP	Parámetro de salida de API	
MQBMHO	Opciones de almacenamiento intermedio a manejador de mensajes	MQBUFMH
MQBO	Opciones de inicio	MQBEGIN
MQCBD	Descriptor de devolución de llamada	MQCB
MQCBO	Opciones de creación de paquete	mqCreateBag
MQCHARV	Serie de longitud variable	MQINQMP
MQCNO	Opciones de conexión	MQCONNX
MQCSP	Parámetros de seguridad	MQCONNX
MQCTLO	Opciones de devolución de llamada	MQCTL
MQDMPO	Suprimir opciones de propiedad de mensaje	MQDLTMP
MQGMO	Opciones para obtener mensaje	MQGet
MQIMPO	Consultar opciones de propiedad de mensaje	MQINQMP
MQMD	Descriptor de mensaje	MQBUFMH, MQMHBUF, MQCB, MQGET, MQPUT, MQPUT1
MQMHBO	Manejador de mensajes para opciones de almacenamiento intermedio	MQMHBUF
MQOD	Descriptor de objetos	MQOPEN, MQPUT1
MQOR	Registro de objeto	MQOPEN, MQPUT1
MQPD	Descriptor de propiedad	MQSETMP
MQPMO	Opciones para transferir mensaje	MQPUT, MQPUT1

Tabla 467. Tipos de datos de estructura utilizados en llamadas MQI (o funciones de salida): (continuación)		
Estructura	**Descripción**	**Llamadas donde se utilizan**
MQPMR	Registro de transferencia de mensaje	MQPUT, MQPUT1
MQRR	Registro de respuestas	MQOPEN, MQPUT, MQPUT1
MQSCO	Opciones de configuración de SSL	MQCONNX
MQSD	Descriptor de suscripción	MQSUB
MQSMPO	Establecer opción de propiedad de mensaje	MQSETMP
MQSRO	Opciones de solicitud de suscripción	MQSUBRQ
MQSTS	Estructura de informes de estado	MQSTAT

Tabla 468. Tipos de datos de estructura utilizados en datos de mensaje:	
Estructura	**Descripción**
MQCIH	Cabecera de información de CICS
MQCFH	Cabecera PCF
MQEPH	Cabecera PCF incluida
MQDH	Cabecera de distribución
MQDLH	Cabecera de mensaje no entregado (mensaje no entregado)
MQIIH	Cabecera de información de IMS
MQMDE	Extensión del descriptor de mensaje
MQRFH	Cabecera de reglas y formato
MQRFH2	Cabecera de reglas y formato 2
MQRMH	Cabecera de mensaje de referencia
MQTM	Mensaje de desencadenante
MQTMC2	Mensaje desencadenante (formato de caracteres 2)
MQWIH	Cabecera de información de trabajo
MQXQH	Cabecera de cola de transmisión

Nota: La estructura MQDXP (parámetro de salida de conversión de datos) se describe en "salida de conversión de datos" en la página 888, junto con las llamadas de conversión de datos asociadas.

Reglas para tipos de datos de estructura

Los lenguajes de programación varían en su nivel de soporte para estructuras, y se adoptan determinadas reglas y convenios para correlacionar las estructuras MQI de forma coherente en cada lenguaje de programación:

1. Las estructuras deben estar alineadas en sus límites naturales.
   - La mayoría de las estructuras MQI requieren una alineación de 4 bytes.

- En IBM i, las estructuras que contienen punteros requieren una alineación de 16 bytes; estos son: MQCNO, MQOD, MQPMO.

2. Cada campo de una estructura debe estar alineado en su límite natural.
   - Los campos con tipos de datos que equivalen a MQLONG deben estar alineados en límites de 4 bytes.
   - Los campos con tipos de datos que equivalen a MQPTR deben estar alineados en límites de 16 bytes en IBM iy en límites de 4 bytes en otros entornos.
   - Otros campos están alineados en límites de 1 byte.
3. La longitud de una estructura debe ser un múltiplo de su alineación de límite.
   - La mayoría de las estructuras MQI tienen longitudes que son múltiplos de 4 bytes.
   - En IBM i, las estructuras que contienen punteros tienen longitudes que son múltiplos de 16 bytes.
4. Cuando sea necesario, se deben añadir bytes o campos de relleno para garantizar el cumplimiento de las reglas anteriores.

Convenciones utilizadas en las descripciones

La descripción de cada tipo de datos de estructura incluye:

- Una visión general de la finalidad y el uso de la estructura
- Descripciones de los campos de la estructura, en un formato independiente del lenguaje de programación
- Ejemplos de cómo se declara la estructura en cada uno de los lenguajes de programación soportados

La descripción de cada tipo de datos de estructura contiene las secciones siguientes:

Nombre de estructura
El nombre de la estructura, seguido de un resumen de los campos de la estructura.

Visión general
Una breve descripción del propósito y uso de la estructura.

Campos
Descripciones de los campos. Para cada campo, el nombre del campo va seguido de su tipo de datos elemental entre paréntesis (). En el texto, los nombres de campo se muestran utilizando un tipo de letra cursiva; por ejemplo, *Version*.

También hay una descripción de la finalidad del campo, junto con una lista de los valores que el campo puede tomar. Los nombres de constantes se muestran en mayúsculas; por ejemplo, MQGMO_STRUC_ID. Un conjunto de constantes que tienen el mismo prefijo se muestra utilizando el carácter *, por ejemplo: MQIA_ *.

En las descripciones de los campos, se utilizan los términos siguientes:

entrada
Proporcione información en el campo cuando realice una llamada.

salida
El gestor de colas devuelve información en el campo cuando la llamada se completa o falla.

entrada/salida
Proporcione información en el campo cuando realice una llamada y el gestor de colas cambie la información cuando la llamada se complete o falle.

Valores iniciales
Una tabla que muestra los valores iniciales para cada campo en los archivos de definición de datos proporcionados con la MQI.

Declaración C
Declaración típica de la estructura en C.

declaración COBOL
Declaración típica de la estructura en COBOL.

Declaración PL/I
Declaración típica de la estructura en PL/I.

System/390
Declaración típica de la estructura en el lenguaje ensamblador System/390 .

Declaración de Visual Basic
Declaración típica de la estructura en Visual Basic.

Programación C

Esta sección contiene información para ayudarle a utilizar la MQI del lenguaje de programación C.

Archivos de cabecera

Los archivos de cabecera se proporcionan para ayudarle a escribir programas de aplicación C que utilizan la MQI.

Estos archivos de cabecera se resumen en Tabla 469 en la página 244.

Tabla 469. Archivos de cabecera C

Archivo	Contenido
CMQC	Prototipos de función, tipos de datos y constantes con nombre para la MQI principal
CMQXC	Prototipos de función, tipos de datos y constantes con nombre para la salida de conversión de datos
CMQEC	Prototipos de función, tipos de datos y constantes con nombre para la MQI principal, la salida de conversión de datos y la estructura de puntos de entrada de interfaz (CMQEC incluye CMQXC y CMQC.)

Para mejorar la portabilidad de las aplicaciones, codifique el nombre del archivo de cabecera en minúsculas en la directiva de preprocesador `#include` :

```
#include "cmqec.h"
```

Funciones

No es necesario especificar todos los parámetros que se pasan por dirección cada vez que se invoca una función.

- Pasar parámetros que son *de sólo entrada* y de tipo MQHCONN, MQHOBJ o MQLONG por valor.
- Pase todos los demás parámetros por dirección.

Cuando no sea necesario un parámetro determinado, utilice un puntero nulo como parámetro en la invocación de la función, en lugar de la dirección de los datos del parámetro. Los parámetros para los cuales esto es posible se identifican en las descripciones de llamada.

No se devuelve ningún parámetro como valor de la función; en terminología C, esto significa que todas las funciones devuelven `void`.

Los atributos de la función se definen mediante la variable de macro MQENTRY; el valor de esta variable de macro depende del entorno.

Parámetros con tipo de datos no definido

El parámetro *`Buffer`* en las funciones MQGET, MQPUT y MQPUT1 tiene un tipo de datos no definido. Este parámetro se utiliza para enviar y recibir los datos de mensaje de la aplicación.

Los parámetros de este tipo se muestran en los ejemplos de C como matrices de MQBYTE. Puede declarar los parámetros de esta forma, pero normalmente es más conveniente declararlos como la estructura particular que describe el diseño de los datos en el mensaje. Declare el parámetro de función real como puntero a vacío y especifique la dirección de cualquier tipo de datos como parámetro en la invocación de la función.

Tipos de datos

Defina todos los tipos de datos utilizando la sentencia C `typedef` . Para cada tipo de datos, defina también el tipo de datos de puntero correspondiente. El nombre del tipo de datos de puntero es el nombre del tipo de datos elemental o de estructura prefijado con la letra P para denotar un puntero. Defina los atributos del puntero utilizando la variable de macro MQPOINTER; el valor de esta variable de macro depende del entorno. A continuación se muestra cómo declarar tipos de datos de puntero:

```
#define MQPOINTER *               /* depends on environment */
...
typedef MQLONG MQPOINTER PMQLONG;  /* pointer to MQLONG */
typedef MQMD   MQPOINTER PMQMD;    /* pointer to MQMD */
```

Manipulación de series binarias
Declare series de datos binarios como uno de los tipos de datos MQBYTEn.

Siempre que copie, compare o establezca campos de este tipo, utilice las funciones C `memcpy`, `memcmp`o `memset`; por ejemplo:

```
#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));
```

No utilice las funciones de serie `strcpy`, `strcmp`, `strncpy`o `strncmp`, porque no funcionan correctamente para los datos declarados con los tipos de datos MQBYTEn.

Manipulación de series de caracteres
Cuando el gestor de colas devuelve datos de tipo carácter a la aplicación, el gestor de colas siempre rellena los datos de tipo carácter con espacios en blanco hasta la longitud definida del campo; el gestor de colas *no* devuelve series terminadas en nulo.

Por lo tanto, al copiar, comparar o concatenar dichas series, utilice las funciones de serie `strncpy`, `strncmp`o `strncat`.

No utilice las funciones de serie que requieren que la serie termine con un valor nulo (`strcpy`, `strcmp`, `strcat`). Además, no utilice la función `strlen` para determinar la longitud de la serie; utilice en su lugar la función `sizeof` para determinar la longitud del campo.

Valores iniciales para estructuras
Los archivos de cabecera definen diversas variables de macro que puede utilizar para proporcionar valores iniciales para las estructuras de MQ al declarar instancias de dichas estructuras.

Estas variables de macro tienen nombre con el formato MQxxx_DEFAULT, donde MQxxx representa el nombre de la estructura. Se utilizan de la siguiente manera:

```
MQMD   MyMsgDesc = {MQMD_DEFAULT};
MQPMO  MyPutOpts = {MQPMO_DEFAULT};
```

Para algunos campos de caracteres (por ejemplo, los campos *StrucId* que aparecen en la mayoría de las estructuras, o el campo *Format* que aparece en MQMD), la MQI define valores concretos que son válidos. Para cada uno de los valores válidos, se proporcionan *dos* variables de macro:

- Una variable de macro define el valor como una serie con una longitud, excluyendo las coincidencias nulas implícitas, exactamente la longitud definida del campo. Por ejemplo, para el campo *Format* en MQMD se proporciona la siguiente variable de macro (¬ representa un carácter en blanco):

  ```
  #define MQFMT_STRING "MQSTR¬¬¬"
  ```

  Utilice este formulario con las funciones `memcpy` y `memcmp` .
- La otra variable de macro define el valor como una matriz de caracteres; el nombre de esta variable de macro es el nombre del formato de serie con el sufijo _ARRAY. Por ejemplo:

  ```
  #define MQFMT_STRING_ARRAY 'M','Q','S','T','R','¬','¬','¬'
  ```

  Utilice este formulario para inicializar el campo cuando declare una instancia de la estructura con valores distintos de los proporcionados por la variable de macro MQMD_DEFAULT. (Esto no siempre es necesario; en algunos entornos puede utilizar el formato de serie del valor en ambas situaciones. Sin embargo, puede utilizar el formato de matriz para las declaraciones, porque esto es necesario para la compatibilidad con el lenguaje de programación C++.)

Valores iniciales para estructuras dinámicas

Cuando se necesita un número variable de instancias de una estructura, las instancias normalmente se crean en el almacenamiento principal obtenido dinámicamente utilizando las funciones `calloc` o `malloc` . Para inicializar los campos en dichas estructuras, tenga en cuenta la técnica siguiente:

1. Declare una instancia de la estructura utilizando la variable de macro MQxxx_DEFAULT adecuada para inicializar la estructura. Esta instancia se convierte en el modelo para otras instancias:

   ```
   MQMD Model = {MQMD_DEFAULT};  /* declare model instance */
   ```

   Las palabras clave `static` o `auto` se pueden codificar en la declaración para proporcionar a la instancia de modelo un tiempo de vida estático o dinámico, según sea necesario.
2. Utilice las funciones `calloc` o `malloc` para obtener almacenamiento para una instancia dinámica de la estructura:

   ```
   PMQMD Instance;
   Instance = malloc(sizeof(MQMD));  /* get storage for dynamic instance */
   ```
3. Utilice la función `memcpy` para copiar la instancia de modelo en la instancia dinámica:

   ```
   memcpy(Instance,&Model,sizeof(MQMD));  /* initialize dynamic instance */
   ```

Utilizar desde C++

Para el lenguaje de programación C++, los archivos de cabecera contienen las siguientes sentencias adicionales que sólo se incluyen cuando se utiliza un compilador C++:

```
#ifdef __cplusplus
  extern "C" {
#endif

/* rest of header file */

#ifdef __cplusplus
  }
#endif
```

convenios de anotación
Esta información muestra cómo invocar las funciones y declarar parámetros.

En algunos casos, los parámetros son matrices con un tamaño que no es fijo. Para estos, se utiliza una n minúscula para representar una constante numérica. Cuando codifique la declaración para ese parámetro, sustituya el n por el valor numérico necesario.

Programación COBOL

Esta sección contiene información para ayudarle a utilizar la MQI del lenguaje de programación COBOL.

Archivos de copia

Se proporcionan varios archivos COPY para ayudarle a escribir programas de aplicación COBOL que utilizan MQI. Hay dos archivos que contienen constantes con nombre y dos archivos para cada una de las estructuras.

Cada estructura se proporciona en dos formatos: un formulario con valores iniciales y un formulario sin:

- Utilice las estructuras con valores iniciales en la SECCIÓN WORKING-STORAGE de un programa COBOL; están contenidas en archivos COPY con nombres con el sufijo de la letra V (para Valores).
- Utilice las estructuras sin valores iniciales en el `LINKAGE SECTION` de un programa COBOL; están contenidas en archivos COPY con nombres con el sufijo de la letra L (para enlace).

Los archivos COPY se resumen en Tabla 470 en la página 247. No todos los archivos listados están disponibles en todos los entornos.

Tabla 470. Archivos de copia COBOL

Archivo (con valores iniciales)	**Archivo (sin valores iniciales)**	**Contenido**
CMQAIRV	CMQAIRL	Registro de información de autenticación
CMQBOV	CMQBOL	Estructura de opciones de inicio
CMQCIHV	CMQCIHL	Estructura de cabecera de información de CICS
CMQCNOV	CMQCNOL	Estructura de opciones de conexión
CMQDHV	CMQDHL	Estructura de cabecera de distribución
CMQDLHV	CMQDLHL	Estructura de cabecera de mensaje no entregado
CMQDXPV	CMQDXPL	Estructura de parámetros de salida de conversión de datos
CMQGMOV	CMQGMOL	Estructura de opciones de obtención de mensajes
CMQIIHV	CMQIIHL	Estructura de cabecera de información de IMS
CMQMDV	CMQMDL	Estructura del descriptor de mensaje
CMQMDEV	CMQMDEL	Estructura de extensión de descriptor de mensaje
CMQMD1V	CMQMD1L	Estructura de descriptor de mensaje versión 1
CMQODV	CMQODL	Estructura de descriptor de objeto
CMQORV	CMQORL	Estructura de registro de objeto
CMQPMOV	CMQPMOL	Estructura de opciones de colocación de mensaje
CMQRFHV	CMQRFHL	Reglas y estructura de cabecera de formato
CMQRFH2V	CMQRFH2L	Reglas y estructura de cabecera de formato versión 2
CMQRMHV	CMQRMHL	Estructura de cabecera de mensaje de referencia
CMQRRV	CMQRRL	Estructura de registro de respuesta
CMQSCOV	CMQSCOL	Opciones de configuración de SSL

Tabla 470. Archivos de copia COBOL (continuación)		
Archivo (con valores iniciales)	**Archivo (sin valores iniciales)**	**Contenido**
CMQTMV	CMQTML	Estructura de mensajes desencadenantes
CMQTMCV	CMQTMCL	Estructura de mensaje desencadenante (formato de caracteres)
CMQTMC2V	CMQTMC2L	Estructura de mensaje desencadenante (formato de caracteres) versión 2
CMQWIHV	CMQWIHL	Estructura de cabecera de información de trabajo
CMQXQHV	CMQXQHL	Estructura de cabecera de cola de transmisión
CMQV	-	Constantes con nombre para MQI principal
CMQXV	-	Constantes con nombre para la salida de conversión de datos
CMQMD2V	CMQMD2L	Estructura de descriptor de mensaje versión 2

Estructuras

En el archivo COPY, cada declaración de estructura empieza con un elemento level-10 ; esto le permite declarar varias instancias de la estructura codificando la declaración level-01 y, a continuación, utilizando la sentencia COPY para copiar en el resto de la declaración de estructura. Para hacer referencia a la instancia adecuada, utilice la palabra clave `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.
```

Alinee las estructuras en los límites apropiados. Si utiliza la sentencia COPY para incluir una estructura a continuación de un elemento que no es el elemento level-01 , asegúrese de que la estructura empieza en el desplazamiento adecuado desde el inicio del elemento level-01 . La mayoría de las estructuras MQI requieren una alineación de 4 bytes; las excepciones son MQCNO, MQOD y MQPMO, que requieren una alineación de 16 bytes en IBM i.

En esta sección, los nombres de los campos de las estructuras se muestran sin un prefijo. En COBOL, los nombres de campo tienen como prefijo el nombre de la estructura seguido de un guión. Sin embargo, si el nombre de estructura termina con un dígito numérico, lo que indica que la estructura es una segunda o una versión posterior de la estructura original, el dígito numérico se omite del prefijo. Los nombres de campo en COBOL se muestran en mayúsculas (aunque se pueden utilizar mayúsculas o minúsculas si es necesario). Por ejemplo, el campo *MsgType* descrito en “MQMD - Descriptor de mensaje” en la página 392 pasa a ser MQMD-MSGTYPE en COBOL.

Las estructuras de sufijo V se declaran con valores iniciales para todos los campos; sólo es necesario establecer los campos en los que desea un valor que sea diferente del valor inicial proporcionado.

Punteros
Algunas estructuras necesitan direccionar datos opcionales que pueden no estar contiguos con la estructura, como los registros MQOR y MQRR a los que se dirige la estructura MQOD.

Para abordar estos datos opcionales, las estructuras contienen campos que se declaran con el tipo de datos de puntero. Sin embargo, COBOL no soporta el tipo de datos de puntero en todos los entornos. Debido a esto, los datos opcionales también se pueden abordar utilizando campos que contienen el desplazamiento de los datos desde el inicio de la estructura.

Si desea portar una aplicación entre entornos, compruebe si el tipo de datos de puntero está disponible en todos los entornos previstos. Si no es así, la aplicación debe direccionar los datos opcionales utilizando los campos de desplazamiento en lugar de los campos de puntero.

En aquellos entornos en los que los punteros no están soportados, declare los campos de puntero como series de bytes de la longitud adecuada, siendo el valor inicial la serie de bytes all-null. No altere este valor inicial si está utilizando los campos de desplazamiento.

Constantes con nombre

En esta sección, se muestran los nombres de constantes que contienen el carácter de subrayado (_) como parte del nombre. En COBOL, utilice el carácter de guión (-) en lugar del subrayado.

Las constantes que tienen valores de serie de caracteres utilizan las comillas simples como delimitador de serie ('). En algunos entornos, es posible que tenga que especificar una opción de compilador adecuada para que el compilador acepte la comilla simple como delimitador de serie en lugar de la comilla doble.

Las constantes con nombre se declaran en los archivos COPY como elementos level-10 . Para utilizar las constantes, declare el elemento level-01 explícitamente y, a continuación, utilice la sentencia COPY para copiar en las declaraciones de las constantes:

```
* Declare a structure to hold the constants
 01  MY-MQ-CONSTANTS.
     COPY CMQV.
```

El método anterior hace que las constantes ocupen almacenamiento en el programa aunque no estén referenciadas. Si incluye las constantes en muchos programas separados dentro de la misma unidad de ejecución, existen varias copias de las constantes, consumiendo almacenamiento principal innecesariamente. Evite este efecto utilizando una de las técnicas siguientes:

- Añada la cláusula GLOBAL a la declaración level-01 :

  ```
  * Declare a global structure to hold the constants
   01  MY-MQ-CONSTANTS GLOBAL.
       COPY CMQV.
  ```

  Esto hace que se asigne almacenamiento para un solo conjunto de constantes dentro de la unidad de ejecución. Sin embargo, cualquier programa de la unidad de ejecución puede hacer referencia a las constantes, no sólo al programa que contiene la declaración level-01 .

  Nota: La cláusula GLOBAL no está soportada en todos los entornos.
- Copie manualmente en cada programa sólo las constantes a las que hace referencia dicho programa. No utilice la sentencia COPY para copiar todas las constantes en el programa.

convenios de anotación

Los últimos temas de esta sección muestran cómo invocar las llamadas y declarar parámetros. En algunos casos, los parámetros son tablas o series de caracteres cuyo tamaño no es fijo. Para estos, se utiliza una n minúscula para representar una constante numérica. Cuando codifique la declaración para ese parámetro, sustituya el n por el valor numérico necesario.

Programación del ensamblador de System/390

Esta sección contiene información para ayudarle a utilizar la MQI del lenguaje de programación System/390 Assembler.

Macros

Se proporcionan varias macros para ayudarle a escribir programas de aplicación de ensamblador que utilizan MQI.

Hay dos macros para constantes con nombre y una macro para cada una de las estructuras. Estos archivos se resumen en Tabla 471 en la página 250.

Tabla 471. Macros de Assembler	
Archivo	**Contenido**
CMQA	Constantes con nombre (equates) para MQI principal
CMQCIHA	Estructura de cabecera de información de CICS
CMQCNOA	Estructura de opciones de conexión
CMQDLHA	Estructura de cabecera de mensaje no entregado
CMQDXPA	Estructura de parámetros de salida de conversión de datos
CMQGMOA	Estructura de opciones de obtención de mensajes
CMQIIHA	Estructura de cabecera de información de IMS
CMQMDA	Estructura del descriptor de mensaje
CMQMDEA	Estructura de extensión de descriptor de mensaje
CMQODA	Estructura de descriptor de objeto
CMQPMOA	Estructura de opciones de colocación de mensaje
CMQRFHA	Reglas y estructura de cabecera de formato
CMQRFH2A	Reglas y estructura de cabecera de formato versión 2
CMQRMHA	Estructura de cabecera de mensaje de referencia
CMQTMA	Estructura de mensajes desencadenantes
CMQTMC2A	Estructura de mensaje desencadenante (formato de caracteres) versión 2
CMQVERA	Control de versión de estructura
CMQWIHA	Estructura de cabecera de información de trabajo
CMQXA	Constantes con nombre para la salida de conversión de datos
CMQXPA	Estructura de parámetros de salida cruzada de API
CMQXQHA	Estructura de cabecera de cola de transmisión

Estructuras
Las estructuras son generadas por macros que tienen varios parámetros para controlar la acción de la macro. Estos parámetros se describen en las secciones siguientes.

De vez en cuando se introducen nuevas versiones de las estructuras de MQ . Los campos adicionales en una nueva versión pueden hacer que una estructura que anteriormente era menor que 256 bytes sea mayor que 256 bytes. Debido a esto, escriba las instrucciones del ensamblador que están pensadas para copiar una estructura MQ , o para establecer una estructura MQ en nulos, para trabajar correctamente con estructuras que pueden tener más de 256 bytes. De forma alternativa, utilice el parámetro de macro `DCLVER` o la macro CMQVERA con el parámetro `VERSION` para declarar una versión específica de la estructura.

Especificación del nombre de la estructura
Para declarar más de una instancia de una estructura, la macro prefija el nombre de cada campo de la estructura con una serie especificada por el usuario y un carácter de subrayado.

La serie utilizada es la etiqueta especificada en la invocación de la macro. Si no se especifica ninguna etiqueta, se utiliza el nombre de la estructura para construir el prefijo:

```
* Declare two object descriptors
```

```
         CMQODA  ,        Prefix used="MQOD_" (the default)
MY_MQOD  CMQODA  ,        Prefix used="MY_MQOD_"
```

Las declaraciones de estructura que se muestran en esta sección utilizan el prefijo predeterminado.

Especificación del formato de la estructura

Las declaraciones de estructura pueden ser generadas por la macro en uno de dos formatos, controlados por el parámetro DSECT :

DSECT=YES
Se utiliza una instrucción DSECT de ensamblador para iniciar una nueva sección de datos; la definición de estructura sigue inmediatamente a la sentencia DSECT . La etiqueta en la invocación de macro se usa como nombre de la sección de datos; si no se especifica ninguna etiqueta, se usará el nombre de la estructura.

DSECT=NO
Las instrucciones del ensamblador DC se utilizan para definir la estructura en la posición actual de la rutina. Los campos se inicializan con valores, que se pueden especificar codificando los parámetros relevantes en la invocación de macro. Los campos para los que no se especifican valores en la invocación de la macro se inicializan con valores predterminados.

El valor especificado debe estar en mayúsculas. Si no se especifica el parámetro DSECT , se asume DSECT=NO .

Control de la versión de la estructura
De forma predeterminada, las macros siempre declaran la versión más reciente de cada estructura.

Aunque puede utilizar el parámetro de macro VERSION para especificar un valor para el campo *Version* en la estructura, dicho parámetro define el valor inicial para el campo *Version* y no controla la versión de la estructura realmente declarada. Para controlar la versión de la estructura que se declara, utilice el parámetro DCLVER :

DCLVER=ACTUAL
La versión declarada es la versión actual (más reciente).

DCLVER=ESPECIFICADO
La versión declarada es la versión especificada por el parámetro VERSION . Si omite el parámetro VERSION , el valor predeterminado es la versión 1.

Si especifica el parámetro VERSION , el valor debe ser una constante numérica autodefinida o la constante con nombre para la versión necesaria (por ejemplo, MQCNO_VERSION_3). Si especifica algún otro valor, la estructura se declara como si se hubiera especificado DCLVER=CURRENT , incluso si el valor de VERSION se resuelve en un valor válido.

El valor especificado debe estar en mayúsculas. Si omite el parámetro DCLVER , el valor utilizado se toma de la variable de macro global MQDCLVER . Puede establecer esta variable utilizando la macro CMQVERA.

Declaración de una estructura incluida en otra

Para declarar una estructura como un componente de otra estructura, utilice el parámetro NESTED :

NESTED=SÍ
La declaración de estructura está anidada dentro de otra.

NESTED=NO
La declaración de estructura no está anidada dentro de otra.

El valor especificado debe estar en mayúsculas. Si omite el parámetro NESTED , se presupone NESTED=NO .

Especificación del valor inicial de un campo
Especifique el valor que se utilizará para inicializar un campo en una estructura codificando el nombre de ese campo (sin el prefijo) como parámetro en la invocación de la macro, acompañado del valor necesario.

Por ejemplo, para declarar una estructura de descriptor de mensaje con el campo *MsgType* inicializado con MQMT_REQUEST y el campo *ReplyToQ* inicializado con la serie "MY_REPLY_TO_QUEUE", utilice lo siguiente:

```
MY_MQMD  CMQMDA  MSGTYPE=MQMT_REQUEST,                              X
               REPLYTOQ=MY_REPLY_TO_QUEUE
```

Si especifica una constante con nombre (equate) como un valor en la invocación de la macro, utilice la macro CMQA para definir la constante con nombre. No encierre los valores de serie de caracteres entre comillas simples.

Control del listado

Controle el aspecto de la declaración de estructura en el listado del ensamblador utilizando el parámetro LIST :

LIST=YES
La declaración de estructura aparece en el listado del ensamblador.

LIST=NO
La declaración de estructura no aparece en el listado del ensamblador.

El valor especificado debe estar en mayúsculas. Si omite el parámetro LIST , se presupone LIST=NO .

Macro MQVERA

Esta macro le permite establecer el valor predeterminado que se utilizará para el parámetro DCLVER en las macros de estructura. El valor especificado por CMQVERA lo utiliza la macro de estructura sólo si omite el parámetro DCLVER de la invocación de la macro de estructura. El valor predeterminado se establece codificando la macro CMQVERA con el parámetro DCLVER :

DCLVER=ACTUAL
La versión predeterminada se establece en la versión actual (más reciente).

DCLVER=ESPECIFICADO
La versión predeterminada se establece en la versión especificada por el parámetro VERSION .

Debe especificar el parámetro DCLVER y el valor debe estar en mayúsculas. El valor establecido por CMQVERA sigue siendo el valor predeterminado hasta la siguiente invocación de CMQVERA o el final del ensamblaje. Si omite CMQVERA, el valor predeterminado es DCLVER=CURRENT.

convenios de anotación

Las secciones posteriores muestran cómo invocar las llamadas y declarar parámetros. En algunos casos, los parámetros son matrices o series de caracteres con un tamaño que no está fijado para el cual, se utiliza una n minúscula para representar una constante numérica. Cuando codifique la declaración para ese parámetro, sustituya el n por el valor numérico necesario.

MQAIR-Registro de información de autenticación

La estructura MQAIR representa el registro de información de autenticación.

La tabla siguiente resume los campos de la estructura.

Tabla 472. Campos en MQAIR

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Versión	Número de versión de la estructura	Versión
AuthInfoType	Tipo de información de autenticación	AuthInfoType
AuthInfoConnName	Nombre de conexión del servidor CRL LDAP	AuthInfoConnName

Tabla 472. Campos en MQAIR (continuación)		
Campo	**Descripción**	**Tema**
LDAPUserNamePtr	Dirección del nombre de usuario LDAP	LDAPUserNamePtr
Desplazamiento de LDAPUserName	Desplazamiento del nombre de usuario LDAP desde el inicio de MQSCO	LDAPUserNameDesplazamiento
LDAPUserNameLongitud	Longitud del nombre de usuario LDAP	LDAPUserNameLongitud
LDAPPassword	Contraseña para acceder al servidor LDAP	LDAPPassword
Nota: Los campos restantes se ignoran si *Versión* es menor que MQAIR_VERSION_2.		
OCSPResponderURL	URL en el que se puede contactar con el programa de respuesta OCSP	OCSPResponderURL

Visión general de MQAIR

La estructura MQAIR permite a una aplicación que se ejecuta como un cliente MQI de WebSphere MQ especificar información sobre un autenticador que se va a utilizar para la conexión de cliente. La estructura es un parámetro de entrada en la llamada MQCONNX.

Disponibilidad: clientes AIX, HP-UX, Solaris, Linux y Windows .

Conjunto de caracteres y codificación: Los datos de MQAIR deben estar en el juego de caracteres y la codificación del gestor de colas local; los proporciona el atributo de gestor de colas **CodedCharSetId** y MQENC_NATIVE.

Campos para MQAIR

La estructura MQAIR contiene los campos siguientes; los campos se describen en **orden alfabético**:

AuthInfoConnName (MQCHAR264)
Es el nombre de host o la dirección de red de un host en el que se ejecuta el servidor LDAP. Esto puede ir seguido de un número de puerto opcional, entre paréntesis. El número de puerto predeterminado es 389.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellena el valor con blancos hasta la longitud del campo. Si el valor no es válido, la llamada falla con el código de razón MQRC_AUTH_INFO_CONN_NAME_ERROR.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_AUTH_INFO_CONN_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y los caracteres en blanco en otros lenguajes de programación.

Tipo AuthInfo(MQLONG)
Este es el tipo de información de autenticación contenida en el registro.

El valor puede ser uno de los dos parámetros siguientes:

MQAIT_CRL_LDAP
Comprobación de revocación de certificados utilizando el servidor LDAP.

MQAIT_OCSP
Comprobación de revocación de certificados utilizando OCSP.

Si el valor no es válido, la llamada falla con el código de razón MQRC_AUTH_INFO_TYPE_ERROR.

Este es un campo de entrada. El valor inicial de este campo es MQAIT_CRL_LDAP.

LDAPPassword (MQCHAR32)
Esta es la contraseña necesaria para acceder al servidor CRL de LDAP. Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellena el valor con blancos hasta la longitud del campo.

Si el servidor LDAP no requiere una contraseña, u omite el nombre de usuario LDAP, *LDAPPassword* debe ser nulo o estar en blanco. Si omite el nombre de usuario LDAP y *LDAPPassword* no es nulo o está en blanco, la llamada falla con el código de razón MQRC_LDAP_PASSWORD_ERROR.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_LDAP_PASSWORD_LENGTH. El valor inicial de este campo es la serie nula en C y los caracteres en blanco en otros lenguajes de programación.

LDAPUserNameLongitud (MQLONG)

Es la longitud, en bytes, del nombre de usuario LDAP al que se dirige el campo *LDAPUserNamePtr* o *LDAPUserNameOffset* . El valor debe estar en el rango de cero a MQ_DISTINGUISHED_NAME_LENGTH. Si el valor no es válido, la llamada falla con el código de razón MQRC_LDAP_USER_NAME_LENGTH_ERR.

Si el servidor LDAP implicado no requiere un nombre de usuario, establezca este campo en cero.

Este es un campo de entrada. El valor inicial de este campo es 0.

LDAPUserNameDesplazamiento (MQLONG)
Es el desplazamiento en bytes del nombre de usuario LDAP desde el inicio de la estructura MQAIR.

El desplazamiento puede ser positivo o negativo. El campo se ignora si *LDAPUserNameLength* es cero.

Puede utilizar *LDAPUserNamePtr* o *LDAPUserNameOffset* para especificar el nombre de usuario LDAP, pero no ambos; consulte la descripción del campo *LDAPUserNamePtr* para obtener más detalles.

Este es un campo de entrada. El valor inicial de este campo es 0.

LDAPUserNamePtr (PMQCHAR)
Es el nombre de usuario de LDAP.

Consta del nombre distinguido del usuario que está intentando acceder al servidor CRL de LDAP. Si el valor es más corto que la longitud especificada por *LDAPUserNameLength*, termine el valor con un carácter nulo, o rellena el valor con espacios en blanco hasta la longitud *LDAPUserNameLength*. El campo se ignora si *LDAPUserNameLength* es cero.

Puede proporcionar el nombre de usuario LDAP de una de estas dos maneras:

- Utilizando el campo de puntero *LDAPUserNamePtr*

  En este caso, la aplicación puede declarar una serie que esté separada de la estructura MQAIR y establecer *LDAPUserNamePtr* en la dirección de la serie.

  Considere la posibilidad de utilizar *LDAPUserNamePtr* para lenguajes de programación que den soporte al tipo de datos de puntero de una forma que sea portable a distintos entornos (por ejemplo, el lenguaje de programación C).
- Utilizando el campo de desplazamiento *LDAPUserNameOffset*

  En este caso, la aplicación debe declarar una estructura compuesta que contenga la estructura MQSCO seguida de la matriz de registros MQAIR seguida de las series de nombre de usuario LDAP y establecer *LDAPUserNameOffset* en el desplazamiento de la serie de nombre adecuada desde el inicio de la estructura MQAIR. Asegúrese de que este valor es correcto y tiene un valor que se puede acomodar en un MQLONG (el lenguaje de programación más restrictivo es COBOL, para el que el rango válido es de -999 999 999 a +999 999 999 999).

  Considere la posibilidad de utilizar *LDAPUserNameOffset* para lenguajes de programación que no soportan el tipo de datos de puntero, o que implementan el tipo de datos de puntero de una forma que podría no ser portable a entornos diferentes (por ejemplo, el lenguaje de programación COBOL).

Sea cual sea la técnica elegida, utilice sólo uno de *LDAPUserNamePtr* y *LDAPUserNameOffset*; la llamada falla con el código de razón MQRC_LDAP_USER_NAME_ERROR si ambos son distintos de cero.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

OCSPResponderURL (MQCHAR256)
Para una estructura MQAIR que representa detalles de conexión para un programa de respuesta OCSP, este campo contiene el URL en el que se puede contactar con el programa de respuesta.

El valor de este campo es un URL HTTP. Este campo tiene prioridad sobre un URL en una extensión de certificado AIA (AuthorityInfoAccess).

El valor se ignora a menos que las dos sentencias siguientes sean verdaderas:

- La estructura MQAIR es de la versión 2 o posterior (el campo Versión se establece en MQAIR_VERSION_2 o superior).
- El campo de tipo AuthInfose establece en MQAIT_OCSP.

Si el campo no contiene un URL HTTP en el formato correcto (y no se ignora), la llamada MQCONNX falla con el código de razón MQRC_OCSP_URL_ERROR.

Este campo es sensible a las mayúsculas y minúsculas. Debe empezar con la serie http:// en minúsculas. El resto del URL puede ser sensible a las mayúsculas y minúsculas, en función de la implementación del servidor OCSP.

Este campo no está sujeto a la conversión de datos.

StrucId (MQCHAR4)

El valor debe ser:

MQAIR_ID_ESTRUCTURA
Identificador del registro de información de autenticación.

Para el lenguaje de programación C, también se define la constante MQAIR_STRUC_ID_ARRAY; tiene el mismo valor que MQAIR_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQAIR_STRUC_ID.

Versión (MQLONG)
El número de versión de la estructura MQAIR.

El valor debe ser uno de los siguientes:

MQAIR_VERSION_1
Registro de información de autenticación Version-1 .

MQAIR_VERSION_2
Registro de información de autenticación Version-2 .

La constante siguiente especifica el número de versión de la versión actual:

VERSIÓN_ACTUAL_MQAIR_VERSIÓN
Versión actual del registro de información de autenticación.

Siempre es un campo de entrada. El valor inicial de este campo es MQAIR_VERSION_1.

Valores iniciales y declaraciones de idioma para MQAIR

Tabla 473. Valores iniciales de los campos en MQAIR

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQAIR_ID_ESTRUCTURA	'AIR¬'
Versión	MQAIR_VERSION_1	1
AuthInfoType	MQAIT_CRL_LDAP	1

Tabla 473. Valores iniciales de los campos en MQAIR (continuación)

Nombre de campo	Nombre de constante	Valor de constante
AuthInfoConnName	Ninguna	Serie nula o espacios en blanco
LDAPUserNamePtr	Ninguna	Puntero nulo o bytes nulos
Desplazamiento de LDAPUserName	Ninguna	0
LDAPUserNameLongitud	Ninguna	0
LDAPPassword	Ninguna	Serie nula o espacios en blanco
OCSPResponderURL	Ninguna	Serie nula o espacios en blanco

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQAIR_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQAIR MyAIR = {MQAIR_DEFAULT};
```

Declaración C

```
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 */

};
```

declaración COBOL

```
**   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).
```

Declaración de Visual Basic

```
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
```

MQBMHO-Opciones de almacenamiento intermedio a manejador de mensajes

La tabla siguiente resume los campos de la estructura. MQBMHO structure-opciones de almacenamiento intermedio a manejador de mensajes

Tabla 474. Campos en MQBMHO

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Options	Opciones que controlan la acción de MQBMHO	Opciones

Visión general de MQBMHO

Disponibilidad: Todo. Estructura de opciones de almacenamiento intermedio a manejador de mensajes-visión general

Finalidad: la estructura MQBMHO permite a las aplicaciones especificar opciones que controlan cómo se generan los manejadores de mensajes a partir de los almacenamientos intermedios. La estructura es un parámetro de entrada en la llamada MQBUFMH.

Juego de caracteres y codificación: Los datos de MQBMHO deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (MQENC_NATIVE).

Campos para MQBMHO

Almacenamiento intermedio a campos de estructura de opciones de manejador de mensajes

La estructura MQBMHO contiene los campos siguientes; los campos se describen en **orden alfabético**:

Opciones (MQLONG)
Estructura de almacenamiento intermedio a descriptor de contexto de mensaje-Campo Opciones

El valor puede ser:

MQBMHO_DELETE_PROPERTIEs
Las propiedades que se añaden al descriptor de contexto de mensaje se suprimen del almacenamiento intermedio. Si la llamada falla, no se suprimen las propiedades.

Opciones predeterminadas: Si no necesita la opción descrita, utilice la opción siguiente:

MQBMHO_NONE
No se ha especificado ninguna opción.

Siempre es un campo de entrada. El valor inicial de este campo es MQBMHO_DELETE_PROPERcities.

StrucId (MQCHAR4)
Estructura de almacenamiento intermedio a descriptor de contexto de mensaje-Campo StrucId

Es el identificador de estructura. El valor debe ser:

MQBMHO_STRUC_ID
Identificador de la estructura de almacenamiento intermedio a descriptor de contexto de mensaje.

Para el lenguaje de programación C, la constante MQBMHO_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQBMHO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQBMHO_STRUC_ID.

Versión (MQLONG)
Estructura de almacenamiento intermedio a descriptor de contexto de mensaje-Campo Versión

Es el número de versión de la estructura. El valor debe ser:

MQBMHO_VERSION_1
Número de versión para la estructura de almacenamiento intermedio a descriptor de contexto de mensaje.

La constante siguiente especifica el número de versión de la versión actual:

MQBMHO_CURRENT_VERSION
Versión actual de la estructura de almacenamiento intermedio a descriptor de contexto de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es MQBMHO_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQBMHO
Estructura de almacenamiento intermedio a manejador de mensajes-Valores iniciales

Tabla 475. Valores iniciales de los campos en MQBMHO

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQBMHO_STRUC_ID	'BMHO'
Version	MQBMHO_VERSION_1	1
Options	MQBMHO_NONE	0

Notas:

1. En el lenguaje de programación C, la variable de macroMQBMHO_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQBMHO MyBMHO = {MQBMHO_DEFAULT};
```

Declaración C
Almacenamiento intermedio a estructura de manejador de mensajes-Declaración de lenguaje C

```
typedef struct tagMQBMHO MQBMHO;
struct tagMQBMHO {
  MQCHAR4  StrucId;        /* Structure identifier */
  MQLONG   Version;        /* Structure version number */
  MQLONG   Options;        /* Options that control the action of
                              MQBUFMH */
};
```

declaración COBOL
Almacenamiento intermedio a estructura de manejador de mensajes-Declaración de lenguaje COBOL

```
**   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.
```

Declaración PL/I
Almacenamiento intermedio a estructura de manejador de mensajes-Declaración de lenguaje PL/I

```
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 */
```

Declaración High Level Assembler
Almacenamiento intermedio a estructura de manejador de mensajes-Declaración de lenguaje Assembler

```
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)
```

MQBO-Opciones de inicio

La tabla siguiente resume los campos de la estructura.

Tabla 476. Campos en MQBO		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Options	Opciones que controlan la acción de MQBEGIN	Opciones

Visión general de MQBO

Disponibilidad: AIX, HP-UX, IBM i, Solaris, Linux, Windows; no disponible para clientes MQI de WebSphere MQ .

Finalidad: la estructura MQBO permite a la aplicación especificar opciones relacionadas con la creación de una unidad de trabajo. La estructura es un parámetro de entrada/salida en la llamada MQBEGIN.

Conjunto de caracteres y codificación: Los datos de MQBO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos para MQBO

La estructura MQBO contiene los campos siguientes; los campos se describen en **orden alfabético**:

Opciones (MQLONG)
Este campo siempre es un campo de entrada. Su valor inicial es MQBO_NONE.

El valor debe ser:

MQBO_NONE
No se ha especificado ninguna opción.

StrucId (MQCHAR4)
Este campo siempre es un campo de entrada. Su valor inicial es MQBO_STRUC_ID.

El valor debe ser:

MQBO_STRUC_ID
Identificador de la estructura de opciones de inicio.

Para el lenguaje de programación C, también se define la constante MQBO_STRUC_ID_ARRAY; tiene el mismo valor que MQBO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión (MQLONG)
Este campo siempre es un campo de entrada. Su valor inicial es MQBO_VERSION_1.

El valor debe ser:

MQBO_VERSION_1
Número de versión para la estructura de opciones de inicio.

La constante siguiente especifica el número de versión de la versión actual:

MQBO_CURRENT_VERSION
Versión actual de la estructura de opciones de inicio.

Valores iniciales y declaraciones de lenguaje para MQBO

Tabla 477. Valores iniciales de campos en MQBO para MQBO

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQBO_STRUC_ID	'BO¬¬'
Version	MQBO_VERSION_1	1
Options	MQBO_NONE	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQBO_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQBO MyBO = {MQBO_DEFAULT};
```

Declaración C

```
typedef struct tagMQBO MQBO;
struct tagMQBO {
  MQCHAR4  StrucId;  /* Structure identifier */
  MQLONG   Version;  /* Structure version number */
  MQLONG   Options;  /* Options that control the action of MQBEGIN */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración de Visual Basic

```
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
```

MQCBC-Contexto de devolución de llamada

La tabla siguiente resume los campos de la estructura. Estructura que describe la rutina de devolución de llamada.

Tabla 478. Campos en MQCBC		
Campo	**Descripción**	**Tema**
StrucID	Identificador de la estructura	StrucID
Version	Número de versión de la estructura	Versión
CallType	Por qué se ha llamado a la función	CallType
Hobj	Descriptor de contexto del objeto	Hobj
CallbackArea	Campo para que lo utilice la función de devolución de llamada	CallbackArea
ConnectionArea	Campo para que lo utilice la función de devolución de llamada	ConnectionArea
CompCode	Código de terminación	CompCode
Reason	Código de razón	Razón
State	Indicación del estado del consumidor actual	Estado
DataLength	Longitud del mensaje	DataLength
BufferLength	Longitud del almacenamiento intermedio de mensajes en bytes	BufferLength
Flags	Distintivos generales	Distintivos
Nota: El campo restante se ignora si la versión es menor que MQCBC_VERSION_2		
ReconnectDelay	Número de milisegundos antes del intento de reconexión	ReconnectDelay

Visión general de MQCBC

Disponibilidad: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS, más clientes MQI de WebSphere MQ conectados a estos sistemas.

Finalidad: la estructura MQCBC se utiliza para especificar información de contexto que se pasa a una función de devolución de llamada.

La estructura es un parámetro de entrada/salida en la llamada a una rutina de consumidor de mensajes.

Versión: La versión actual de MQCBC es MQCBC_VERSION_2.

Conjunto de caracteres y codificación: Los datos de MQCBC deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura estará en el juego de caracteres y la codificación del cliente.

Campos para MQCBC

Lista alfabética de campos para la estructura MQCBC.

La estructura MQCBC contiene los campos siguientes; los campos se describen en orden alfabético:

BufferLength (MQLONG)
Este campo es la longitud en bytes del almacenamiento intermedio de mensajes que se ha pasado a esta función.

El almacenamiento intermedio puede ser mayor que el valor de longitud MaxMsgdefinido para el consumidor y el valor de ReturnedLength en el MQGMO.

La longitud real del mensaje se proporciona en el campo DataLength .

La aplicación puede utilizar todo el almacenamiento intermedio para sus propios fines durante la duración de la función de devolución de llamada.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de excepciones.

CallbackArea (MQPTR)
Este campo está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma decisiones basadas en el contenido de este campo y se pasa sin cambios desde el campo "CallbackArea (MQPTR)" en la página 270 de la estructura MQCBD, que es un parámetro de la llamada MQCB utilizada para definir la función de devolución de llamada.

Los cambios en *CallbackArea* se conservan en las invocaciones de la función de devolución de llamada para un *HObj*. Este campo no se comparte con las funciones de devolución de llamada para otros manejadores.

Es un campo de entrada/salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

CallType (MQLONG)
Campo que contiene información sobre por qué se ha llamado a esta función; se definen los siguientes.

Tipos de llamada de entrega de mensajes: estos tipos de llamada contienen información sobre un mensaje. Los parámetros *DataLength* y *BufferLength* son válidos para estos tipos de llamada.

MQCBCT_MSG_REMOVED
La función de consumidor de mensajes se ha invocado con un mensaje que se ha eliminado de forma destructiva del descriptor de contexto de objeto.

Si el valor de *CompCode* es MQCC_WARNING, el valor del campo *Reason* es MQRC_TRUNCATED_MSG_ACCEPTED o uno de los códigos que indican un problema de conversión de datos.

MQCBCT_MSG_NOT_REMOVED
La función de consumidor de mensajes se ha invocado con un mensaje que todavía no se ha eliminado de forma destructiva del descriptor de objeto. El mensaje se puede eliminar de forma destructiva del descriptor de objeto utilizando *MsgToken*.

Es posible que el mensaje no se haya eliminado porque:

- Las opciones MQGMO solicitaron una operación de examen, MQGMO_BROWSE_ *
- El mensaje es mayor que el almacenamiento intermedio disponible y las opciones MQGMO no especifican MQGMO_ACCEPT_TRUNCATED_MSG

Si el valor de *CompCode* es MQCC_WARNING, el valor del campo *Reason* es MQRC_TRUNCATED_MSG_FAILED o uno de los códigos que indican un problema de conversión de datos.

Tipos de llamada de control de devolución de llamada: estos tipos de llamada contienen información sobre el control de la devolución de llamada y no contienen detalles sobre un mensaje. Estos tipos de llamada se solicitan utilizando Opciones en la estructura MQCBD.

Los parámetros *DataLength* y *BufferLength* no son válidos para estos tipos de llamada.

MQCBCT_REGISTER_CALL
La finalidad de este tipo de llamada es permitir que la función de devolución de llamada realice alguna configuración inicial.

La función de devolución de llamada se invoca inmediatamente después de que se registre la devolución de llamada, es decir, al volver de una llamada MQCB utilizando un valor para el campo *Operation* de MQOP_REGISTER.

Este tipo de llamada se utiliza tanto para consumidores de mensajes como para manejadores de sucesos.

Si se solicita, es la primera invocación de la función de devolución de llamada.

El valor del campo *Reason* es MQRC_NONE.

MQCBCT_START_CALL
La finalidad de este tipo de llamada es permitir que la función de devolución de llamada realice alguna configuración cuando se inicia, por ejemplo, restableciendo los recursos que se han limpiado cuando se detuvo anteriormente.

La función de devolución de llamada se invoca cuando se inicia la conexión utilizando MQOP_START o MQOP_START_WAIT.

Si una función de devolución de llamada se registra dentro de otra función de devolución de llamada, este tipo de llamada se invoca cuando se devuelve la devolución de llamada.

Este tipo de llamada sólo se utiliza para los consumidores de mensajes.

El valor del campo *Reason* es MQRC_NONE.

MQCBCT_STOP_CALL
La finalidad de este tipo de llamada es permitir que la función de devolución de llamada realice alguna limpieza cuando se detiene durante un tiempo, por ejemplo, limpiando recursos adicionales que se han adquirido durante el consumo de mensajes.

La función de devolución de llamada se invoca cuando se emite una llamada MQCTL utilizando un valor para el campo *Operation* de MQOP_STOP.

Este tipo de llamada sólo se utiliza para los consumidores de mensajes.

El valor del campo *Reason* se establece para indicar la razón de la detención.

MQCBCT_DEREGISTER_CALL
La finalidad de este tipo de llamada es permitir que la función de devolución de llamada realice la limpieza final al final del proceso de consumo. La función de devolución de llamada se invoca cuando:

- La función de devolución de llamada se anula del registro utilizando una llamada MQCB con MQOP_DEREGISTER.
- La cola está cerrada, lo que provoca un anulación de registro implícito. En esta instancia, la función de devolución de llamada se pasa a MQHO_UNUSABLE_HOBJ como descriptor de contexto de objeto.
- La llamada MQDISC se completa-causando un cierre implícito y, por lo tanto, un anulación del registro. En este caso, la conexión no se desconecta inmediatamente y cualquier transacción en curso todavía no se confirma.

Si alguna de estas acciones se realiza dentro de la propia función de devolución de llamada, la acción se invoca una vez que se devuelve la devolución de llamada.

Este tipo de llamada se utiliza tanto para consumidores de mensajes como para manejadores de sucesos.

Si se solicita, esta es la última invocación de la función de devolución de llamada.

El valor del campo *Reason* se establece para indicar la razón de la detención.

MQCBCT_EVENT_CALL
Función de manejador de sucesos

La función de manejador de sucesos se ha invocado sin un mensaje cuando el gestor de colas o la conexión se detiene o desactiva temporalmente.

Esta llamada se puede utilizar para realizar la acción adecuada para todas las funciones de devolución de llamada.

Función de consumidor de mensajes

La función de consumidor de mensajes se ha invocado sin un mensaje cuando se ha detectado un error (*CompCode* = MQCC_FAILED) que es específico del descriptor de contexto de objeto; por ejemplo, *Reason* code = MQRC_GET_inhibiTED.

El valor del campo *Reason* se establece para indicar la razón de la llamada.

MQCBCT_MC_EVENT_CALL

La función de manejador de sucesos se ha invocado para sucesos de multidifusión; el manejador de sucesos se envía WebSphere MQ Sucesos de multidifusión en lugar de sucesos 'normales' WebSphere MQ .

Para obtener más información sobre MQCBCT_MC_EVENT_CALL, consulte Informes de excepciones de multidifusión .

CompCode (MQLONG)
Este campo es el código de terminación. Indica si se han producido problemas al consumir el mensaje.

El valor puede ser uno de los siguientes:

MQCC_OK
Realización satisfactoria

MQCC_WARNING
Aviso (terminación parcial)

MQCC_FAILED
Llamada fallida

Este es un campo de entrada. El valor inicial de este campo es MQCC_OK.

ConnectionArea (MQPTR)
Este campo está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma decisiones basadas en el contenido de este campo y se pasa sin cambios desde el campo "ConnectionArea (MQPTR)" en la página 317 en la estructura MQCTLO, que es un parámetro en la llamada MQCTL utilizada para controlar la función de devolución de llamada.

Los cambios realizados en este campo por las funciones de devolución de llamada se conservan en las invocaciones de la función de devolución de llamada. Esta área se puede utilizar para pasar información que deben compartir todas las funciones de devolución de llamada. A diferencia de *CallbackArea* , esta área es común en todas las devoluciones de llamada para un descriptor de conexión.

Es un campo de entrada y salida. El valor inicial de este campo es un puntero nulo o bytes nulos.

DataLength (MQLONG)
Es la longitud en bytes de los datos de aplicación del mensaje. Si el valor es cero, significa que el mensaje no contiene datos de aplicación.

El campo DataLength contiene la longitud del mensaje, pero no necesariamente la longitud de los datos de mensaje pasados al consumidor. Puede ser que el mensaje se haya truncado. Utilice el campo ReturnedLength en el MQGMO para determinar cuántos datos se han pasado realmente al consumidor.

Si el código de razón indica que el mensaje se ha truncado, puede utilizar el campo DataLength para determinar el tamaño del mensaje real. Esto le permite determinar el tamaño del almacenamiento intermedio necesario para acomodar los datos del mensaje y, a continuación, emitir una llamada MQCB para actualizar la LongitudMaxMsg con un valor adecuado.

Si se especifica la opción MQGMO_CONVERT, el mensaje convertido podría ser mayor que el valor devuelto para DataLength. En tales casos, es probable que la aplicación tenga que emitir una llamada MQCB para actualizar MaxMsgLength para que sea mayor que el valor devuelto por el gestor de colas para DataLength.

Para evitar problemas de truncamiento de mensajes, especifique MaxMsgde longitud como MQCBD_FULL_MSG_LENGTH. Esto hace que el gestor de colas asigne un almacenamiento intermedio para la longitud completa del mensaje después de la conversión de datos. Sin embargo, tenga en cuenta que incluso si se especifica esta opción, todavía es posible que no haya suficiente almacenamiento disponible para procesar correctamente la solicitud. Las aplicaciones siempre deben comprobar el código de razón devuelto. Por ejemplo, si no es posible asignar almacenamiento suficiente para convertir el mensaje, los mensajes se devuelven a la aplicación sin convertir.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de sucesos.

Flags (MQLONG)
Distintivos que contienen información sobre este consumidor.

Se define la opción siguiente:

MQCBCF_READA_BUFFER_EMPTY
Este distintivo se puede devolver si una llamada MQCLOSE anterior que utilizaba la opción MQCO_QUIESCE ha fallado con un código de razón de MQRC_READ_AHEAD_MSGS.

Este código indica que se está devolviendo el último mensaje de lectura anticipada y que el almacenamiento intermedio está ahora vacío. Si la aplicación emite otra llamada MQCLOSE utilizando la opción MQCO_QUIESCE), se ejecuta correctamente.

Tenga en cuenta que no se garantiza que a una aplicación se le asigne un mensaje con este distintivo establecido, ya que es posible que todavía haya mensajes en el almacenamiento intermedio de lectura anticipada que no coincidan con los criterios de selección actuales. En este caso, la función de consumidor se invoca con el código de razón MQRC_HOBJ_QUIESCED.

Si el almacenamiento intermedio de lectura anticipada está completamente vacío, el consumidor se invoca con el distintivo MQCBCF_READA_BUFFER_EMPTY y el código de razón MQRC_HOBJ_QUIESCED_NO_MSGS.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de sucesos.

Hobj (MQHOBJ)
Es el descriptor de contexto de objeto para las llamadas al consumidor de mensajes.

Para un manejador de sucesos, este valor es MQHO_NONE

La aplicación puede utilizar este descriptor de contexto y la señal de mensaje en el bloque Obtener opciones de mensaje para obtener el mensaje si no se ha eliminado un mensaje de la cola.

Siempre es un campo de entrada. El valor inicial de este campo es MQHO_UNUSABLE_HOBJ

Razón (MQLONG)
Este es el código de razón que califica el *CompCode*.

Este es un campo de entrada. El valor inicial de este campo es MQRC_NONE.

Estado (MQLONG)
Indicación del estado del consumidor actual. Este campo es de mayor valor para una aplicación cuando se pasa un código de razón distinto de cero a la función de consumidor.

Puede utilizar este campo para simplificar la programación de aplicaciones porque no es necesario codificar el comportamiento para cada código de razón.

Este es un campo de entrada. El valor inicial de este campo es MQCS_NONE

Estado	**Acción del gestor de colas**	**Valor de constante**
MQCS_NONE Este código de razón representa una llamada normal sin información de razón adicional	No; esta es la operación normal.	0
MQCS_SUSPENDED_TEMPORARY Estos códigos de razón representan condiciones temporales.	Se llama a la rutina de devolución de llamada para informar de la condición y, a continuación, se suspende. Después de un periodo de tiempo, el sistema puede volver a intentar la operación, lo que puede llevar a que se vuelva a generar la misma condición.	1
MQCS_SUSPENDED_USER_ACTION Estos códigos de razón representan condiciones en las que la devolución de llamada debe realizar una acción para resolver la condición.	El consumidor se suspende y se llama a la rutina de devolución de llamada para informar de la condición. La rutina de devolución de llamada debe resolver la condición si es posible y RESUME o cerrar la conexión.	2
MQCS_SUSPENDED Estos códigos de razón representan anomalías que impiden más devoluciones de llamada de mensaje.	El gestor de colas suspende automáticamente la función de devolución de llamada. Si se reanuda la función de devolución de llamada, es probable que vuelva a recibir el mismo código de razón.	3
MQCS_STOPPED Estos códigos de razón representan el final del consumo de mensajes.	Se entrega al manejador de excepciones y a las devoluciones de llamada que han especificado MQCBDO_STOP_CALL. No se pueden consumir más mensajes.	4

StrucId (MQCHAR4)
El valor de este campo es el identificador de estructura.

El valor debe ser:

MQCBC_STRUC_ID
Identificador de la estructura de contexto de devolución de llamada.

Para el lenguaje de programación C, también se define la constante MQCBC_STRUC_ID_ARRAY; tiene el mismo valor que MQCBC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQCBC_STRUC_ID.

Versión (MQLONG)
El valor de este campo es el número de versión de la estructura.

El valor debe ser:

MQCBC_VERSION_1
Estructura de contexto de devolución de llamada Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQCBC_CURRENT_VERSION
Versión actual de la estructura de contexto de devolución de llamada.

Siempre es un campo de entrada. El valor inicial de este campo es MQCBC_VERSION_1.

La función de devolución de llamada siempre se pasa a la última versión de la estructura.

ReconnectDelay (MQLONG)
ReconnectDelay indica cuánto tiempo esperará el gestor de colas antes de intentar volver a conectarse. El campo puede ser modificado por un manejador de sucesos para cambiar el retardo o detener la reconexión por completo.

Utilice el campo `ReconnectDelay` sólo si el valor del campo `Razón` en el Contexto de devolución de llamada es `MQRC_RECONNECTING`.

Al entrar en el manejador de sucesos, el valor de `ReconnectDelay` es el número de milisegundos que el gestor de colas va a esperar antes de realizar un intento de reconexión. Tabla 479 en la página 267 lista los valores que puede establecer para modificar el comportamiento del gestor de colas a la devolución del manejador de sucesos.

Tabla 479. Valores de `ReconnectDelay`

Nombre	Valor	Descripción
`MQRD_NO_RECONNECT`	`-1`	No realice más intentos de reconexión. Se devuelve un error a la aplicación.
`MQRD_NO_DELAY`	`0`	Intente volver a conectarse inmediatamente.
`Milliseconds`	`>0`	Espere este número de milisegundos antes de volver a intentar la conexión.

Valores iniciales y declaraciones de lenguaje para MQCBC
Estructura de contexto de devolución de llamada-valores iniciales

No hay valores iniciales para la estructura **MQCBC** . La estructura se pasa como un parámetro a una rutina de devolución de llamada. El gestor de colas inicializa la estructura; las aplicaciones nunca la inicializan.

Declaración C
Estructura de contexto de devolución de llamada-Declaración de lenguaje C

```
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 */
```

declaración COBOL

```
** 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 **
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
    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)
```

MQCBD-Descriptor de devolución de llamada

La tabla siguiente resume los campos de la estructura. Estructura que especifica la función de devolución de llamada.

Tabla 480. Campos en MQCBD

Campo	Descripción	Tema
StrucID	Identificador de la estructura	StrucID
Version	Número de versión de la estructura	Versión
CallbackType	Tipo de función de devolución de llamada	CallbackType
Options	Opciones que controlan el consumo de mensajes	Opciones
Callback Area	Campo para que lo utilice la función de devolución de llamada	CallbackArea
CallbackFunction	Si la función se invoca como una llamada de API	CallbackFunction
CallbackName	Si la función se invoca como un programa enlazado dinámicamente	CallbackName
MaxMsgLength	Longitud del mensaje más largo que se puede leer	MaxMsgLength

Visión general de MQCBD

Disponibilidad: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OSy clientes MQI de WebSphere MQ conectados a estos sistemas.

Finalidad: la estructura MQCBD se utiliza para especificar una función de devolución de llamada y las opciones que controlan su uso por parte del gestor de colas.

La estructura es un parámetro de entrada en la llamada MQCB.

Versión: La versión actual de MQCBD es MQCBD_VERSION_1.

Conjunto de caracteres y codificación: los datos de MQCBD deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos para MQCBD

Lista alfabética de campos para la estructura MQCBD.

La estructura MQCBD contiene los campos siguientes; los campos se describen en orden alfabético:

CallbackArea (MQPTR)
Estructura del descriptor de devolución de llamada-Campo CallbackArea

Este es un campo que está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma decisiones basadas en el contenido de este campo y se pasa sin cambios desde el campo "CallbackArea (MQPTR)" en la página 262 en la estructura MQCBC, que es un parámetro en la declaración de función de devolución de llamada.

El valor sólo se utiliza en un *Operation* que tenga un valor MQOP_REGISTER, sin ninguna devolución de llamada definida actualmente, no sustituye a una definición anterior.

Es un campo de entrada y salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

CallbackFunction (MQPTR)
Estructura del descriptor de devolución de llamada-Campo CallbackFunction

La función de devolución de llamada se invoca como una llamada de función.

Utilice este campo para especificar un puntero a la función de devolución de llamada.

Debe especificar *CallbackFunction* o *CallbackName*. Si especifica ambos, se devuelve el código de razón MQRC_CALLBACK_ROUTINE_ERROR.

Si no se establece *CallbackName* ni *CallbackFunction* , la llamada falla con el código de razón MQRC_CALLBACK_ROUTINE_ERROR.

Esta opción no está soportada en el entorno siguiente: lenguajes de programación y compiladores que no dan soporte a referencias de puntero de función. En tales situaciones, la llamada falla con el código de razón MQRC_CALLBACK_ROUTINE_ERROR.

En z/OS , la función debe esperarse que se llame con convenios de enlace de sistema operativo. Por ejemplo, en el lenguaje de programación C, especifique:

```
#pragma linkage(MQCB_FUNCTION,OS)
```

Este es un campo de entrada. El valor inicial de este campo es un puntero nulo o bytes nulos.

Nota: Cuando se utiliza CICS con WebSphere MQ V7.0.1, se da soporte al consumo asíncrono si:

- Apar PK66866 se aplica a CICS TS 3.2
- Apar PK89844 se aplica a CICS TS 4.1

CallbackName (MQCHAR128)
Estructura del descriptor de devolución de llamada-Campo CallbackName

La función de devolución de llamada se invoca como un programa enlazado dinámicamente.

Debe especificar *CallbackFunction* o *CallbackName*. Si especifica ambos, se devuelve el código de razón MQRC_CALLBACK_ROUTINE_ERROR.

Si no se establece *CallbackName* ni *CallbackFunction* , la llamada falla con el código de razón MQRC_CALLBACK_ROUTINE_ERROR.

El módulo se carga cuando se registra la primera rutina de devolución de llamada que se va a utilizar y se descarga cuando se anula el registro de la última rutina de devolución de llamada que se va a utilizar.

Excepto cuando se indica en el texto siguiente, el nombre se justifica por la izquierda dentro del campo, sin blancos intercalados; el propio nombre se rellena con blancos hasta la longitud del campo. En las descripciones siguientes, los corchetes ([]) indican información opcional:

IBM i
El nombre de devolución de llamada puede tener uno de los formatos siguientes:

- Programa de biblioteca "/"

- Library "/" ServiceProgram "("FunctionName")"

Por ejemplo, `MyLibrary/MyProgram(MyFunction)`.

El nombre de biblioteca puede ser *LIBL. Los nombres de biblioteca y de programa están limitados a un máximo de 10 caracteres.

Sistemas UNIX

El nombre de devolución de llamada es el nombre de un módulo o biblioteca cargable dinámicamente, con el sufijo del nombre de una función que reside en dicha biblioteca. El nombre de función debe estar entre paréntesis. Opcionalmente, el nombre de biblioteca puede tener como prefijo una vía de acceso de directorio:

```
[path]library(function)
```

Si no se especifica la vía de acceso, se utiliza la vía de acceso de búsqueda del sistema.

El nombre está limitado a un máximo de 128 caracteres.

Windows

El nombre de devolución de llamada es el nombre de una biblioteca de enlace dinámico, con el sufijo del nombre de una función que reside en dicha biblioteca. El nombre de la función debe estar entre paréntesis. El nombre de la biblioteca puede tener como prefijo opcional una ruta de directorio y una unidad:

```
[d:][path]library(function)
```

Si la unidad y la vía de acceso no se especifican, se utiliza la vía de acceso de búsqueda del sistema.

El nombre está limitado a un máximo de 128 caracteres.

z/OS

El nombre de devolución de llamada es el nombre de un módulo de carga que es válido para la especificación en el parámetro EP de la macro LINK o LOAD.

El nombre está limitado a un máximo de 8 caracteres.

z/OS CICS

El nombre de devolución de llamada es el nombre de un módulo de carga que es válido para la especificación en el parámetro PROGRAM de la macro de mandato EXEC CICS LINK.

El nombre está limitado a un máximo de 8 caracteres.

El programa puede definirse como remoto utilizando la opción REMOTESYTEM de la definición PROGRAM instalada o mediante el programa de direccionamiento dinámico.

La región CICS remota debe estar conectada a WebSphere MQ si el programa va a utilizar llamadas de API de WebSphere MQ . Sin embargo, tenga en cuenta que el campo “Hobj (MQHOBJ)” en la página 266 de la estructura MQCBC no es válido en un sistema remoto.

Si se produce una anomalía al intentar cargar *CallbackName*, se devuelve uno de los siguientes códigos de error a la aplicación:

- MQRC_MODULE_NOT_FOUND
- MQRC_MODULE_INVALID
- MQRC_MODULE_ENTRY_NOT_FOUND

También se graba un mensaje en el registro de errores que contiene el nombre del módulo para el que se ha intentado la carga y el código de razón de anomalía del sistema operativo.

Este es un campo de entrada. El valor inicial de este campo es una serie nula o espacios en blanco.

CallbackType (MQLONG)
Estructura del descriptor de devolución de llamada-Campo CallbackType

Es el tipo de la función de devolución de llamada. El valor debe ser uno de los siguientes:

MQCBT_MESSAGE_CONSUMER
Define esta devolución de llamada como una función de consumidor de mensajes.

Se llama a una función de devolución de llamada de consumidor de mensajes cuando un mensaje, que cumple los criterios de selección especificados, está disponible en un descriptor de contexto de objeto y se inicia la conexión.

MQCBT_EVENT_HANDLER
Define esta devolución de llamada como la rutina de suceso asíncrona; no se controla que consuma mensajes para un descriptor de contexto.

Hobj no es necesario en la llamada MQCB que define el manejador de sucesos y se ignora si se especifica.

Se llama al manejador de sucesos para condiciones que afectan a todo el entorno de consumidor de mensajes. La función de consumidor se invoca sin un mensaje cuando se produce un suceso, por ejemplo, un gestor de colas o una conexión deteniéndose o desactivándose temporalmente. No se llama para condiciones que son específicas de un único consumidor de mensajes, por ejemplo, MQRC_GET_INITED.

Los sucesos se entregan a la aplicación, independientemente de si la conexión se ha iniciado o detenido, excepto en los entornos siguientes:

- Entorno CICS en z/OS
- aplicaciones sin hebras

Si el llamante no pasa uno de estos valores, la llamada falla con un código *Reason* de MQRC_CALLBACK_TYPE_ERROR

Siempre es un campo de entrada. El valor inicial de este campo es MQCBT_MESSAGE_CONSUMER.

MaxMsgLongitud (MQLONG)
Es la longitud en bytes del mensaje más largo que se puede leer desde el descriptor de contexto y se puede asignar a la rutina de devolución de llamada. Estructura del descriptor de devolución de llamada-Campo de longitud MaxMsg

Si un mensaje tiene una longitud mayor, la rutina de devolución de llamada recibe *MaxMsgLength* bytes del mensaje y el código de razón:

- MQRC_TRUNCATED_MSG_FAILED o
- MQRC_TRUNCATED_MSG_ACCEPTED si ha especificado MQGMO_ACCEPT_TRUNCATED_MSG.

La longitud real del mensaje se proporciona en el campo "DataLength (MQLONG)" en la página 265 de la estructura MQCBC.

Se define el siguiente valor especial:

MQCBD_FULL_MSG_LENGTH
El sistema ajusta la longitud del almacenamiento intermedio para devolver mensajes sin truncamiento.

Si no hay suficiente memoria disponible para asignar un almacenamiento intermedio para recibir el mensaje, el sistema llama a la función de devolución de llamada con un código de razón MQRC_STORAGE_NOT_AVAILABLE.

Si, por ejemplo, solicita la conversión de datos y no hay suficiente memoria disponible para convertir los datos del mensaje, el mensaje no convertido se pasa a la función de devolución de llamada.

Este es un campo de entrada. El valor inicial del campo *MaxMsgLength* es MQCBD_FULL_MSG_LENGTH.

Opciones (MQLONG)
Estructura de descriptor de devolución de llamada-Campo Opciones

Se puede especificar cualquiera de los siguientes, o todos ellos. Si se necesita más de una opción, los valores pueden ser:

- Sumado (no añadir la misma constante más de una vez), o
- Combinado utilizando la operación OR bit a bit (si el lenguaje de programación da soporte a operaciones de bits).

MQCBDO_FAIL_IF_QUIESCING
La llamada MQCB falla si el gestor de colas está en estado de inmovilización.

En z/OS, esta opción también fuerza que la llamada MQCB falle si la conexión (para una aplicación CICS o IMS) está en estado de inmovilización.

Especifique MQGMO_FAIL_IF_QUIESCING, en las opciones MQGMO pasadas en la llamada MQCB, para que se notifique a los consumidores de mensajes cuando estén inmovilizados.

Opciones de control: Las opciones siguientes controlan si se llama a la función de devolución de llamada, sin un mensaje, cuando cambia el estado del consumidor:

MQCBDO_REGISTER_CALL
La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_REGISTER_CALL.

MQCBDO_START_CALL
La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_START_CALL.

MQCBDO_STOP_CALL
La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_STOP_CALL.

MQCBDO_DEREGISTER_CALL
La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_DEREGISTER_CALL.

MQCBDO_EVENT_CALL
La función de devolución de llamada se invoca con el tipo de llamada MQCBCT_EVENT_CALL.

MQCBDO_MC_EVENT_CALL
La función de devolución de llamada se invoca con el tipo de llamada MQCBCT__MC_EVENT_CALL.

Consulte "CallType (MQLONG)" en la página 262 para obtener más detalles sobre estos tipos de llamada.

Opción predeterminada: Si no necesita ninguna de las opciones descritas, utilice la opción siguiente:

MQCBDO_NONE
Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

MQCBDO_NONE está definido para ayudar a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Este es un campo de entrada. El valor inicial del campo *Options* es MQCBDO_NONE.

StrucId (MQCHAR4)
Estructura del descriptor de devolución de llamada-Campo StrucId

Este es el identificador de estructura; el valor debe ser:

MQCBD_ID_STRUCD
Identificador de la estructura del descriptor de devolución de llamada.

Para el lenguaje de programación C, también se define la constante MQCBD_STRUC_ID_ARRAY; tiene el mismo valor que MQCBD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQCBD_STRUC_ID.

Versión (MQLONG)
Estructura de descriptor de devolución de llamada-Campo Versión

Este es el número de versión de la estructura; el valor debe ser:

MQCBD_VERSION_1
Estructura del descriptor de devolución de llamada Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQCBD_CURRENT_VERSION
Versión actual de la estructura del descriptor de devolución de llamada.

Siempre es un campo de entrada. El valor inicial de este campo es MQCBD_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQCBD

Estructura de descriptor de devolución de llamada-Valores iniciales

Tabla 481. Valores iniciales de los campos en MQCBD

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQCBD_ID_STRUCD	'CBD¬'
Version	MQCBD_VERSION_1	1
CallBackType	MQCBT_MESSAGE_CONSUMER	1
Options	MQCBDO_NONE	0
CallbackArea	Ninguna	Puntero nulo o blancos nulos
CallbackFunction	Ninguna	Puntero nulo o blancos nulos
CallbackName	Ninguna	Serie nula o espacios en blanco
MaxMsgLength	MQCBD_FULL_MSG_LENGTH	-1

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Cadena nula o blancos indica la cadena nula en el lenguaje de programación C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQCBD_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQCBD MyCBD = {MQCBD_DEFAULT};
```

Declaración C
Estructura de descriptor de devolución de llamada-Declaración de lenguaje C

```
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 */
 };
```

declaración COBOL

```
** 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.
```

Declaración PL/I

```
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 */
```

MQCHARV-Serie de longitud variable

La tabla siguiente resume los campos de la estructura.

Campo	Descripción	Tema
VSPtr	Puntero a la serie de longitud variable	VSPtr
VSOffset	Desplazamiento en bytes de la serie de longitud variable desde el inicio de la estructura que contiene esta estructura MQCHARV	VSOffset
VSLength	Longitud en bytes de la serie de longitud variable a la que se dirige el campo VSPtr o VSOffset.	Longitud VSLength
VSBufSize	El tamaño en bytes del almacenamiento intermedio direccionado por el campo VSPtr o VSOffset.	VSBufSize
VSCCSID	Identificador de juego de caracteres de la serie de longitud variable a la que se dirige el campo VSPtr o VSOffset.	VSCCSID

Visión general de MQCHARV

Disponibilidad: AIX, HP-UX, Solaris, Linux, IBM i, Windows, más clientes MQI de WebSphere MQ conectados a estos sistemas.

Finalidad: Utilice la estructura MQCHARV para describir una serie de longitud variable.

Conjunto de caracteres y codificación: Los datos de MQCHARV deben estar en la codificación del gestor de colas local que proporciona MQENC_NATIVE y el juego de caracteres del campo VSCCSID dentro de la estructura. Si la aplicación se ejecuta como un cliente MQ , la estructura debe estar en la codificación del cliente. Algunos juegos de caracteres tienen una representación que depende de la codificación. Si VSCCSID es uno de estos juegos de caracteres, la codificación utilizada es la misma que la de los otros campos de MQCHARV. El juego de caracteres identificado por VSCCSID puede ser un juego de caracteres de doble byte (DBCS).

Uso: la estructura MQCHARV direcciona los datos que pueden no estar contiguos con la estructura que los contiene. Para abordar estos datos, se pueden utilizar los campos declarados con el tipo de datos de

puntero. Tenga en cuenta que COBOL no da soporte al tipo de datos de puntero en todos los entornos. Debido a esto, los datos también se pueden abordar utilizando campos que contienen el desplazamiento de los datos desde el inicio de la estructura que contiene MQCHARV.

Programación COBOL

Si desea portar una aplicación entre entornos, debe determinar si el tipo de datos de puntero está disponible en todos los entornos previstos. Si no es así, la aplicación debe direccionar los datos utilizando los campos de desplazamiento en lugar de los campos de puntero.

En aquellos entornos en los que los punteros no están soportados, puede declarar los campos de puntero como series de bytes de la longitud adecuada, siendo el valor inicial la serie de bytes all-null. No altere este valor inicial si está utilizando los campos de desplazamiento. Una forma de hacerlo sin cambiar los libros de copia suministrados es utilizar lo siguiente:

```
COPY CMQCHRVV REPLACING POINTER BY ==BINARY PIC S9(9)==.
```

donde CMQCHRVV se puede intercambiar por el libro de copias que se va a utilizar.

Campos para MQCHARV

La estructura MQCHARV contiene los campos siguientes; los campos se describen en **orden alfabético**:

VSBufSize (MQLONG)
Es el tamaño en bytes del almacenamiento intermedio direccionado por el campo VSPtr o VSOffset.

Cuando se utiliza la estructura MQCHARV como campo de salida en una llamada a función, este campo debe inicializarse con la longitud del almacenamiento intermedio proporcionado. Si el valor de VSLength es mayor que VSBufSize , sólo se devuelven VSBufSize bytes de datos al interlocutor en el almacenamiento intermedio.

Este valor debe ser un valor mayor o igual que cero, o el siguiente valor especial que se reconoce:

MQVS_USE_VSLENGTH
Cuando se especifica, la longitud del almacenamiento intermedio se toma del campo VSLength en la estructura MQCHARV. No utilice este valor cuando utilice la estructura como campo de salida y se proporcione un almacenamiento intermedio.

Este es el valor inicial de este campo.

VSCCSID (MQLONG)
Es el identificador de juego de caracteres de la serie de longitud variable a la que se dirige el campo VSPtr o VSOffset.

El valor inicial de este campo es MQCCSI_APPL, definido por MQ para indicar que debe cambiarse por el identificador de juego de caracteres verdadero del proceso actual. Como resultado, el valor MQCCSI_APPL nunca se asocia con una serie de longitud variable. El valor inicial de este campo se puede cambiar definiendo un valor diferente para la constante MQCCSI_APPL para la unidad de compilación por los medios adecuados para el lenguaje de programación de la aplicación.

VSLength (MQLONG)
Longitud en bytes de la serie de longitud variable a la que se dirige el campo VSPtr o VSOffset.

El valor inicial de este campo es 0. El valor debe ser mayor o igual que cero o el siguiente valor especial que se reconoce:

MQVS_NULL_TERMINATED

Si no se especifica MQVS_NULL_TERMINATED, los bytes VSLength se incluyen como parte de la serie. Si hay caracteres nulos, no delimitan la serie.

Si se especifica MQVS_NULL_TERMINATED, la serie está delimitada por el primer nulo encontrado en la serie. El propio nulo no se incluye como parte de esa serie.

Nota: El carácter nulo utilizado para terminar una serie si se especifica MQVS_NULL_TERMINATED es un valor nulo del conjunto de códigos especificado por VSCCSID.

Por ejemplo, en UTF-16 (UCS-2 CCSID 1200 y 13488), es la codificación Unicode de dos bytes donde un valor nulo se representa mediante un número de 16 bits de todos los ceros. En UTF-16 es común encontrar bytes únicos establecidos en cero que forman parte de caracteres (por ejemplo, caracteres ASCII de 7 bits), pero las series sólo terminarán en nulo cuando se encuentren dos bytes 'cero' en un límite de bytes par. Es posible obtener dos 'cero' bytes en un límite impar cuando cada uno forma parte de caracteres válidos. Por ejemplo, x '01' x '00 x' 00 'x' 30 ' representa dos caracteres Unicode válidos y no termina de forma nula la serie.

VSOffset (MQLONG)
El desplazamiento puede ser positivo o negativo. Puede utilizar el campo VSPtr o VSOffset para especificar la serie de longitud variable, pero no ambos. Desplazamiento en bytes de la serie de longitud variable desde el inicio de MQCHARV, o la estructura que la contiene.

Cuando la estructura MQCHARV está incorporada en otra estructura, este valor es el desplazamiento en bytes de la serie de longitud variable desde el inicio de la estructura que contiene esta estructura MQCHARV. Cuando la estructura MQCHARV no está incorporada dentro de otra estructura, por ejemplo, si se especifica como parámetro en una llamada de función, el desplazamiento es relativo al inicio de la estructura MQCHARV.

El valor inicial de este campo es 0.

VSPtr (MQPTR)
Es un puntero a la serie de longitud variable.

Puede utilizar el campo VSPtr o VSOffset para especificar la serie de longitud variable, pero no ambos.

El valor inicial de este campo es un puntero nulo o bytes nulos.

Valores iniciales y declaraciones de lenguaje para MQCHARV

Valores iniciales de los campos en MQCHARV

Nombre de campo	Nombre de constante	Valor de constante
VSPtr	Ninguna	Puntero nulo o bytes nulos.
VSOffset	Ninguna	0
VSBufSize	MQVS_USE_VSLENGTH	0
VSLength	Ninguna	0
VSCCSID	MQCCSI_APPL	-3

Nota: En el lenguaje de programación C, la variable de macro MQCHARV_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQCHARV MyVarStr = {MQCHARV_DEFAULT};
```

Declaración C

```
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 */
};
```

Declaración COBOL para 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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Redefinición de MQCCSI_APPL

Los ejemplos siguientes muestran cómo puede alterar temporalmente el valor de MQCCSI_APPL en varios lenguajes de programación. Puede cambiar el valor de MQCCSI_APPL, eliminando la necesidad de establecer el VSCCSID para cada serie de longitud variable por separado.

En estos ejemplos, el CCSID se establece en 1208; cámbielo por el valor que necesite. Esto se convierte en el valor predeterminado, que puede alterar temporalmente estableciendo el VSCCSID en cualquier instancia específica de MQCHARV.

Uso de C

```
#define MQCCSI_APPL 1208
#include <cmqc.h>
```

Utilización de COBOL

```
COPY CMQXYZV REPLACING -3 BY 1208.
```

Uso de PL/I

```
%MQCCSI_APPL = '1208';
%include syslib(cmqp);
```

Uso del ensamblador de System/390

```
MQCCSI_APPL EQU 1208
         CMQA LIST=NO
```

MQCIH-Cabecera de puente CICS

La tabla siguiente resume los campos de la estructura.

Tabla 482. Campos en MQCIH

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
StrucLength	Longitud de la estructura MQCIH	StrucLength
Encoding	Reserved	Codificación
CodedCharSetId	Reserved	CodedCharSetId
Format	Nombre del formato MQ de los datos que siguen a MQCIH	Formato
Flags	Distintivos	Distintivos
ReturnCode	Código de retorno del puente	ReturnCode
CompCode	MQ código de terminación o CICS EIBRESP	CompCode
Reason	Código de razón o comentarios de MQ o CICS EIBRESP2	Razón
UOWControl	Control de unidad de trabajo	UOWControl
GetWaitInterval	Intervalo de espera para llamada MQGET emitida por tarea de puente	GetWaitInterval
LinkType	Tipo de enlace	LinkType
OutputDataLength	Longitud de datos de COMMAREA de salida	OutputDataLength
FacilityKeepTime	Tiempo de liberación de recurso de puente	FacilityKeepTime
ADSDescriptor	Descriptor ADS de envío/recepción	ADSDescriptor
ConversationalTask	Indica si la tarea puede ser conversacional	ConversationalTask
TaskEndStatus	Estado al final de la tarea	TaskEndStatus
Facility	Señal de recurso puente	Recurso
Function	Nombre de llamada de MQ o función EIBFN de CICS	Función
AbendCode	Código de terminación anómala	AbendCode
Authenticator	Contraseña o pase	Authenticator
Reserved1	Reserved	Reserved1

Tabla 482. Campos en MQCIH (continuación)		
Campo	**Descripción**	**Tema**
ReplyToFormat	Nombre del formato MQ del mensaje de respuesta	ReplyToFormat
RemoteSysId	ID de sistema CICS remoto a utilizar	RemoteSysId
RemoteTransId	CICS RTRANSID a utilizar	RemoteTransId
TransactionId	Transacción a adjuntar	TransactionId
FacilityLike	Atributos emulados de terminal	FacilityLike
AttentionId	Clave AID	AttentionId
StartCode	Código de inicio de transacción	StartCode
CancelCode	Código de transacción de terminación	CancelCode
NextTransactionId	Siguiente transacción a adjuntar	NextTransactionId
Reserved2	Reserved	Reserved2
Reserved3	Reserved	Reserved3
Nota: Los campos restantes no están presentes si *Version* es menor que MQCIH_VERSION_2.		
CursorPosition	Posición del cursor	CursorPosition
ErrorOffset	Desplazamiento de error en mensaje	ErrorOffset
InputItem	Reserved	InputItem
Reserved4	Reserved	Reserved4

Visión general de MQCIH

Disponibilidad: AIX, HP-UX, z/OS, Solaris, Linux, Windows, más clientes MQI de WebSphere MQ conectados a estos sistemas.

Finalidad: la estructura MQCIH describe la información que puede estar presente al inicio de un mensaje enviado al puente CICS a través de WebSphere MQ para z/OS.

Nombre de formato: MQFMT_CICS.

Versión: La versión actual de MQCIH es MQCIH_VERSION_2. Los campos que existen sólo en la versión más reciente de la estructura se identifican como tales en las descripciones siguientes.

Los archivos de cabecera, COPY e INCLUDE proporcionados para los lenguajes de programación soportados contienen la versión más reciente de MQCIH, con el valor inicial del campo *Version* establecido en MQCIH_VERSION_2.

Conjunto de caracteres y codificación: se aplican condiciones especiales al juego de caracteres y la codificación utilizados para la estructura MQCIH y los datos de mensaje de aplicación:

- Las aplicaciones que se conectan al gestor de colas propietario de la cola puente CICS deben proporcionar una estructura MQCIH que esté en el juego de caracteres y la codificación del gestor de colas. Esto se debe a que la conversión de datos de la estructura MQCIH no se realiza en este caso.
- Las aplicaciones que se conectan a otros gestores de colas pueden proporcionar una estructura MQCIH que esté en cualquiera de los conjuntos de caracteres y codificaciones soportados; el agente de canal de mensajes receptor conectado al gestor de colas propietario de la cola de puente CICS convierte la estructura MQCIH.
- Los datos del mensaje de aplicación que siguen a la estructura MQCIH deben estar en el mismo juego de caracteres y codificación que la estructura MQCIH. No puede utilizar los campos *CodedCharSetId*

y *Encoding* en la estructura MQCIH para especificar el juego de caracteres y la codificación de los datos del mensaje de aplicación.

Debe proporcionar una salida de conversión de datos para convertir los datos del mensaje de aplicación si los datos no son uno de los formatos incorporados soportados por el gestor de colas.

Uso: Si la aplicación requiere valores que son los mismos que los valores iniciales que se muestran en Tabla 484 en la página 290, y el puente se ejecuta con AUTH=LOCAL o AUTH=IDENTIFY, puede omitir la estructura MQCIH del mensaje. En todos los demás casos, la estructura debe estar presente.

El puente acepta una estructura MQCIH version-1 o version-2 , pero para las transacciones 3270, debe utilizar una estructura version-2 .

La aplicación debe asegurarse de que los campos documentados como campos de solicitud tengan los valores adecuados en el mensaje enviado al puente; estos campos se introducen en el puente.

Los campos documentados como campos de respuesta se establecen mediante el puente CICS en el mensaje de respuesta que el puente envía a la aplicación. La información de error se devuelve en los campos *ReturnCode*, *Function*, *CompCode*, *Reason*y *AbendCode* , pero no todos se establecen en todos los casos. La Tabla 483 en la página 281 muestra qué campos se establecen para distintos valores de *ReturnCode*.

Tabla 483. Contenido de los campos de información de error en la estructura MQCIH para MQCIH

ReturnCode	***Function***	***CompCode***	***Reason***	***AbendCode***
MQCRC_OK	-	-	-	-
ERROR DE MQCRC_BRIDGE_	-	-	MQFB_CICS_ *	-
MQCRC_MQ_API_ERROR MQCRC_BRIDGE_TIMEOUT	Nombre de llamada de MQ	MQ *CompCode*	MQ *Reason*	-
MQCRC_CICS_EXEC_ERROR MQCRC_SECURITY_ERROR MQCRC_PROGRAM_NOT_AVAILABLE MQCRC_TRANSID_NOT_AVAILABLE	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	-
MQCRC_BRIDGE_ABEND MQCRC_APPLICATION_ABEND	-	-	-	ABCODE de CICS

Campos para MQCIH

La estructura MQCIH contiene los campos siguientes; los campos se describen en **orden alfabético**:

AbendCode (MQCHAR4)
AbendCode es un campo de respuesta. La longitud de este campo la proporciona MQ_ABEND_CODE_LENGTH. El valor inicial de este campo es de 4 caracteres en blanco.

El valor devuelto en este campo sólo es significativo si el campo *ReturnCode* tiene el valor MQCRC_APPLICATION_ABEND o MQCRC_BRIDGE_ABEND. Si lo hace, *AbendCode* contiene el valor ABCODE de CICS .

ADSDescriptor (MQLONG)
Este campo es un indicador que especifica si se deben enviar descriptores ADS en solicitudes SEND y RECEIVE BMS.

Los valores siguientes están definidos:

MQCADSD_NONE
No envíe ni reciba descriptores ADS.

MQCADSD_SEND
Enviar descriptores ADS.

MQCADSD_RECV
Recibir descriptores ADS.

MQCADSD_MSGFORMAT
Utilice el formato de mensaje para los descriptores ADS.

Esto envía o recibe los descriptores ADS utilizando la forma larga del descriptor ADS. El formulario largo tiene campos alineados en límites de 4 bytes.

Establezca el campo *ADSDescriptor* como se indica a continuación:

- Si no utiliza descriptores ADS, establezca el campo en MQCADSD_NONE.
- Si utiliza descriptores ADS con el *mismo* CCSID en cada entorno, establezca el campo en la suma de MQCADSD_SEND y MQCADSD_RECV.
- Si utiliza descriptores ADS con CCSID *diferentes* en cada entorno, establezca el campo en la suma de MQCADSD_SEND, MQCADSD_RECV y MQCADSD_MSGFORMAT.

Es un campo de solicitud utilizado sólo para transacciones 3270. El valor inicial de este campo es MQCADSD_NONE.

AttentionId (MQCHAR4)
El valor de este campo determina el valor inicial de la tecla AID cuando se inicia la transacción. Es un valor de 1 byte, alineado a la izquierda.

AttentionId es un campo de solicitud utilizado sólo para transacciones 3270. La longitud de este campo la proporciona MQ_ATTENTION_ID_LENGTH. El valor inicial de este campo es de cuatro blancos.

Autenticador (MQCHAR8)
El valor de este campo es la contraseña o el pase.

Si la autenticación de identificador de usuario está activa para el puente CICS , se utiliza *Authenticator* con el identificador de usuario en el contexto de identidad MQMD para autenticar el remitente del mensaje.

Este es un campo de solicitud. La longitud de este campo la proporciona MQ_AUTHENTICATOR_LENGTH. El valor inicial de este campo es de 8 blancos.

CancelCode (MQCHAR4)
El valor de este campo es el código de terminación anómala que se utilizará para terminar la transacción (normalmente una transacción conversacional que solicita más datos). De lo contrario, este campo se establece en blancos.

Este campo es un campo de solicitud utilizado sólo para transacciones 3270. La longitud de este campo la proporciona MQ_CANCEL_CODE_LENGTH. El valor inicial de este campo es de cuatro blancos.

CodedCharSetId (MQLONG)
CodedCharSetId es un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

El ID de juego de caracteres para las estructuras soportadas que siguen a una estructura MQCIH es el mismo que el ID de juego de caracteres de la propia estructura MQCIH y se toma de cualquier cabecera WebSphere MQ anterior.

CompCode (MQLONG)
Este campo es un campo de respuesta. Su valor inicial es MQCC_OK

El valor devuelto en este campo depende de *ReturnCode*; consulte Tabla 483 en la página 281.

ConversationalTask (MQLONG)
Este campo es un indicador que especifica si se debe permitir que la tarea emita solicitudes para obtener más información, o si se debe detener la tarea y emitir un mensaje de terminación anómala.

El valor debe ser una de las opciones siguientes:

MQCCT_YES
La tarea es conversacional.

MQCCT_NO
La tarea no es conversacional.

Este campo es un campo de solicitud utilizado sólo para transacciones 3270. El valor inicial de este campo es MQCCT_NO.

CursorPosition (MQLONG)
El valor de este campo muestra la posición inicial del cursor cuando se inicia la transacción. Para transacciones conversacionales, la posición del cursor está en el vector RECEIVE.

Este campo es un campo de solicitud utilizado sólo para transacciones 3270. El valor inicial de este campo es 0. Este campo no está presente si *Version* es menor que MQCIH_VERSION_2.

Encoding (MQLONG)
Este campo es un campo reservado; su valor no es significativo. Su valor inicial es 0.

La codificación para estructuras soportadas que siguen a una estructura MQCIH es la misma que la codificación de la propia estructura MQCIH y se toma de cualquier cabecera WebSphere MQ anterior.

ErrorOffset (MQLONG)
El campo ErrorOffset muestra la posición de los datos no válidos detectados por la salida del puente. Este campo proporciona el desplazamiento desde el inicio del mensaje hasta la ubicación de los datos no válidos.

ErrorOffset es un campo de respuesta utilizado sólo para transacciones 3270. El valor inicial de este campo es 0. Este campo no está presente si *Version* es menor que MQCIH_VERSION_2.

Recurso (MQBYTE8)
Este campo muestra la señal de recurso de puente de 8 bytes.

Una señal de recurso de puente permite que varias transacciones en una pseudoconversación utilicen el mismo recurso de puente (terminal 3270 virtual). En el primer mensaje, o sólo en una pseudoconversación, establezca un valor de MQCFAC_NONE. Este valor indica a CICS que asigne un nuevo recurso de puente para este mensaje. Se devuelve una señal de recurso de puente en los mensajes de respuesta cuando se especifica un *FacilityKeepTime* distinto de cero en el mensaje de entrada. Los mensajes de entrada posteriores dentro de una pseudoconversación deben utilizar la misma señal de recurso de puente.

Se define el siguiente valor especial:

MQCFAC_NONE
No se ha especificado ninguna señal de recurso.

Para el lenguaje de programación C, la constante MQCFAC_NONE_ARRAY también está definida y tiene el mismo valor que MQCFAC_NONE, pero es una matriz de caracteres en lugar de una serie.

Este campo es a la vez un campo de solicitud y un campo de respuesta utilizado sólo para transacciones 3270. La longitud de este campo la proporciona MQ_FACILITY_LENGTH. El valor inicial de este campo es MQCFAC_NONE.

Tiempo de FacilityKeep(MQLONG)
FacilityKeepTiempo es el tiempo en segundos que se mantiene el recurso de puente después de que finalice la transacción de usuario.

Para transacciones pseudoconversacionales, especifique un valor que corresponda a la duración esperada de una pseudoconversación; especifique cero para la última transacción de una pseudoconversación, y para otros tipos de transacción especifique cero.

Este campo es un campo de solicitud utilizado sólo para transacciones 3270. El valor inicial de este campo es 0.

FacilityLike (MQCHAR4)
FacilityLike es el nombre de un terminal instalado que se va a utilizar como modelo para el recurso de puente.

Un valor de blancos significa que *FacilityLike* se toma de la definición de perfil de transacción de puente o que se utiliza un valor por omisión.

Este campo es un campo de solicitud utilizado sólo para transacciones 3270. La longitud de este campo la proporciona MQ_FACILITY_LIKE_LENGTH. El valor inicial de este campo es de cuatro blancos.

Flags (MQLONG)
Este campo es un campo de solicitud. El valor inicial de este campo es MQCIH_NONE.

El valor debe ser:

MQCIH_NONE
Sin distintivos.

MQCIH_PASS_EXPIRATION
El mensaje de respuesta contiene:

- Las mismas opciones de informe de caducidad que el mensaje de solicitud.
- El tiempo de caducidad restante del mensaje de solicitud sin realizar ningún ajuste para el tiempo de proceso del puente.

Si omite este valor, la hora de caducidad se establece en *unlimited*.

MQCIH_REPLY_WITHOUT_NULLS
La longitud del mensaje de respuesta de una solicitud de programa CICS DPL se ajusta para excluir los nulos finales (X'00 ') al final de la COMMAREA devuelta por el programa DPL. Si no se establece este valor, es posible que los valores nulos sean significativos y se devuelva el COMMAREA completo.

MQCIH_SYNC_ON_RETURN
El enlace CICS para solicitudes DPL utiliza la opción SYNCONRETURN, lo que hace que CICS tome un punto de sincronización cuando el programa se completa si se envía a otra región CICS . El puente no especifica a qué región CICS enviar la solicitud; esto lo controla la definición de programa CICS o los recursos de equilibrio de carga de trabajo.

Format (MQCHAR8)
Este campo muestra el nombre de formato WebSphere MQ de los datos que siguen a la estructura MQCIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que las reglas para codificar el campo *Format* en MQMD.

Este nombre de formato también se utiliza para el mensaje de respuesta, si el campo *ReplyToFormat* tiene el valor MQFMT_NONE.

- Para las solicitudes DPL, *Format* debe ser el nombre de formato de COMMAREA.
- Para las solicitudes 3270, *Format* debe ser CSQCBDCIy el puente establece el formato en CSQCBDC0 para los mensajes de respuesta.

Las salidas de conversión de datos para estos formatos deben estar instaladas en el gestor de colas donde se van a ejecutar.

Si el mensaje de solicitud genera un mensaje de respuesta de error, el mensaje de respuesta de error tiene un nombre de formato de MQFMT_STRING.

Este campo es un campo de solicitud. La longitud de este campo la proporciona MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

Función (MQCHAR4)
Este campo es un campo de respuesta. La longitud de este campo la proporciona MQ_FUNCTION_LENGTH. El valor inicial de este campo es MQCFUNC_NONE.

El valor devuelto en este campo depende de *ReturnCode*; consulte Tabla 483 en la página 281. Los valores siguientes son posibles cuando *Function* contiene un nombre de llamada de WebSphere MQ :

MQCFUNC_MQCONN
Llamada MQCONN.

MQCFUNC_MQGET
Llamada MQGET.

MQCFUNC_MQINQ
Llamada MQINQ.

MQCFUNC_MQOPEN
Llamada MQOPEN.

MQCFUNC_MQPUT
Llamada MQPUT.

MQCFUNC_MQPUT1
Llamada MQPUT1 .

MQCFUNC_NONE
Sin llamada.

En todos los casos, para el lenguaje de programación C también se definen las constantes MQCFUNC_ * _ARRAY; estas constantes tienen los mismos valores que las constantes MQCFUNC_ * correspondientes, pero son matrices de caracteres en lugar de series.

Intervalo GetWait(MQLONG)
Este campo es un campo de solicitud. Su valor inicial es MQCGWI_DEFAULT.

Este campo sólo se aplica cuando *UOWControl* tiene el valor MQCUOWC_FIRST. Permite a la aplicación emisora especificar el tiempo aproximado en milisegundos que las llamadas MQGET emitidas por el puente esperarán el segundo y los mensajes de solicitud subsiguientes para la unidad de trabajo iniciada por este mensaje. Este recurso altera temporalmente el intervalo de espera predeterminado utilizado por el puente. Puede utilizar los siguientes valores especiales:

MQCGWI_DEFAULT
Intervalo de espera predeterminado.

Este valor hace que el puente CICS espere el tiempo especificado cuando se inició el puente.

MQWI_UNLIMITED
Intervalo de espera ilimitado.

InputItem (MQLONG)
Este campo es un campo reservado. El valor debe ser 0.

Este campo no está presente si *Version* es menor que MQCIH_VERSION_2.

LinkType (MQLONG)
Este campo es un campo de solicitud. Su valor inicial es MQCLT_PROGRAM.

Este valor indica el tipo de objeto que el puente intenta enlazar. Tiene que ser uno de los valores siguientes:

MQCLT_PROGRAM
Programa DPL.

TRANSACCIÓN_MQC
Transacción 3270.

NextTransactionId (MQCHAR4)
Este valor es el nombre de la siguiente transacción devuelta por la transacción de usuario (normalmente por EXEC CICS RETURN TRANSID). Si no hay ninguna transacción siguiente, este campo se establece en blancos.

Este campo es un campo de respuesta utilizado sólo para transacciones 3270. La longitud de este campo la proporciona MQ_TRANSACTION_ID_LENGTH. El valor inicial de este campo es de cuatro blancos.

Longitud de OutputData(MQLONG)
Este campo es un campo de petición utilizado sólo para programas DPL. Su valor inicial es MQCODL_AS_INPUT.

Este valor es la longitud de los datos de usuario que deben devolverse al cliente en un mensaje de respuesta. Esta longitud incluye el nombre de programa de 8 bytes. La longitud de COMMAREA pasada al programa enlazado es el máximo de este campo y la longitud de los datos de usuario en el mensaje de solicitud, menos 8.

Nota: La longitud de los datos de usuario en un mensaje es la longitud del mensaje excluyendo la estructura MQCIH.

Si la longitud de los datos de usuario en el mensaje de solicitud es menor que *OutputDataLength*, se utiliza la opción DATALENGTH del mandato LINK , lo que permite que LINK se envíe de forma eficaz a otra región CICS .

Puede utilizar el siguiente valor especial:

MQCODL_AS_INPUT
La longitud de salida es la misma que la longitud de entrada.

Este valor puede ser necesario incluso si no se solicita ninguna respuesta, para asegurarse de que el COMMAREA pasado al programa enlazado tiene un tamaño suficiente.

Razón (MQLONG)
Este campo es un campo de respuesta. Su valor inicial es MQRC_NONE.

El valor devuelto en este campo depende de *ReturnCode*; consulte Tabla 483 en la página 281.

RemoteSysId (MQCHAR4)
Este campo muestra el identificador del sistema CICS del sistema CICS que procesa la solicitud.

Si este campo está en blanco, la solicitud del sistema CICS se procesa en el mismo sistema CICS que el supervisor de puente. El SYSID utilizado se devuelve en el mensaje de respuesta.

Para una pseudoconversación 3270, todos los mensajes posteriores de la conversación deben especificar el SYSID remoto devuelto en la respuesta inicial. Si se especifica, el SYSID debe:

- Estar activo.
- Tenga acceso a la cola de solicitudes de WebSphere MQ .
- Ser accesible mediante los enlaces ISC de CICS desde el sistema CICS del supervisor de puente.

RemoteTransId (MQCHAR4)
Este campo es un campo de solicitud opcional. La longitud de este campo la proporciona MQ_TRANSACTION_ID_LENGTH.

Si se especifica, el campo se utiliza como el valor RTRANSID de CICS START.

Formato ReplyTo(MQCHAR8)
El valor de este campo es el nombre de formato WebSphere MQ del mensaje de respuesta que se envía en respuesta al mensaje actual.

Las reglas para codificar este campo son las mismas que las reglas para codificar el campo *Format* en MQMD.

Este campo es un campo de petición utilizado sólo para programas DPL. La longitud de este campo la proporciona MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

Reserved1 (MQCHAR8)
Este campo es un campo reservado. El valor debe ser 8 espacios en blanco.

Reserved2 (MQCHAR8)
Este campo es un campo reservado. El valor debe ser 8 espacios en blanco.

Reserved3 (MQCHAR8)
Este campo es un campo reservado. El valor debe ser 8 espacios en blanco.

Reserved4 (MQLONG)
Este campo es un campo reservado. El valor debe ser 0.

Este campo no está presente si *Version* es menor que MQCIH_VERSION_2.

ReturnCode (MQLONG)
El valor de este campo es el código de retorno del puente CICS que describe el resultado del proceso realizado por el puente. Este campo es un campo de respuesta, con un valor inicial de MQCRC_OK.

Los campos *Function*, *CompCode*, *Reason* y *AbendCode* pueden contener información adicional (consulte Tabla 483 en la página 281). El valor puede ser uno de los siguientes:

MQCRC_APPLICATION_ABEND
(5, X'005 ') La aplicación ha finalizado de forma anómala.

MQCRC_BRIDGE_ABEND
(4, X'004 ') El puente CICS ha finalizado de forma anómala.

ERROR DE MQCRC_BRIDGE_
(3, X'003 ') El puente CICS ha detectado un error.

MQCRC_BRIDGE_TIMEOUT
(8, X'008 ') Segundo o posterior mensaje dentro de la unidad de trabajo actual no recibido dentro del tiempo especificado.

MQCRC_CICS_EXEC_ERROR
(1, X'001 ') La sentencia EXEC CICS ha detectado un error.

MQCRC_MQ_API_ERROR
(2, X'002 ') La llamada MQ ha detectado un error.

MQCRC_OK
(0, X'000 ') Sin error.

MQCRC_PROGRAM_NOT_AVAILABLE
(7, X'007 ') Programa no disponible.

MQCRC_SECURITY_ERROR
(6, X'006 ') Se ha producido un error de seguridad.

MQCRC_TRANSID_NOT_AVAILABLE
(9, X'009 ') Transacción no disponible.

StartCode (MQCHAR4)
El valor de este campo es un indicador que especifica si el puente emula una transacción de terminal o una transacción iniciada con START.

El valor debe ser uno de los siguientes:

MQCSC_START
Inicio.

MQCSC_STARTDATA
Datos de inicio.

MQCSC_TERMINPUT
Entrada de terminal.

MQCSC_NONE
Ninguno.

En todos los casos, para el lenguaje de programación C también se definen las constantes MQCSC_ * _ARRAY; estas constantes tienen los mismos valores que las constantes MQCSC_ * correspondientes, pero son matrices de caracteres en lugar de series.

En la respuesta del puente, este campo se establece en el código de inicio adecuado para el siguiente ID de transacción contenido en el campo *NextTransactionId* . Los siguientes códigos de inicio son posibles en la respuesta:

- MQCSC_START
- MQCSC_STARTDATA
- MQCSC_TERMINPUT

Para CICS Transaction Server Versión 1.2, este campo es sólo un campo de solicitud; su valor en la respuesta no está definido.

Para CICS Transaction Server Versión 1.3 y releases posteriores, este campo es un campo de solicitud y un campo de respuesta.

Este campo sólo se utiliza para transacciones 3270. La longitud de este campo la proporciona MQ_START_CODE_LENGTH. El valor inicial de este campo es MQCSC_NONE.

StrucId (MQCHAR4)
Este campo es un campo de solicitud, con un valor inicial de MQCIH_STRUC_ID.

El valor debe ser:

MQCIH_STRUC_ID
Identificador de la estructura de cabecera de información de CICS .

Para el lenguaje de programación C, también se define la constante MQCIH_STRUC_ID_ARRAY; tiene el mismo valor que MQCIH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

StrucLength (MQLONG)
Este campo es un campo de solicitud, con un valor inicial de MQCIH_LENGTH_2.

El valor debe ser uno de los siguientes:

MQCIH_LENGTH_1
Longitud de la estructura de cabecera de información version-1 CICS .

MQCIH_LENGTH_2
Longitud de la estructura de cabecera de información version-2 CICS .

La constante siguiente especifica la longitud de la versión actual:

MQCIH_CURRENT_LENGTH
Longitud de la versión actual de la estructura de cabecera de información de CICS .

Estado de TaskEnd(MQLONG)
Este campo es un campo de respuesta, que muestra el estado de la transacción de usuario al final de la tarea. El campo sólo se utiliza para transacciones 3270 y su valor inicial es MQCTES_NOSYNC.

Se devuelve uno de los siguientes valores:

MQCTES_NOSYNC
No sincronizado.

La transacción de usuario todavía no se ha completado y no se ha sincronizado. El campo *MsgType* en MQMD es MQMT_REQUEST en este caso.

MQCTES_COMMIT
Confirmar unidad de trabajo.

La transacción de usuario todavía no se ha completado, pero ha sincronizado la primera unidad de trabajo. El campo *MsgType* en MQMD es MQMT_DATAGRAM en este caso.

MQCTES_BACKOUT
Restituir unidad de trabajo.

La transacción de usuario todavía no se ha completado. La unidad de trabajo actual se restituye. El campo *MsgType* en MQMD es MQMT_DATAGRAM en este caso.

MQCTES_ENDTASK
Finalizar tarea.

La transacción de usuario ha finalizado (o ha terminado de forma anómala). El campo *MsgType* en MQMD es MQMT_REPLY en este caso.

TransactionId (MQCHAR4)
Este campo es un campo de solicitud. Su longitud la proporciona MQ_TRANSACTION_ID_LENGTH. El valor inicial de este campo es de cuatro blancos.

Si *LinkType* tiene el valor MQCLT_TRANSACTION, *TransactionId* es el identificador de transacción de la transacción de usuario que se va a ejecutar; especifique un valor no en blanco en este caso.

Si *LinkType* tiene el valor MQCLT_PROGRAM, *TransactionId* es el código de transacción bajo el que se van a ejecutar todos los programas de la unidad de trabajo. Si especifica un valor en blanco, se utiliza el código de transacción predeterminado del puente CICS DPL (CKBP). Si el valor no está en blanco, debe haberlo definido en CICS como una transacción local con un programa inicial que sea CSQCBP00. Este campo sólo se aplica cuando *UOWControl* tiene el valor MQCUOWC_FIRST o MQCUOWC_ONLY.

UOWControl (MQLONG)
Este campo es un campo de solicitud que controla el proceso de unidad de trabajo realizado por el puente CICS . El valor inicial de este campo es MQCUOWC_ONLY.

Puede solicitar al puente que ejecute una sola transacción o uno o varios programas dentro de una unidad de trabajo. El campo indica si el puente CICS inicia una unidad de trabajo, realiza la función solicitada dentro de la unidad de trabajo actual o finaliza la unidad de trabajo comprometiéndola o restituyéndola. Se soportan varias combinaciones, para optimizar los flujos de transmisión de datos.

El valor debe ser uno de los siguientes:

MQCUOWC_ONLY
Inicie la unidad de trabajo, realice la función y, a continuación, confirme la unidad de trabajo.

MQCUOWC_CONTINUE
Datos adicionales para la unidad de trabajo actual (sólo 3270).

MQCUOWC_FIRST
Inicie la unidad de trabajo y realice la función.

MQCUOWC_MIDDLE
Realizar función dentro de la unidad de trabajo actual

MQCUOWC_LAST
Realice la función y, a continuación, confirme la unidad de trabajo.

MQCUOWC_COMMIT
Confirme la unidad de trabajo (sólo DPL).

MQCUOWC_BACKOUT
Restituir la unidad de trabajo (sólo DPL).

Versión (MQLONG)
Este campo es un campo de solicitud. Su valor inicial es MQCIH_VERSION_2.

El valor debe ser uno de los siguientes:

MQCIH_VERSION_1
Version-1 Estructura de cabecera de información de CICS .

MQCIH_VERSION_2
Version-2 Estructura de cabecera de información de CICS .

Los campos que existen sólo en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQCIH_CURRENT_VERSION
Versión actual de la estructura de cabecera de información de CICS .

Valores iniciales y declaraciones de lenguaje para MQCIH

Tabla 484. Valores iniciales de campos en MQCIH para MQCIH

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQCIH_STRUC_ID	'CIH¬'
Version	MQCIH_VERSION_2	2
StrucLength	MQCIH_LENGTH_2	180
Encoding	Ninguna	0
CodedCharSetId	Ninguna	0
Format	MQFMT_NONE	Espacios en blanco
Flags	MQCIH_NONE	0
ReturnCode	MQCRC_OK	0
CompCode	MQCC_OK	0
Reason	MQRC_NONE	0
UOWControl	MQCUOWC_ONLY	273
GetWaitInterval	MQCGWI_DEFAULT	-2
LinkType	MQCLT_PROGRAM	1
OutputDataLength	MQCODL_AS_INPUT	-1
FacilityKeepTime	Ninguna	0
ADSDescriptor	MQCADSD_NONE	0
ConversationalTask	MQCCT_NO	0
TaskEndStatus	MQCTES_NOSYNC	0
Facility	MQCFAC_NONE	Nulos
Function	MQCFUNC_NONE	Espacios en blanco
AbendCode	Ninguna	Espacios en blanco
Authenticator	Ninguna	Espacios en blanco
Reserved1	Ninguna	Espacios en blanco
ReplyToFormat	MQFMT_NONE	Espacios en blanco
RemoteSysId	Ninguna	Espacios en blanco
RemoteTransId	Ninguna	Espacios en blanco
TransactionId	Ninguna	Espacios en blanco

Tabla 484. Valores iniciales de campos en MQCIH para MQCIH (continuación)

Nombre de campo	Nombre de constante	Valor de constante
FacilityLike	Ninguna	Espacios en blanco
AttentionId	Ninguna	Espacios en blanco
StartCode	MQCSC_NONE	Espacios en blanco
CancelCode	Ninguna	Espacios en blanco
NextTransactionId	Ninguna	Espacios en blanco
Reserved2	Ninguna	Espacios en blanco
Reserved3	Ninguna	Espacios en blanco
CursorPosition	Ninguna	0
ErrorOffset	Ninguna	0
InputItem	Ninguna	0
Reserved4	Ninguna	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQCIH_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQCIH MyCIH = {MQCIH_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

MQCMHO-Crear opciones de manejador de mensajes

La tabla siguiente resume los campos de la estructura.

Tabla 485. Campos en MQCMHO		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Options	Opciones	Opciones

Visión general de MQCMHO

Disponibilidad: clientes AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS y WebSphere MQ .

Finalidad: la estructura **MQCMHO** permite a las aplicaciones especificar opciones que controlan cómo se crean los manejadores de mensajes. La estructura es un parámetro de entrada en la llamada **MQCRTMH** .

Juego de caracteres y codificación: Los datos de **MQCMHO** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (**MQENC_NATIVE**).

Campos para MQCMHO

La estructura MQCMHO contiene los campos siguientes; los campos se describen en **orden alfabético**:

Opciones (MQLONG)
Este campo siempre es un campo de entrada. Su valor inicial es MQCMHO_DEFAULT_VALIDATION.

Se puede especificar una de las opciones siguientes:

MQCMHO_VALIDAR

Cuando se llama a **MQSETMP** para establecer una propiedad en este descriptor de mensaje, el nombre de propiedad se valida para asegurarse de que:

- no contiene caracteres no válidos.
- no empieza JMS o usr.JMS excepto para lo siguiente:
  - JMSCorrelationID
  - JMSReplyTo
  - JMSType
  - JMSXGroupID
  - JMSXGroupSeq

  Estos nombres están reservados para las propiedades JMS.
- no es una de las palabras clave siguientes, en cualquier mezcla de mayúsculas o minúsculas:
  - Y
  - BETWEEN (entre)
  - ESCAPE
  - FALSE
  - IN
  - ES

- LIKE
- NO
- NULL
- OR
- TRUE

- no empieza el cuerpo. o raíz. (excepto para Root.MQMD.).

Si la propiedad es MQ-defined (mq. *) y el nombre se reconoce, los campos de descriptor de propiedad se establecen en los valores correctos para la propiedad. Si la propiedad no se reconoce, el campo *Support* del descriptor de propiedad se establece en **MQPD_OPTIONAL**.

MQCMHO_DEFAULT_VALIDATION

Este valor especifica que se produce el nivel predeterminado de validación de los nombres de propiedad.

El nivel predeterminado de validación es equivalente al nivel especificado por **MQCMHO_VALIDATE**.

Este es el valor predeterminado.

MQCMHO_NO_VALIDATION

No se produce ninguna validación en el nombre de propiedad. Consulte la descripción de **MQCMHO_VALIDATE**.

Opción predeterminada: Si ninguna de las opciones anteriores descritas es necesaria, se puede utilizar la opción siguiente:

MQCMHO_NONE

Todas las opciones asumen sus valores predeterminados. Utilice este valor para indicar que no se ha especificado ninguna otra opción. **MQCMHO_NONE** ayuda a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

StrucId (MQCHAR4)
Este campo siempre es un campo de entrada. Su valor inicial es MQCMHO_STRUC_ID.

Este es el identificador de estructura; el valor debe ser:

MQCMHO_STRUC_ID
Identificador para la estructura de opciones de creación de manejadores de mensajes.

Para el lenguaje de programación C, la constante **MQCMHO_STRUC_ID_ARRAY** también está definida; tiene el mismo valor que **MQCMHO_STRUC_ID**, pero es una matriz de caracteres en lugar de una serie.

Versión (MQLONG)
Este campo siempre es un campo de entrada. Su valor inicial es MQCMHO_VERSION_1.

Este es el número de versión de la estructura; el valor debe ser:

MQCMHO_VERSION_1
Version-1 crea la estructura de opciones de manejador de mensajes.

La constante siguiente especifica el número de versión de la versión actual:

MQCMHO_CURRENT_VERSION
Versión actual de la estructura de opciones de creación de manejadores de mensajes.

Valores iniciales y declaraciones de lenguaje para MQCMHO

Tabla 486. Valores iniciales de los campos en MQCMHO

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQCMHO_STRUC_ID	'CMHO'
Version	MQCMHO_VERSION_1	1
Options	MQCMHO_DEFAULT_VALIDATION	0

Notas:

1. En el lenguaje de programación C, la variable de macroMQCMHO_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQCMHO MyCMHO = {MQCMHO_DEFAULT};
```

Declaración C

```
struct tagMQCMHO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of MQCRTMH */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

MQCNO - Opciones de conexión

La tabla siguiente resume los campos de la estructura.

Tabla 487. Campos en MQCNO		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Options	Opciones que controlan la acción de MQCONNX	Opciones
Nota: Los campos restantes se ignoran si *Version* es menor que MQCNO_VERSION_2.		
ClientConnOffset	Desplazamiento de la estructura MQCD para conexión de cliente	ClientConnDesplazamiento
ClientConnPtr	Dirección de la estructura MQCD para conexión de cliente	ClientConnPtr
Nota: Los campos restantes se ignoran si *Version* es menor que MQCNO_VERSION_3.		
ConnTag	Etiqueta de conexión del gestor de colas	ConnTag
Nota: Los campos restantes se ignoran si *Version* es menor que MQCNO_VERSION_4.		
SSLConfigPtr	Dirección de la estructura MQSCO para conexión de cliente	SSLConfigPtr
SSLConfigOffset	Desplazamiento de la estructura MQSCO para conexión de cliente	SSLConfigOffset
Nota: Los campos restantes se ignoran si *Version* es menor que MQCNO_VERSION_5.		
ConnectionId	ID de conexión exclusivo	ConnectionId
SecurityParmsOffset	Parámetros de seguridad	SecurityParmsDesplazamiento
SecurityParmsPtr	Parámetros de seguridad	SecurityParmsPtr

Tareas relacionadas

Utilización de MQCONNX

Visión general de MQCNO

Disponibilidad: todas las versiones excepto MQCNO_VERSION_4: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más clientes MQI de WebSphere MQ conectados a estos sistemas.

Finalidad: la estructura MQCNO permite a la aplicación especificar opciones relacionadas con la conexión con el gestor de colas local. La estructura es un parámetro de entrada/salida en la llamada MQCONNX. Para obtener más información sobre cómo utilizar manejadores compartidos y la llamada MQCONNX, consulte Conexiones compartidas (independientes de hebra) con MQCONNX.

Versión: los archivos de cabecera, COPY e INCLUDE proporcionados para los lenguajes de programación soportados contienen la versión más reciente de MQCNO, pero con el valor inicial del campo *Version* establecido en MQCNO_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1 , la aplicación debe establecer el campo *Version* en el número de versión de la versión necesaria.

Conjunto de caracteres y codificación: Los datos de MQCNO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de WebSphere MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos para MQCNO

La estructura MQCNO contiene los campos siguientes; los campos se describen en **orden alfabético**:

ClientConnDesplazamiento (MQLONG)
ClientConnDesplazamiento es el desplazamiento en bytes de una estructura de definición de canal MQCD desde el inicio de la estructura MQCNO. El desplazamiento puede ser positivo o negativo. Este campo es un campo de entrada con un valor inicial de 0.

Utilice *ClientConnOffset* sólo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como un cliente MQI de WebSphere MQ . Para obtener información sobre cómo utilizar este campo, consulte la descripción del campo *ClientConnPtr* .

Este campo se ignora si *Version* es menor que MQCNO_VERSION_2.

ClientConnPtr (MQPTR)
ClientConnPtr es un campo de entrada. Su valor inicial es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula.

Utilice *ClientConnOffset* y *ClientConnPtr* sólo cuando la aplicación que emite la llamada MQCONNX se ejecute como un cliente MQI de WebSphere MQ . Al especificar uno u otro de estos campos, la aplicación puede controlar la definición del canal de conexión de cliente proporcionando una estructura de definición de canal MQCD que contenga los valores necesarios.

Si la aplicación se ejecuta como un cliente MQI de WebSphere MQ , pero no proporciona una estructura MQCD, se utiliza la variable de entorno MQSERVER para seleccionar la definición de canal. Si no se establece MQSERVER , se utiliza la tabla de canales de cliente.

Si la aplicación no se ejecuta como un cliente MQI de WebSphere MQ , *ClientConnOffset* y *ClientConnPtr* se ignoran.

Si la aplicación proporciona una estructura MQCD, establezca los campos listados en los valores necesarios; los demás campos de MQCD se ignoran. Puede rellenar las series de caracteres con espacios en blanco hasta la longitud del campo, o terminarlas con un carácter nulo. Consulte "Campos" en la página 1036 para obtener más información sobre los campos de la estructura MQCD.

Campo en MQCD	Valor
ChannelName	Nombre de canal.
Version	Número de versión de la estructura. No debe ser menor que MQCD_VERSION_7.
TransportType	Cualquier tipo de transporte soportado.
ModeName	Nombre de modalidad de LU 6.2 .
TpName	Nombre de programa de transacción de LU 6.2 .
SecurityExit	Nombre de la salida de seguridad de canal.
SendExit	Nombre de la salida de envío de canal.
ReceiveExit	Nombre de la salida de recepción de canal.
MaxMsgLength	Longitud máxima en bytes de mensajes que se pueden enviar a través del canal de conexión de cliente.
SecurityUserData	Datos de usuario para salida de seguridad.
SendUserData	Datos de usuario para salida de envío.
ReceiveUserData	Datos de usuario para salida de recepción.
UserIdentifier	Identificador de usuario que se utilizará para establecer una sesión de LU 6.2 .

Campo en MQCD	Valor
Password	Contraseña que se utilizará para establecer una sesión de LU 6.2 .
ConnectionName	Nombre de conexión.
HeartbeatInterval	Tiempo en segundos entre flujos de pulsaciones.
StrucLength	Longitud de la estructura MQCD.
ExitNameLength	Longitud de los nombres de salida a los que se dirigen *SendExitPtr* y *ReceiveExitPtr*. Debe ser mayor que cero si *SendExitPtr* o *ReceiveExitPtr* se establece en un valor que no sea el puntero nulo.
ExitDataLength	Longitud de los datos de salida direccionado por *SendUserDataPtr* y *ReceiveUserDataPtr*. Debe ser mayor que cero si *SendUserDataPtr* o *ReceiveUserDataPtr* se establece en un valor que no sea el puntero nulo.
SendExitsDefined	Número de salidas de envío dirigidas por *SendExitPtr*. Si es cero, *SendExit* y *SendUserData* proporcionan el nombre de salida y los datos. Si es mayor que cero, *SendExitPtr* y *SendUserDataPtr* proporcionan los nombres y datos de salida, y *SendExit* y *SendUserData* deben estar en blanco.
ReceiveExitsDefined	Número de salidas de recepción dirigidas por *ReceiveExitPtr*. Si es cero, *ReceiveExit* y *ReceiveUserData* proporcionan el nombre de salida y los datos. Si es mayor que cero, *ReceiveExitPtr* y *ReceiveUserDataPtr* proporcionan los nombres y datos de salida, y *ReceiveExit* y *ReceiveUserData* deben estar en blanco.
SendExitPtr	Dirección del nombre de la primera salida de envío.
SendUserDataPtr	Dirección de datos para la primera salida de envío.
ReceiveExitPtr	Dirección del nombre de la primera salida de recepción.
ReceiveUserDataPtr	Dirección de datos para la primera salida de recepción.
LongRemoteUserIdLength	Longitud del identificador de usuario remoto largo.
LongRemoteUserIdPtr	Dirección del identificador de usuario remoto largo.
RemoteSecurityId	Identificador de seguridad remota.
SSLCipherSpec	CipherSpecde SSL.
SSLPeerNamePtr	Dirección del nombre de igual SSL.
SSLPeerNameLength	Longitud del nombre de igual SSL.
KeepAliveInterval	Valor pasado a la pila de comunicaciones para la temporización de estado activo para el canal
LocalAddress	La dirección de comunicaciones local, incluida la dirección IP del adaptador de red local que se va a utilizar, y un rango de puertos que se van a utilizar para las conexiones de salida.

Proporcione la estructura de definición de canal de una de estas dos maneras:

- Utilizando el campo de desplazamiento *ClientConnOffset*

  En este caso, la aplicación debe declarar una estructura compuesta que contenga un MQCNO seguido de la estructura de definición de canal MQCD y establecer *ClientConnOffset* en el desplazamiento de la estructura de definición de canal desde el inicio del MQCNO. Asegúrese de que

este desplazamiento sea correcto. *ClientConnPtr* debe establecerse en el puntero nulo o en bytes nulos.

Utilice *ClientConnOffset* para lenguajes de programación que no soportan el tipo de datos de puntero, o que implementan el tipo de datos de puntero de una forma que no es portable a entornos diferentes (por ejemplo, el lenguaje de programación COBOL).

Para el lenguaje de programación Visual Basic, una estructura compuesta denominadaMQCNOCD se proporciona en el archivo de cabecera CMQXB.BAS; esta estructura contiene una estructura MQCNO seguida de una estructura MQCD. Inicialice MQCNOCD invocando la subrutina MQCNOCD_DEFAULTS. MQCNOCD se utiliza con elVariante MQCONNXAny de la llamada MQCONNX; consulte la descripción de la llamada MQCONNX para obtener más detalles.

- Utilizando el campo de puntero *ClientConnPtr*

  En este caso, la aplicación puede declarar la estructura de definición de canal por separado de la estructura MQCNO y establecer *ClientConnPtr* en la dirección de la estructura de definición de canal. Establezca *ClientConnOffset* en cero.

  Utilice *ClientConnPtr* para lenguajes de programación que den soporte al tipo de datos de puntero de una forma que sea portable a distintos entornos (por ejemplo, el lenguaje de programación C).

  En el lenguaje de programación C, puede utilizar la variable de macro MQCD_CLIENT_CONN_DEFAULT para proporcionar valores iniciales para la estructura que son más adecuados para su uso en la llamada MQCONNX que los valores iniciales proporcionados por MQCD_DEFAULT.

Sea cual sea la técnica que elija, sólo puede utilizar uno de *ClientConnOffset* y *ClientConnPtr*; la llamada falla con el código de razón MQRC_CLIENT_CONN_ERROR si ambos son distintos de cero.

Cuando se ha completado la llamada MQCONNX, no se vuelve a hacer referencia a la estructura MQCD.

Este campo se ignora si *Version* es menor que MQCNO_VERSION_2.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada, siendo el valor inicial la serie de bytes totalmente nula.

ConnectionId (MQBYTE24)
ConnectionId es un identificador exclusivo de 24 bytes que permite a WebSphere MQ identificar de forma fiable una aplicación. Una aplicación puede utilizar este identificador para la correlación en llamadas PUT y GET. Este parámetro de salida tiene un valor inicial de 24 bytes nulos en todos los lenguajes de programación.

El gestor de colas asigna un ID exclusivo a todas las conexiones, sin embargo, se establecen. Si un MQCONNX establece la conexión con un MQCNO de la versión 5, la aplicación puede determinar el ConnectionId del MQCNO devuelto. Se garantiza que el identificador asignado es exclusivo entre todos los demás identificadores que genera WebSphere MQ , como por ejemplo CorrelId, MsgIDy GroupId.

Utilice ConnectionId para identificar unidades de trabajo de larga ejecución utilizando el mandato PCF Inquire Connection o el mandato MQSC DISPLAY CONN. El ConnectionId utilizado por los mandatos MQSC (CONN) se deriva del ConnectionId devuelto aquí. Los mandatos Consultar y Detener conexión de PCF pueden utilizar el ConnectionId devuelto aquí sin modificación.

Puede utilizar el ConnectionId para forzar el final de una unidad de trabajo de larga ejecución, especificando el ConnectionId utilizando el mandato PCF Detener conexión o el mandato MQSC STOP CONN. Consulte Detener conexión y STOP CONN para obtener más información sobre cómo utilizar estos mandatos.

Este campo no se devuelve si la versión es inferior a MQCNO_VERSION_5.

La longitud de este campo la proporciona MQ_CONNECTION_ID_LENGTH.

ConnTag (MQBYTE128)
ConnTag es una etiqueta que el gestor de colas asocia con los recursos afectados por la aplicación durante esta conexión. Cada aplicación o instancia de aplicación debe utilizar un valor diferente para la etiqueta, para que el gestor de colas pueda serializar correctamente el acceso a los recursos afectados. Este campo es un campo de entrada y su valor inicial es MQCT_NONE.

Consulte las descripciones de las opciones MQCNO_ * _CONN_TAG_ * para obtener más detalles sobre los valores que deben utilizar las distintas aplicaciones. La etiqueta deja de ser válida cuando la aplicación termina o emite la llamada MQDISC.

Nota: Los valores de etiqueta de conexión que empiezan por MQ en mayúsculas, minúsculas o mayúsculas y minúsculas en ASCII o EBCDIC están reservados para que los utilicen los productos IBM . No utilice valores de etiqueta de conexión que empiecen por estas letras.

Utilice el siguiente valor especial si no necesita ninguna etiqueta:

MQCT_NONE
El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQCT_NONE_ARRAY; esta constante tiene el mismo valor que MQCT_NONE, pero es una matriz de caracteres en lugar de una serie.

Este campo se utiliza al conectarse a un gestor de colas z/OS . En otros entornos, especifique el valor MQCT_NONE.

La longitud de este campo la proporciona MQ_CONN_TAG_LENGTH. Este campo se ignora si *Version* es menor que MQCNO_VERSION_3.

Opciones (MQLONG)
Opciones que controlan la acción de MQCONNX.

Opciones de contabilidad

Las opciones siguientes controlan el tipo de contabilidad si el atributo de gestor de colas *AccountingConnOverride* se establece en MQMON_ENABLED:

MQCNO_ACCOUNTING_MQI_ENABLED
Cuando la recopilación de datos de supervisión se desactiva en la definición del gestor de colas estableciendo el atributo *MQIAccounting* en MQMON_OFF, el establecimiento de este distintivo habilita la recopilación de datos de contabilidad MQI.

MQCNO_ACCOUNTING_MQI_DISABLED
Cuando la recopilación de datos de supervisión se desactiva en la definición del gestor de colas estableciendo el atributo *MQIAccounting* en MQMON_OFF, el establecimiento de este distintivo detiene la recopilación de datos de contabilidad MQI.

MQCNO_ACCOUNTING_Q_ENABLED
Cuando la recopilación de datos de contabilidad de cola se desactiva en la definición del gestor de colas estableciendo el atributo *MQIAccounting* en MQMON_OFF, el establecimiento de este distintivo habilita la recopilación de datos de contabilidad para las colas que especifican un gestor de colas en el campo *MQIAccounting* de su definición de cola.

MQCNO_ACCOUNTING_Q_DISABLED
Cuando la recopilación de datos de contabilidad de colas se desactiva en la definición del gestor de colas estableciendo el atributo *MQIAccounting* en MQMON_OFF, el establecimiento de este distintivo desactiva la recopilación de datos de contabilidad para las colas que especifican un gestor de colas en el campo *MQIAccounting* de su definición de cola.

Si no se ha definido ninguno de estos distintivos, la contabilidad de la conexión es la que se ha definido en los atributos del gestor de colas.

Opciones de enlace

Las opciones siguientes controlan el tipo de enlace de WebSphere MQ que se va a utilizar. Especifique sólo una de estas opciones:

MQCNO_STANDARD_BINDING
La aplicación y el agente del gestor de colas local (el componente que gestiona las operaciones de puesta en cola) se ejecutan en unidades de ejecución separadas (normalmente, en procesos separados). Esta disposición mantiene la integridad del gestor de colas; es decir, protege al gestor de colas de programas errantes.

Si el gestor de colas da soporte a varios tipos de enlace y establece MQCNO_STANDARD_BINDING, el gestor de colas utiliza el atributo *DefaultBindType* en la stanza *Connection* del archivo `qm.ini` (o la entrada de registro Windows equivalente) para seleccionar el tipo real de enlace. Si esta stanza no está definida, o el valor no se puede utilizar o no es adecuado para la aplicación, el gestor de colas selecciona un tipo de enlace adecuado. El gestor de colas establece el tipo de enlace real utilizado en las opciones de conexión.

Utilice MQCNO_STANDARD_BINDING en situaciones en las que es posible que la aplicación no se haya probado completamente o que no sea fiable o no sea fiable. MQCNO_STANDARD_BINDING es el valor predeterminado.

Esta opción está soportada en todos los entornos.

Si se va a enlazar a la biblioteca `mqm`, primero se intentará establecer una conexión de servidor estándar mediante el tipo de enlace predeterminado. Si no se ha podido cargar la biblioteca de servidor subyacente, en su lugar se intenta una conexión de cliente.

- Si se especifica la variable de entorno MQ_CONNECT_TYPE, se puede proporcionar una de las opciones siguientes para cambiar el comportamiento de MQCONN o MQCONNX si se especifica MQCNO_STANDARD_BINDING. La excepción a esto es si se especifica MQCNO_FASTPATH_BINDING con MQ_CONNECT_TYPE establecido en LOCAL o STANDARD, para permitir que el administrador degrade las conexiones de vía de acceso rápida sin que exista un cambio relativo en la aplicación:

Valor	**Significado**
CLIENT	Sólo se intenta una conexión de cliente.
FASTPATH	Este valor estaba soportado en los releases anteriores, pero ahora se pasa por alto si se ha especificado.
LOCAL	Sólo se intenta una conexión con el servidor. Las conexiones fastpath se reducen a una conexión con el servidor estándar.
ESTÁNDAR	Soportado por motivos de compatibilidad con releases anteriores. Este valor se trata ahora como LOCAL.

- Si la variable de entorno MQ_CONNECT_TYPE no se establece cuando se llama a MQCONNX, se intenta una conexión de servidor estándar utilizando el tipo de enlace predeterminado. Si no se puede cargar la biblioteca del servidor, se intenta una conexión de cliente.

MQCNO_FASTPATH_BINDING
La aplicación y el agente del gestor de colas local forman parte de la misma unidad de ejecución. Esto contrasta con el método típico de enlace, donde la aplicación y el agente del gestor de colas local se ejecutan en unidades de ejecución independientes.

MQCNO_FASTPATH_BINDING se ignora si el gestor de colas no da soporte a este tipo de enlace; el proceso continúa como si no se hubiera especificado la opción.

MQCNO_FASTPATH_BINDING puede ser una ventaja en situaciones en las que varios procesos consumen más recursos que el recurso global utilizado por la aplicación.Una aplicación que utiliza el enlace de vía de acceso rápida se conoce como *aplicación de confianza*.

Tenga en cuenta los siguientes puntos importantes al decidir si se debe utilizar el enlace de vía de acceso rápida:

- El uso de la opción MQCNO_FASTPATH_BINDING no impide que una aplicación altere o dañe mensajes y otras áreas de datos pertenecientes al gestor de colas. Utilice esta opción sólo en situaciones en las que haya evaluado completamente estos problemas.
- La aplicación no debe utilizar señales asíncronas o interrupciones de temporizador (como `sigkill`) con MQCNO_FASTPATH_BINDING. También hay restricciones sobre el uso de segmentos de memoria compartida.
- La aplicación debe utilizar la llamada MQDISC para desconectarse del gestor de colas.
- La aplicación debe finalizar antes de finalizar el gestor de colas con el mandato `endmqm` .
- En IBM i, el trabajo debe ejecutarse bajo un perfil de usuario que pertenezca al grupo `QMQMADM` . Además, el programa no debe detenerse de forma anómala, de lo contrario se pueden producir resultados imprevisibles.
- En sistemas UNIX , el identificador de usuario `mqm` debe ser el identificador de usuario efectivo y el identificador de grupo `mqm` debe ser el identificador de grupo efectivo. Para que la aplicación se ejecute de este modo, configure el programa para que sea propiedad del identificador de usuario `mqm` y del identificador de grupo `mqm` y, a continuación, establezca los bits de permiso `setuid` y `setgid` en el programa.

  WebSphere MQ Object Authority Manager (OAM) sigue utilizando el ID de usuario real para la comprobación de autorización.
- En Windows, el programa debe ser miembro del grupo `mqm` . El enlace de vía de acceso rápida no está soportado para aplicaciones de 64 bits.

La opción MQCNO_FASTPATH_BINDING está soportada en los entornos siguientes: AIX, HP-UX, IBM i, Solaris, Linuxy Windows. En z/OS, la opción se acepta pero se ignora.

Para obtener más información sobre las implicaciones de utilizar aplicaciones de confianza, consulte Restricciones para aplicaciones de confianza .

MQCNO_SHARED_BINDING

Con MQCNO_SHARED_BINDING, la aplicación y el agente del gestor de colas local comparten algunos recursos. MQCNO_SHARED_BINDING se omite si el gestor de colas no soporta este tipo de enlace. El proceso continuará como si no se hubiese especificado la opción.

MQCNO_ISOLATED_BINDING

En este caso, el proceso de aplicaciones y el agente del gestor de colas local se aíslan entre sí en el sentido de que no comparten recursos. Si el gestor de colas no da soporte a este tipo de enlace, se ignora MQCNO_ISOLATED_BINDING. El proceso continuará como si no se hubiese especificado la opción.

MQCNO_CLIENT_BINDING

Especifique esta opción para que la aplicación intente únicamente una conexión con el cliente. Esta opción tiene las siguientes limitaciones:

- MQCNO_CLIENT_BINDING se rechaza en z/OS con MQRC_OPTIONS_ERROR.
- MQCNO_CLIENT_BINDING se rechaza con MQRC_OPTIONS_ERROR, si se ha especificado con cualquier opción de enlace MQCNO distinta de MQCNO_STANDARD_BINDING.
- MQCNO_CLIENT_BINDING no está disponible para Java o .NET ya que tienen sus propios mecanismos para elegir el tipo de enlace.
- Si la variable de entorno MQ_CONNECT_TYPE no se establece cuando se llama a MQCONNX, se intenta una conexión de servidor estándar utilizando el tipo de enlace predeterminado. Si no se puede cargar la biblioteca del servidor, se intenta una conexión de cliente.

MQCNO_LOCAL_BINDING

Especifique esta opción para hacer que la aplicación intente una conexión con el servidor. Si también se ha especificado MQCNO_FASTPATH_BINDING, MQCNO_ISOLATED_BINDING o MQCNO_SHARED_BINDING, la conexión será de dicho tipo, y se ha documentado en esta sección. De lo contrario, se intenta efectuar una conexión de servidor estándar utilizando el tipo de enlace predeterminado. MQCNO_LOCAL_BINDING tiene las limitaciones siguientes:

- MQCNO_LOCAL_BINDING se ignora en z/OS.
- MQCNO_LOCAL_BINDING se rechaza con MQRC_OPTIONS_ERROR si se ha especificado con cualquier opción de reconexión MQCNO que no sea MQCNO_RECONNECT_AS_DEF.
- MQCNO_LOCAL_BINDING no está disponible para Java o .NET ya que tienen sus propios mecanismos para elegir el tipo de enlace.
- Si la variable de entorno MQ_CONNECT_TYPE no se establece cuando se llama a MQCONNX, se intenta una conexión de servidor estándar utilizando el tipo de enlace predeterminado. Si no se puede cargar la biblioteca del servidor, se intenta una conexión de cliente.

En AIX, HP-UX, Solaris, Linuxy Windows, puede utilizar la variable de entorno MQ_CONNECT_TYPE con el tipo de enlace especificado por el campo *Options* , para controlar el tipo de enlace utilizado. Si especifica esta variable de entorno, debe tener el valor FASTPATH o STANDARD ; si tiene un valor diferente, se ignora. El valor de la variable de entorno distingue entre mayúsculas y minúsculas; consulte Variable de entorno MQCONNX para obtener más información.

La variable de entorno y el campo *Options* interactúan de la forma siguiente:

- Si omite la variable de entorno, o le da un valor que no está soportado, el uso del enlace de vía de acceso rápida lo determina únicamente el campo *Options* .
- Si proporciona a la variable de entorno un valor soportado, el enlace de vía de acceso rápida sólo se utiliza si *ambos* la variable de entorno y el campo *Options* especifican el enlace de vía de acceso rápida.

Opciones de etiqueta de conexión

Estas opciones solo están soportadas al conectarse a un gestor de colas z/OS y controlan el uso de la etiqueta de conexión *ConnTag*. Sólo puede especificar una de estas opciones:

MQCNO_SERIALIZE_CONN_TAG_Q_MGR

Esta opción solicita el uso exclusivo del código de conexión dentro del gestor de colas local. Si el código de conexión ya está en uso en el gestor de colas local, la llamada MQCONNX falla con el código de razón MQRC_CONN_TAG_IN_USE. El resultado de la llamada no se ve afectado por el uso de la etiqueta de conexión en otro lugar del grupo de compartición de colas al que pertenece el gestor de colas local.

MQCNO_SERIALIZE_CONN_TAG_QSG

Esta opción solicita el uso exclusivo del código de conexión dentro del grupo de compartición de colas al que pertenece el gestor de colas local. Si el código de conexión ya está en uso en el grupo de compartición de colas, la llamada MQCONNX falla con el código de razón MQRC_CONN_TAG_IN_USE.

MQCNO_RESTRICT_CONN_TAG_Q_MGR

Esta opción solicita el uso compartido del código de conexión dentro del gestor de colas local. Si el código de conexión ya está en uso en el gestor de colas local, la llamada MQCONNX puede tener éxito si la aplicación solicitante se ejecuta en el mismo ámbito de proceso que el usuario existente del código. Si no se cumple esta condición, la llamada MQCONNX falla con el código de razón MQRC_CONN_TAG_IN_USE. El resultado de la llamada no se ve afectado por el uso del código de conexión en otro lugar del grupo de compartición de colas al que pertenece el gestor de colas local.

- Las aplicaciones deben ejecutarse dentro del mismo espacio de direcciones MVS para compartir la etiqueta de conexión. Si la aplicación que utiliza la etiqueta de conexión es una aplicación cliente, no se permite MQCNO_RESTRICT_CONN_TAG_Q_MGR.

MQCNO_RESTRICT_CONN_TAG_QSG

Esta opción solicita el uso compartido del código de conexión dentro del grupo de compartición de colas al que pertenece el gestor de colas local. Si el código de conexión ya está en uso en el grupo de compartición de colas, la llamada MQCONNX puede ser satisfactoria siempre que la aplicación solicitante se esté ejecutando en el mismo ámbito de proceso y esté conectada al mismo gestor de colas, que el usuario existente del código.

Si no se cumplen estas condiciones, la llamada MQCONNX falla con el código de razón MQRC_CONN_TAG_IN_USE.

- Las aplicaciones deben ejecutarse dentro del mismo espacio de direcciones MVS para compartir la etiqueta de conexión. Si la aplicación que utiliza la etiqueta de conexión es una aplicación cliente, no se permite MQCNO_RESTRICT_CONN_TAG_QSG.

Si no se especifica ninguna de estas opciones, no se utiliza *ConnTag* . Estas opciones no son válidas si *Version* es menor que MQCNO_VERSION_3.

Opciones de manejo compartido

Estas opciones están soportadas en los entornos siguientes: AIX, HP-UX, IBM i, Solaris, Linux y Windows. Controlan la compartición de manejadores entre diferentes hebras (unidades de proceso paralelo) dentro del mismo proceso. Sólo puede especificar una de estas opciones:

MQCNO_HANDLE_SHARE_NONE

Esta opción indica que la conexión y los descriptores de contexto de objeto sólo pueden ser utilizados por la hebra que ha hecho que se asignara el descriptor de contexto (es decir, la hebra que ha emitido la llamada MQCONN, MQCONNX o MQOPEN). Otras hebras que pertenecen al mismo proceso no pueden utilizar los descriptores de contexto.

MQCNO_HANDLE_SHARE_BLOCK

Esta opción indica que las conexiones y los manejadores de objetos asignados por una hebra de un proceso pueden ser utilizados por otras hebras pertenecientes al mismo proceso. Sin embargo, sólo una hebra a la vez puede utilizar un descriptor de contexto determinado; es decir, sólo se permite el uso en serie de un descriptor de contexto. Si una hebra intenta utilizar un descriptor de contexto que ya está en uso por otra hebra, la llamada se bloquea (espera) hasta que el descriptor de contexto esté disponible.

MQCNO_HANDLE_SHARE_NO_BLOCK

Es lo mismo que MQCNO_HANDLE_SHARE_BLOCK, excepto que si el descriptor de contexto está siendo utilizado por otra hebra, la llamada se completa inmediatamente con MQCC_FAILED y MQRC_CALL_IN_PROGRESS en lugar de bloquearse hasta que el descriptor de contexto esté disponible.

Una hebra puede tener cero o un descriptor de contexto no compartido:

- Cada llamada MQCONN o MQCONNX que especifica MQCNO_HANDLE_SHARE_NONE devuelve un nuevo descriptor de contexto no compartido en la primera llamada, y el mismo descriptor de contexto no compartido en la segunda y posteriores llamadas (suponiendo que no haya ninguna llamada MQDISC interviniente). El código de razón es MQRC_ALREADY_CONNECTED para la segunda llamada y posteriores.
- Cada llamada MQCONNX que especifica MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK devuelve un nuevo descriptor de contexto compartido en cada llamada.

Los descriptores de contexto de objeto heredan las mismas propiedades de compartición que el descriptor de contexto de conexión especificado en la llamada MQOPEN que ha creado el descriptor de contexto de objeto. Además, las unidades de trabajo heredan las mismas propiedades de compartición que el descriptor de conexión utilizado para iniciar la unidad de trabajo; si la unidad de trabajo se inicia en una hebra utilizando un descriptor de contexto compartido, la unidad de trabajo se puede actualizar en otra hebra utilizando el mismo descriptor de contexto.

Si no especifica una opción de uso compartido de descriptor de contexto, el valor predeterminado lo determina el entorno:

- En el entorno de Microsoft Transaction Server (MTS), el valor predeterminado es el mismo que MQCNO_HANDLE_SHARE_BLOCK.
- En otros entornos, el valor predeterminado es el mismo que MQCNO_HANDLE_SHARE_NONE.

Opciones de reconexión

Las opciones de reconexión determinan si una conexión es reconectable. Sólo se pueden volver a conectar las conexiones de cliente.

MQCNO_RECONNECT_AS_DEF

La opción de reconexión se resuelve en su valor predeterminado. Si no se establece ningún valor predeterminado, el valor de esta opción se resuelve en DISABLED . El valor de la opción se pasa al servidor y lo pueden consultar PCF y MQSC.

MQCNO_RECONNECT

La aplicación se puede reconectar a cualquier gestor de colas coherente con el valor del parámetro `QmgrName` de `MQCONNX`. Utilice la opción MQCNO_RECONNECT sólo si no hay afinidad entre la aplicación cliente y el gestor de colas con el que estableció inicialmente una conexión. El valor de la opción se pasa al servidor y lo pueden consultar PCF y MQSC.

MQCNO_RECONNECT_DISABLED

La aplicación no se puede volver a conectar. El valor de la opción no se pasa al servidor.

MQCNO_RECONNECT_Q_MGR

La aplicación sólo se puede volver a conectar al gestor de colas con el que se conectó originalmente. Utilice este valor si un cliente se puede reconectar, pero existe una afinidad entre la aplicación cliente y el gestor de colas con el que estableció originalmente una conexión. Elija este valor si desea que el cliente se reconecte automáticamente a la instancia en espera de un gestor de colas con elevada disponibilidad. El valor de la opción se pasa al servidor y lo pueden consultar PCF y MQSC.

Utilice las opciones MQCNO_RECONNECT, MQCNO_RECONNECT_DISABLED y MQCNO_RECONNECT_Q_MGR sólo para conexiones de cliente. Si las opciones se utilizan para una conexión de enlace, `MQCONNX` falla con el código de terminación `MQCC_FAILED` y el código de razón `MQRC_OPTIONS_ERROR`. La reconexión automática de cliente no está soportada por las clases WebSphere MQ para Java

Opciones de compartición de conversación

Las opciones siguientes sólo se aplican a las conexiones de cliente TCP/IP. Para los canales SNA, SPX y NetBios , estos valores se ignoran y el canal se ejecuta como en las versiones anteriores del producto

MQCNO_NO_CONV_SHARING

Esta opción no permite el uso compartido de conversación.

Puede utilizar MQCNO_NO_CONV_SHARING en situaciones en las que las conversaciones están muy cargadas y, por lo tanto, donde la contención es una posibilidad en el extremo de conexión

de servidor de la instancia de canal en la que existen las conversaciones de compartición. MQCNO_NO_CNV_SHARING se comporta como sharecnv (1) cuando se conecta a un canal que da soporte a la compartición de conversación, y sharecnv (0) cuando se conecta a un canal que no da soporte a la compartición de conversación.

MQCNO_ALL_CONVS_SHARE

Esta opción permite el uso compartido de conversaciones; la aplicación no pone ningún límite en el número de conexiones en la instancia de canal. Esta opción es el valor predeterminado.

Si la aplicación indica que la instancia de canal puede compartir, pero la definición de *SharingConversations* (SHARECNV) en el extremo de conexión de servidor del canal está establecida en uno, no se produce ninguna compartición y no se proporciona ningún aviso a la aplicación.

De forma similar, si la aplicación indica que la compartición está permitida pero la definición de *SharingConversations* de conexión con el servidor está establecida en cero, no se proporciona ningún aviso y la aplicación muestra el mismo comportamiento que un cliente en las versiones del producto anteriores a la versión 7.0; se ignora el valor de la aplicación relacionado con la compartición de conversaciones.

MQCNO_NO_CONV_SHARING y MQCNO_ALL_CONVS_SHARE se excluyen mutuamente. Si se especifican ambas opciones en una conexión determinada, la conexión se rechaza con un código de razón de MQRC_OPTIONS_ERROR.

Opciones de definición de canal

Las opciones siguientes controlan el uso de la estructura de definición de canal pasada en MQCNO:

MQCNO_CD_FOR_OUTPUT_ONLY

Esta opción permite que la estructura de definición de canal en MQCNO sólo se utilice para devolver el nombre de canal utilizado en una llamada MQCONNX satisfactoria.

Si no se proporciona una estructura de definición de canal válida, la llamada falla con el código de razón MQRC_CD_ERROR.

Si la aplicación no se está ejecutando como un cliente, la opción se ignora.

El nombre de canal devuelto puede utilizarse en una llamada MQCONNX posterior utilizando la opción MQCNO_USE_CD_SELECTION para volver a conectarse utilizando la misma definición de canal. Esto puede ser útil cuando hay varias definiciones de canal aplicables en la tabla de canales de cliente.

MQCNO_USE_CD_SELECTION

Esta opción permite que la llamada MQCONNX se conecte utilizando el nombre de canal contenido en la estructura de definición de canal pasada en MQCNO.

Si se establece la variable de entorno MQSERVER, se utiliza la definición de canal definida por la misma. Si no se establece MQSERVER, se utiliza la tabla de canales de cliente.

Si no se encuentra una definición de canal con un nombre de canal y un nombre de gestor de colas coincidentes, la llamada falla con el código de razón MQRC_Q_MGR_NAME_ERROR.

Si no se proporciona una estructura de definición de canal válida, la llamada falla con el código de razón MQRC_CD_ERROR.

Si la aplicación no se está ejecutando como un cliente, la opción se ignora.

Opción predeterminada

Si no necesita ninguna de las opciones descritas anteriormente, puede utilizar la opción siguiente:

MQCNO_NONE

No se especifica ninguna opción.

Utilice MQCNO_NONE para ayudar a la documentación del programa. No está previsto que esta opción se utilice con cualquier otra opción MQCNO_ *, pero debido a que su valor es cero, no se puede detectar dicho uso.

SecurityParmsDesplazamiento (MQLONG)
SecurityParmsDesplazamiento es el desplazamiento en bytes de la estructura MQCSP desde el inicio de la estructura MQCNO. El desplazamiento puede ser positivo o negativo. Este campo es un campo de entrada, con un valor inicial de 0.

Este campo se ignora si *Versión* es menor que MQCNO_VERSION_5.

La estructura MQCSP se define en "MQCSP-Parámetros de seguridad" en la página 313.

SecurityParmsPtr (PMQCSP)
SecurityParmsPtr es la dirección de la estructura MQCSP, utilizada para especificar un ID de usuario y una contraseña para la autenticación por parte del servicio de autorización. Este campo es un campo de entrada y su valor inicial es un puntero nulo o bytes nulos.

Este campo se ignora si *Versión* es menor que MQCNO_VERSION_5.

La estructura MQCSP se define en "MQCSP-Parámetros de seguridad" en la página 313.

SSLConfigOffset (MQLONG)
SSLConfigOffset es el desplazamiento en bytes de una estructura MQSCO desde el inicio de la estructura MQCNO. El desplazamiento puede ser positivo o negativo. Este campo es un campo de entrada, con un valor inicial de 0.

Utilice *SSLConfigOffset* sólo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como un cliente MQI de WebSphere MQ . Para obtener información sobre cómo utilizar este campo, consulte la descripción del campo *SSLConfigPtr* .

Este campo se ignora si *Version* es menor que MQCNO_VERSION_4.

SSLConfigPtr (PMQSCO)
SSLConfigPtr es un campo de entrada. El valor itsinitial es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula.

Utilice *SSLConfigPtr* y *SSLConfigOffset* sólo cuando la aplicación que emite la llamada MQCONNX se esté ejecutando como un cliente MQI de WebSphere MQ y el protocolo de canal sea TCP/IP. Si la aplicación no se ejecuta como un cliente WebSphere MQ , o el protocolo de canal no es TCP/IP, *SSLConfigPtr* y *SSLConfigOffset* se ignoran.

Al especificar *SSLConfigPtr* o *SSLConfigOffset*, más *ClientConnPtr* o *ClientConnOffset*, la aplicación puede controlar el uso de SSL para la conexión de cliente. Cuando la información SSL se especifica de esta forma, las variables de entorno MQSSLKEYR y MQSSLCRYP se ignoran; cualquier información relacionada con SSL en la tabla de definiciones de canal de cliente (CCDT) también se ignora.

La información SSL sólo se puede especificar en:

- La primera llamada MQCONNX del proceso de cliente, o
- Una llamada MQCONNX posterior cuando todas las conexiones SSL/TLS anteriores con el gestor de colas han finalizado utilizando MQDISC.

Estos son los únicos estados en los que se puede inicializar el entorno SSL de todo el proceso. Si se emite una llamada MQCONNX especificando información SSL cuando el entorno SSL ya existe, la información SSL de la llamada se ignora y la conexión se realiza utilizando el entorno SSL existente; la llamada devuelve el código de terminación MQCC_WARNING y el código de razón MQRC_SSL_ALREADY_INITIALIZED en este caso.

Puede proporcionar la estructura MQSCO de la misma forma que la estructura MQCD, ya sea especificando una dirección en *SSLConfigPtr*, o especificando un desplazamiento en *SSLConfigOffset*; consulte la descripción de *ClientConnPtr* para obtener detalles sobre cómo hacerlo. Sin embargo, no puede utilizar más de uno de *SSLConfigPtr* y *SSLConfigOffset*; la llamada falla con el código de razón MQRC_SSL_CONFIG_ERROR. si ambos son distintos de cero.

Una vez completada la llamada MQCONNX, no se vuelve a hacer referencia a la estructura MQSCO.

Este campo se ignora si *Version* es menor que MQCNO_VERSION_4.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

StrucId (MQCHAR4)
StrucId es siempre un campo de entrada. Su valor inicial es MQCNO_STRUC_ID.

El valor debe ser:

MQCNO_STRUC_ID
Identificador de la estructura de opciones de conexión.

Para el lenguaje de programación C, también se define la constante MQCNO_STRUC_ID_ARRAY; esta constante tiene el mismo valor que MQCNO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Versión (MQLONG)
La versión es siempre un campo de entrada. Su valor inicial es MQCNO_VERSION_1.

El valor debe ser uno de los siguientes:

MQCNO_VERSION_1
Estructura de opciones de conexión de Version-1 .

MQCNO_VERSION_2
Estructura de opciones de conexión de Version-2 .

MQCNO_VERSION_3
Estructura de opciones de conexión Version-3 .

MQCNO_VERSION_4
Estructura de opciones de conexión de Version-4 .

MQCNO_VERSION_5
Estructura de opciones de conexión Version-5 .

Esta versión de la estructura MQCNO amplía MQCNO_VERSION_3 en z/OSy MQCNO_VERSION_4 en todas las demás plataformas.

Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQCNO_CURRENT_VERSION
Versión actual de la estructura de opciones de conexión.

Valores iniciales y declaraciones de lenguaje para MQCNO

Tabla 488. Valores iniciales de campos en MQCNO para MQCNO

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQCNO_STRUC_ID	'CNO¬'
Version	MQCNO_VERSION_1	1
Options	MQCNO_NONE	0
ClientConnOffset	Ninguna	0

Tabla 488. Valores iniciales de campos en MQCNO para MQCNO (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
ClientConnPtr	Ninguna	Puntero nulo o bytes nulos
ConnTag	MQCT_NONE	Nulos
SSLConfigPtr	Ninguna	Puntero nulo o bytes nulos
SSLConfigOffset	Ninguna	0
ConnectionId	Ninguna	Puntero nulo o bytes nulos
SecurityParmsOffset	Ninguna	Puntero nulo o bytes nulos
SecurityParmsPtr	Ninguna	Puntero nulo o bytes nulos

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQCNO_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQCNO MyCNO = {MQCNO_DEFAULT};
```

Declaración C

```
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-manager connection 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 */
 };
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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-manager connection 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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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'
  End Type
```

MQCSP-Parámetros de seguridad

La tabla siguiente resume los campos de la estructura.

Tabla 489. Campos en MQCSP		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
AuthenticationType	Tipo de autenticación	AuthenticationType
Reserved1	Necesario para la alineación de puntero en IBM i	Reserved1
CSPUserIdPtr	Dirección del ID de usuario	CSPUserIdPtr
CSPUserIdOffset	Desplazamiento del ID de usuario	CSPUserIdDesplazamiento
CSPUserIdLength	Longitud de ID de usuario	CSPUserIdLongitud
Reserved2	Necesario para la alineación de puntero en IBM i	Reserved2
CSPPasswordPtr	Dirección de contraseña	CSPPasswordPtr
CSPPasswordOffset	Desplazamiento de contraseña	CSPPasswordOffset
CSPPasswordLength	Longitud de contraseña	CSPPasswordLength

Visión general de MQCSP

Disponibilidad: todos los productos WebSphere MQ .

Finalidad: la estructura MQCSP permite al servicio de autorización autenticar un ID de usuario y una contraseña. Especifique la estructura de parámetros de seguridad de conexión MQCSP en una llamada MQCONNX.

Conjunto de caracteres y codificación: los datos de MQCSP deben estar en el juego de caracteres y la codificación del gestor de colas local; estos se proporcionan mediante el atributo de gestor de colas *CodedCharSetId* y MQENC_NATIVE, respectivamente.

Campos para MQCSP

La estructura MQCSP contiene los campos siguientes; los campos se describen en **orden alfabético**:

AuthenticationType (MQLONG)
AuthenticationType es un campo de entrada. Su valor inicial es MQCSP_AUTH_NONE.

Este es el tipo de autenticación que se debe realizar. Los valores válidos son:

MQCSP_AUTH_NONE
No utilice los campos de ID de usuario y contraseña.

MQCSP_AUTH_USER_ID_AND_PWD
Autentique los campos de ID de usuario y contraseña.

CSPPasswordLength (MQLONG)
Este campo es la longitud de la contraseña que se utilizará en la autenticación.

La longitud máxima de la contraseña depende de la plataforma, consulte ID de usuario. Si la longitud de la contraseña es mayor que la longitud máxima permitida, la solicitud de autenticación falla con MQRC_NOT_AUTHORIZED.

Este campo es un campo de entrada. El valor inicial de este campo es 0.

CSPPasswordOffset (MQLONG)
Es el desplazamiento en bytes de la contraseña que se va a utilizar en la autenticación. El desplazamiento puede ser positivo o negativo.

Este es un campo de entrada. El valor inicial de este campo es 0.

CSPPasswordPtr (MQPTR)
Es la dirección en bytes de la contraseña que se va a utilizar en la autenticación.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula. Este campo se ignora si *Version* es menor que MQCNO_VERSION_5.

CSPUserIdLongitud (MQLONG)
Este campo es la longitud del ID de usuario que se utilizará en la autenticación.

La longitud máxima del ID de usuario depende de la plataforma, consulte ID de usuario. Si la longitud del ID de usuario es mayor que la longitud máxima permitida, la solicitud de autenticación falla con MQRC_NOT_AUTHORIZED.

Este campo es un campo de entrada. El valor inicial de este campo es 0.

CSPUserIdDesplazamiento (MQLONG)
Es el desplazamiento en bytes del ID de usuario que se va a utilizar en la autenticación. El desplazamiento puede ser positivo o negativo.

Este es un campo de entrada. El valor inicial de este campo es 0.

CSPUserIdPtr (MQPTR)
Es la dirección en bytes del ID de usuario que se va a utilizar en la autenticación.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula. Este campo se ignora si *Version* es menor que MQCNO_VERSION_5.

Reserved1 (MQBYTE4)
Un campo reservado, necesario para la alineación de puntero en IBM i.

Este es un campo de entrada. El valor inicial de este campo es nulo.

Reserved2 (MQBYTE8)
Un campo reservado, necesario para la alineación de puntero en IBM i.

Este es un campo de entrada. El valor inicial de este campo es nulo.

StrucId (MQCHAR4)
Identificador de estructura.

El valor debe ser:

MQCSP_STRUC_ID
Identificador de la estructura de parámetros de seguridad.

Para el lenguaje de programación C, también se define la constante MQCSP_STRUC_ID_ARRAY; tiene el mismo valor que MQCSP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQCSPSTRUC_ID.

Versión (MQLONG)
Número de versión de la estructura.

El valor debe ser:

MQCSP_VERSION_1
Estructura de parámetros de seguridad de Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQCSP_CURRENT_VERSION
Versión actual de la estructura de parámetros de seguridad.

Siempre es un campo de entrada. El valor inicial de este campo es MQCSP_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQCSP

Tabla 490. Valores iniciales de campos en MQCSP para MQCSP

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQCSP_STRUC_ID	'CSP¬'
Version	MQCSP_VERSION_1	1
AuthenticationType	Ninguna	MQCSP_AUTH_NONE
Reserved1	Ninguna	Serie nula o espacios en blanco
CSPUserIdPtr	Ninguna	Puntero nulo o bytes nulos
CSPUserIdOffset	Ninguna	0
CSPUserIdLength	Ninguna	0
Reserved2	Ninguna	Serie nula o espacios en blanco
CSPPasswordPtr	Ninguna	Puntero nulo o bytes nulos
CSPPasswordOffset	Ninguna	0
CSPPasswordLength	Ninguna	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQCSP_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQCSP MyCSP = {MQCSP_DEFAULT};
```

Declaración C

```
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 */
                                  alignment */
  MQPTR      CSPPasswordPtr;     /* Address of password */
  MQLONG     CSPPasswordOffset;  /* Offset of password */
  MQLONG     CSPPasswordLength;  /* Length of password */
  };
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración de Visual Basic

```
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
```

MQCTLO-Estructura de opciones de devolución de llamada de control

La tabla siguiente resume los campos de la estructura. Estructura que especifica la función de devolución de llamada de control.

Tabla 491. Campos en MQCTLO

Campo	Descripción	Tema
StrucID	Identificador de la estructura	StrucID
Version	Número de versión de la estructura	Versión
Options	Opciones	Opciones
Reserved	Reservado, campo	Opciones
ConnectionArea	Campo para que lo utilice la función de devolución de llamada	ConnectionArea

Visión general de MQCTLO

Disponibilidad: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OSy clientes MQI de WebSphere MQ conectados a estos sistemas. Visión general de la estructura MQCTLO.

Finalidad: la estructura MQCTLO se utiliza para especificar opciones relacionadas con una función de devolución de llamada de control.

La estructura es un parámetro de entrada y salida en la llamada "MQCTL-devoluciones de llamada de control" en la página 658 .

Versión: La versión actual de MQCTLO es MQCTLO_VERSION_1.

Conjunto de caracteres y codificación: los datos de MQCTLO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos para MQCTLO

Lista alfabética de campos para la estructura MQCTLO.

La estructura MQCTLO contiene los campos siguientes; los campos se describen en orden alfabético:

ConnectionArea (MQPTR)
Estructura de opciones de control-Campo ConnectionArea

Este es un campo que está disponible para que lo utilice la función de devolución de llamada.

El gestor de colas no toma decisiones basadas en el contenido de este campo y se pasa sin cambios al campo "ConnectionArea (MQPTR)" en la página 264 en la estructura MQCBC, que es un parámetro de entrada para la devolución de llamada.

Este campo se ignora para todas las operaciones que no sean MQOP_START y MQOP_START_WAIT.

Es un campo de entrada y salida para la función de devolución de llamada. El valor inicial de este campo es un puntero nulo o bytes nulos.

Opciones (MQLONG)
Estructura de opciones de control-Campo Opciones

Opciones que controlan la acción de MQCTL.

MQCTLO_FAIL_IF_QUIESCING
Forzar que la llamada MQCTL falle si el gestor de colas o la conexión está en estado de desactivación temporal.

Especifique MQGMO_FAIL_IF_QUIESCING, en las opciones MQGMO pasadas en la llamada MQCB, para que se notifique a los consumidores de mensajes cuando estén inmovilizados.

MQCTLO_THREAD_AFFINITY
Esta opción informa al sistema de que la aplicación requiere que todos los consumidores de mensajes, para la misma conexión, se llamen en la misma hebra. Esta hebra se utilizará para todas las invocaciones de los consumidores hasta que se detenga la conexión.

Opción predeterminada: Si no necesita ninguna de las opciones descritas, utilice la opción siguiente:

MQCTLO_NONE
Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados. MQCTLO_NONE está definido para ayudar a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Este es un campo de entrada. El valor inicial del campo *Options* es MQCTLO_NONE.

Reservado (MQLONG)

Este es un campo reservado. El valor debe ser cero.

StrucId (MQCHAR4)
Estructura de opciones de control-Campo StrucId

Este es el identificador de estructura; el valor debe ser:

MQCTLO_STRUC_ID
Identificador de la estructura de opciones de control.

Para el lenguaje de programación C, también se define la constante MQCTLO_STRUC_ID_ARRAY; tiene el mismo valor que MQCTLO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQCTLO_STRUC_ID.

Versión (MQLONG)
Estructura de opciones de control-Campo Versión

Este es el número de versión de la estructura; el valor debe ser:

MQCTLO_VERSION_1
Version-1 Estructura de opciones de control.

La constante siguiente especifica el número de versión de la versión actual:

MQCTLO_CURRENT_VERSION
Versión actual de la estructura de opciones de control.

Siempre es un campo de entrada. El valor inicial de este campo es MQCTLO_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQCTLO
Estructura de opciones de control-Valores iniciales

Tabla 492. Valores iniciales de campos en MQCTLO

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQCTLO_STRUC_ID	'CTLO'
Version	MQCTLO_VERSION_1	1
Options	MQCTLO_NONE	Nulos

Tabla 492. Valores iniciales de campos en MQCTLO (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
Reserved	Reservado, campo	
ConnectionArea	Ninguna	Puntero nulo o bytes nulos

Notas:

1. En el lenguaje de programación C, la variable de macroMQCTLO_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQCTLO MyCTLO = {MQCTLO_DEFAULT};
```

Declaración C
Estructura de opciones de control-Declaración de lenguaje C

```
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 */
 };
```

declaración COBOL

```
** 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
```

Declaración PL/I

```
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 */
```

MQDH - Cabecera de distribución

La tabla siguiente resume los campos de la estructura.

Tabla 493. Campos en MQDH		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
StrucLength	Longitud de estructura MQDH más los registros siguientes	StrucLength
Encoding	Codificación numérica de los datos que siguen a la matriz de registros MQPMR	Codificación
CodedCharSetId	Identificador del juego de caracteres de los datos que siguen a la matriz de registros MQPMR	CodedCharSetId
Format	Nombre del formato de los datos que siguen a la matriz de registros MQPMR	Formato
Flags	Distintivos generales	Distintivos
PutMsgRecFields	Distintivos que indican qué campos de MQPMR están presentes	PutMsgRecFields
RecsPresent	Número de registros de objeto presentes	RecsPresent
ObjectRecOffset	Desplazamiento del primer registro de objeto desde el principio de MQDH	Desplazamiento deObjectRec
PutMsgRecOffset	Desplazamiento del primer registro de transferencia de mensaje desde el principio de MQDH	PutMsgRecOffset

Visión general de MQDH

Disponibilidad: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más clientes WebSphere MQ conectados a estos sistemas.

Finalidad: la estructura MQDH describe los datos adicionales que están presentes en un mensaje cuando ese mensaje es un mensaje de lista de distribución almacenado en una cola de transmisión. Un mensaje de lista de distribución es un mensaje que se envía a varias colas de destino. Los datos adicionales constan de la estructura MQDH seguida de una matriz de registros MQOR y una matriz de registros MQPMR.

Esta estructura la utilizan aplicaciones especializadas que colocan mensajes directamente en colas de transmisión o que eliminan mensajes de las colas de transmisión (por ejemplo: agentes de canal de mensajes).

Las aplicaciones que deseen colocar mensajes en listas de distribución no deben utilizar esta estructura. En su lugar, deben utilizar la estructura MQOD para definir los destinos en la lista de distribución, y la estructura MQPMO para especificar propiedades de mensaje o recibir información sobre los mensajes enviados a los destinos individuales.

Nombre de formato: MQFMT_DIST_HEADER.

Conjunto de caracteres y codificación: los datos de MQDH deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE.

Establezca el juego de caracteres y la codificación de la MQDH en los campos *CodedCharSetId* y *Encoding* en:

- El MQMD (si la estructura MQDH está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQDH (todos los demás casos).

Uso: Cuando una aplicación coloca un mensaje en una lista de distribución, y algunos o todos los destinos son remotos, el gestor de colas prefija los datos del mensaje de aplicación con las estructuras MQXQH y MQDH, y coloca el mensaje en la cola de transmisión relevante. Por lo tanto, los datos se producen en la secuencia siguiente cuando el mensaje está en una cola de transmisión:

- estructura MQXQH
- Estructura MQDH más matrices de registros MQOR y MQPMR
- Datos de mensaje de aplicación

En función de los destinos, el gestor de colas puede generar más de un mensaje de este tipo y colocarlo en colas de transmisión diferentes. En este caso, las estructuras MQDH de esos mensajes identifican distintos subconjuntos de los destinos definidos por la lista de distribución abierta por la aplicación.

Una aplicación que coloca un mensaje de lista de distribución directamente en una cola de transmisión debe ajustarse a la secuencia descrita anteriormente y debe asegurarse de que la estructura MQDH sea correcta. Si la estructura MQDH no es válida, el gestor de colas puede fallar la llamada MQPUT o MQPUT1 con el código de razón MQRC_DH_ERROR.

Puede almacenar mensajes en una cola en formato de lista de distribución sólo si ha definido la cola como capaz de dar soporte a los mensajes de lista de distribución (consulte el atributo de cola *DistLists* que se describe en "Atributos para colas" en la página 816). Si una aplicación coloca un mensaje de lista de distribución directamente en una cola que no da soporte a listas de distribución, el gestor de colas divide el mensaje de lista de distribución en mensajes individuales y, en su lugar, coloca los mensajes en la cola.

Campos para MQDH

La estructura MQDH contiene los campos siguientes; los campos se describen en **orden alfabético**:

CodedCharSetId (MQLONG)

Es el identificador de juego de caracteres de los datos que siguen a las matrices de registros MQOR y MQPMR; no se aplica a los datos de caracteres de la propia estructura MQDH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Puede utilizar el siguiente valor especial:

MQCCSI_INHERIT
Hereda el identificador de juego de caracteres de esta estructura.

Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Siempre que no se produzca ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

No puede utilizar MQCCSI_INHERIT si el valor del campo *PutApplType* en MQMD es MQAT_BROKER.

Este valor está soportado en los entornos siguientes: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más clientes WebSphere MQ conectados a estos sistemas.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Encoding (MQLONG)

Es la codificación numérica de los datos que siguen a las matrices de registros MQOR y MQPMR; no se aplica a los datos numéricos de la propia estructura MQDH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es 0.

Flags (MQLONG)

Puede especificar el distintivo siguiente:

MQDHF_NEW_MSG_IDS
Genere un nuevo identificador de mensaje para cada destino de la lista de distribución. Establézcalo sólo cuando no haya registros de colocación de mensajes o cuando los registros estén presentes pero no contengan el campo *MsgId* .

El uso de este distintivo difiere la generación de los identificadores de mensaje hasta el momento en que el mensaje de lista de distribución se divide finalmente en mensajes individuales. Esto minimiza la cantidad de información de control que debe fluir con el mensaje de lista de distribución.

Cuando una aplicación coloca un mensaje en una lista de distribución, el gestor de colas establece MQDHF_NEW_MSG_IDS en la MQDH que genera cuando se cumplen las dos condiciones siguientes:

- No hay registros de colocación de mensajes proporcionados por la aplicación, o los registros proporcionados no contienen el campo *MsgId* .
- El campo *MsgId* en MQMD es MQMI_NONE, o el campo *Options* en MQPMO incluye MQPMO_NEW_MSG_ID

Si no se necesitan distintivos, especifique lo siguiente:

MQDHF_NONE
No se han especificado distintivos. MQDHF_NONE está definido para ayudar a la documentación del programa. No se pretende que esta constante se utilice con ninguna otra, pero como su valor es cero, tal uso no se puede detectar.

El valor inicial de este campo es MQDHF_NONE.

Format (MQCHAR8)

Es el nombre de formato de los datos que siguen a las matrices de registros MQOD y MQPMR (lo que ocurra en último lugar).

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *Format* en MQMD.

El valor inicial de este campo es MQFMT_NONE.

Desplazamiento de ObjectRec(MQLONG)

Esto proporciona el desplazamiento en bytes del primer registro de la matriz de registros de objeto MQOR que contiene los nombres de las colas de destino. Hay *RecsPresent* registros en esta matriz. Estos registros (más los bytes omitidos entre el primer registro de objeto y el campo anterior) se incluyen en la longitud proporcionada por el campo *StrucLength* .

Una lista de distribución siempre debe contener al menos un destino, por lo que *ObjectRecOffset* siempre debe ser mayor que cero.

El valor inicial de este campo es 0.

PutMsgRecFields (MQLONG)

Puede especificar ninguno o varios de los distintivos siguientes:

MQPMRF_MSG_ID
El campo de identificador de mensaje está presente.

ID MQPMRF_CORREL_ID
El campo de identificador de correlación está presente.

MQPMRF_ID_grupo
El campo identificador de grupo está presente.

MQPMRF_FEEDBACK
El campo de comentarios está presente.

MQPMRF_ACCOUNTING_TOKEN
El campo de señal de contabilidad está presente.

Si no hay campos MQPMR presentes, especifique lo siguiente:

MQPMRF_NONE
No hay campos de registro de colocación de mensaje. MQPMRF_NONE está definido para ayudar a la documentación del programa. No se pretende que esta constante se utilice con ninguna otra, pero como su valor es cero, tal uso no se puede detectar.

El valor inicial de este campo es MQPMRF_NONE.

PutMsgRecOffset (MQLONG)

Esto proporciona el desplazamiento en bytes del primer registro de la matriz de registros de mensajes de colocación MQPMR que contienen las propiedades del mensaje. Si está presente, hay *RecsPresent* registros en esta matriz. Estos registros (más los bytes omitidos entre el primer registro de mensaje de colocación y el campo anterior) se incluyen en la longitud proporcionada por el campo *StrucLength* .

Los registros de mensajes de colocación son opcionales; si no se proporcionan registros, *PutMsgRecOffset* es cero y *PutMsgRecFields* tiene el valor MQPMRF_NONE.

El valor inicial de este campo es 0.

RecsPresent (MQLONG)

Es el número de destinos. Una lista de distribución siempre debe contener al menos un destino, por lo que *RecsPresent* siempre debe ser mayor que cero.

El valor inicial de este campo es 0.

StrucId (MQCHAR4)

El valor debe ser:

MQDH_STRUC_ID
Identificador de la estructura de cabecera de distribución.

Para el lenguaje de programación C, también se define la constante MQDH_STRUC_ID_ARRAY; tiene el mismo valor que MQDH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQDH_STRUC_ID.

StrucLength (MQLONG)

Es el número de bytes desde el inicio de la estructura MQDH hasta el inicio de los datos de mensaje que siguen a las matrices de registros MQOR y MQPMR. Los datos se producen en la secuencia siguiente:

- estructura MQDH
- Matriz de registros MQOR
- Matriz de registros MQPMR
- Datos de mensaje

Las matrices de registros MQOR y MQPMR se direccionan mediante desplazamientos contenidos en la estructura MQDH. Si estos desplazamientos dan como resultado bytes no utilizados entre una o más de la estructura MQDH, las matrices de registros y los datos de mensaje, dichos bytes no utilizados deben incluirse en el valor de *StrucLength*, pero el gestor de colas no conserva el contenido de dichos bytes. Es válido que la matriz de registros MQPMR preceda a la matriz de registros MQOR.

El valor inicial de este campo es 0.

Versión (MQLONG)

El valor debe ser:

MQDH_VERSION_1
Número de versión para la estructura de cabecera de distribución.

La constante siguiente especifica el número de versión de la versión actual:

MQDH_CURRENT_VERSION
Versión actual de la estructura de cabecera de distribución.

El valor inicial de este campo es MQDH_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQDH

Tabla 494. Valores iniciales de los campos en MQDH para MQDH

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQDH_STRUC_ID	'DH¬¬'
Version	MQDH_VERSION_1	1
StrucLength	Ninguna	0
Encoding	Ninguna	0
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Espacios en blanco
Flags	MQDHF_NONE	0
PutMsgRecFields	MQPMRF_NONE	0
RecsPresent	Ninguna	0
ObjectRecOffset	Ninguna	0
PutMsgRecOffset	Ninguna	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQDH_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQDH MyDH = {MQDH_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración de Visual Basic

```
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
```

MQDLH-Cabecera de mensaje no entregado

La tabla siguiente resume los campos de la estructura.

Tabla 495. Campos en MQDLH		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Reason	Mensaje de razón ha llegado a cola de mensajes no entregados	Razón
DestQName	Nombre de la cola de destino original	DestQName
DestQMgrName	Nombre del gestor de colas de destino original	DestQMgrName
Encoding	Codificación numérica de los datos que siguen a MQDLH	Codificación
CodedCharSetId	Identificador del juego de caracteres de los datos que siguen a MQDLH	CodedCharSetId
Format	Nombre del formato de los datos que siguen a MQDLH	Formato
PutApplType	Tipo de aplicación que transfiere el mensaje a la cola de mensajes no entregados	PutApplType
PutApplName	Nombre de la aplicación que transfiere el mensaje a la cola de mensajes no entregados	PutApplName
PutDate	Fecha en que el mensaje se transfirió a la cola de mensajes no entregados	PutDate
PutTime	Hora en que el mensaje se transfirió a la cola de mensajes no entregados	PutTime

Visión general de MQDLH

Disponibilidad: todas las plataformas WebSphere MQ .

Finalidad: La estructura MQDLH describe la información que prefija los datos de mensaje de aplicación de los mensajes en la cola de mensajes no entregados (undelivered-message). Un mensaje puede llegar a la cola de mensajes no entregados porque el gestor de colas o el agente de canal de mensajes lo ha redirigido a la cola, o porque una aplicación ha colocado el mensaje directamente en la cola.

Nombre de formato: MQFMT_DEAD_LETTER_HEADER.

Conjunto de caracteres y codificación: los campos de la estructura MQDLH están en el juego de caracteres y la codificación proporcionados por los campos *CodedCharSetId* y *Encoding* . Estos se especifican en la estructura de cabecera que precede a la MQDLH, o en la estructura MQMD si la MQDLH está al inicio de los datos del mensaje de aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de cola.

Si está utilizando las clases WMQ para Java/JMS, y la página de códigos definida en MQMD no está soportada por la máquina virtual Java, la MQDLH se escribe en el juego de caracteres UTF-8 .

Uso: las aplicaciones que colocan mensajes directamente en la cola de mensajes no entregados deben anteponer a los datos del mensaje una estructura MQDLH e inicializar los campos con los valores

adecuados. Sin embargo, el gestor de colas no requiere que esté presente una estructura MQDLH, o que se hayan especificado valores válidos para los campos.

Si un mensaje es demasiado largo para colocarlo en la cola de mensajes no entregados, la aplicación debe realizar una de las acciones siguientes:

- Trunque los datos del mensaje para que quepan en la cola de mensajes no entregados.
- Anote el mensaje en el almacenamiento auxiliar y coloque un mensaje de informe de excepción en la cola de mensajes no entregados que lo indique.
- Descarte el mensaje y devuelva un error a su originador. Si el mensaje es (o puede ser) un mensaje crítico, hágalo sólo si se sabe que el originador todavía tiene una copia del mensaje; por ejemplo, un mensaje recibido por un agente de canal de mensajes de un canal de comunicación.

Cuál de los anteriores es apropiado (si lo hay) depende del diseño de la aplicación.

El gestor de colas realiza un proceso especial cuando un mensaje que es un segmento se coloca con una estructura MQDLH en la parte frontal; consulte la descripción de la estructura MQMDE para obtener más detalles.

Colocación de mensajes en la cola de mensajes no entregados: cuando se coloca un mensaje en la cola de mensajes no entregados, la estructura MQMD utilizada para la llamada MQPUT o MQPUT1 debe ser idéntica a la MQMD asociada con el mensaje (normalmente la MQMD devuelta por la llamada MQGET), con la excepción de lo siguiente:

- Establezca los campos *CodedCharSetId* y *Encoding* en cualquier juego de caracteres y codificación que se utilicen para los campos de la estructura MQDLH.
- Establezca el campo *Format* en MQFMT_DEAD_LETTER_HEADER para indicar que los datos empiezan por una estructura MQDLH.
- Establezca los campos de contexto (*AccountingToken*, *ApplIdentityData* , *ApplOriginData*, *PutApplName*, *PutApplType* , *PutDate*, *PutTime*, *UserIdentifier*) utilizando una opción de contexto adecuada a las circunstancias:
  - Una aplicación que coloca en la cola de mensajes no entregados un mensaje que no está relacionado con ningún mensaje anterior debe utilizar la opción MQPMO_DEFAULT_CONTEXT; esto hace que el gestor de colas establezca todos los campos de contexto del descriptor de mensaje en sus valores predeterminados.
  - Una aplicación de servidor que pone en la cola de mensajes no entregados un mensaje que acaba de recibir debe utilizar la opción MQPMO_PASS_ALL_CONTEXT para conservar la información de contexto original.
  - Una aplicación de servidor que pone en la cola de mensajes no entregados una *respuesta* a un mensaje que acaba de recibir debe utilizar la opción MQPMO_PASS_IDENTITY_CONTEXT; esto conserva la información de identidad pero establece la información de origen como la de la aplicación de servidor.
  - Un agente de canal de mensajes que pone en la cola de mensajes no entregados un mensaje que ha recibido de su canal de comunicación debe utilizar la opción MQPMO_SET_ALL_CONTEXT para conservar la información de contexto original.

En la propia estructura MQDLH, establezca los campos como se indica a continuación:

- Establezca los campos *CodedCharSetId*, *Encoding*y *Format* en los valores que describen los datos que siguen a la estructura MQDLH, normalmente los valores del descriptor de mensaje original.
- Establezca los campos de contexto *PutApplType*, *PutApplName*, *PutDate*y *PutTime* en valores adecuados para la aplicación que coloca el mensaje en la cola de mensajes no entregados; estos valores no están relacionados con el mensaje original.
- Establezca otros campos según corresponda.

Asegúrese de que todos los campos tienen valores válidos y de que los campos de caracteres se rellenan con espacios en blanco hasta la longitud definida del campo; no finalice los datos de caracteres de forma prematura utilizando un carácter nulo, porque el gestor de colas no convierte los caracteres nulos y posteriores en blancos en la estructura MQDLH.

Obtener mensajes de la cola de mensajes no entregados: las aplicaciones que obtienen mensajes de la cola de mensajes no entregados deben verificar que los mensajes empiezan con una estructura MQDLH. La aplicación puede determinar si una estructura MQDLH está presente examinando el campo *Format* en el descriptor de mensaje MQMD; si el campo tiene el valor MQFMT_DEAD_LETTER_HEADER, los datos del mensaje empiezan por una estructura MQDLH. Tenga en cuenta también que los mensajes que las aplicaciones obtienen de la cola de mensajes no entregados pueden truncarse si originalmente eran demasiado largos para la cola.

Campos para MQDLH

La estructura MQDLH contiene los campos siguientes; los campos se describen en **orden alfabético**:

CodedCharSetId (MQLONG)
CodedCharSetId es el identificador de juego de caracteres de los datos que fluyen a través de la estructura MQDLH (normalmente los datos del mensaje original); no se aplica a los datos de tipo carácter de la propia estructura MQDLH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

MQCCSI_INHERIT
Los datos de caracteres de los datos que siguen a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

No puede utilizar MQCCSI_INHERIT si el valor del campo *PutApplType* en MQMD es MQAT_BROKER.

Este valor está soportado en los entornos siguientes: AIX, HP-UX, z/OS, IBM i, Solaris, Linux, Windows, además de clientes MQI de WebSphere MQ conectados a estos sistemas.

El valor inicial de este campo es MQCCSI_UNDEFINED.

DestQMgrNombre (MQCHAR48)
DestQMgrNombre es el nombre del gestor de colas que era el destino original del mensaje.

La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

DestQName (MQCHAR48)
DestQName es el nombre de la cola de mensajes que era el destino original del mensaje.

La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

Encoding (MQLONG)
La codificación es la codificación numérica de los datos que siguen a la estructura MQDLH (normalmente los datos del mensaje original); no se aplica a los datos numéricos de la propia estructura MQDLH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es 0.

Format (MQCHAR8)
El formato es el nombre de formato de los datos que siguen a la estructura MQDLH (normalmente los datos del mensaje original).

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que las reglas para codificar el campo *Format* en MQMD.

La longitud de este campo la proporciona MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

PutApplNombre (MQCHAR28)
PutApplNombre es el nombre de la aplicación que ha colocado el mensaje en la cola de mensajes no entregados (undelivered-message).

El formato del nombre depende del campo *PutApplType* . El formato puede variar de un release a otro. Consulte la descripción del campo *PutApplName* en "MQMD - Descriptor de mensaje" en la página 392.

Si el gestor de colas redirige el mensaje a la cola de mensajes no entregados, *PutApplName* contiene los primeros 28 caracteres del nombre del gestor de colas, rellenados con espacios en blanco si es necesario.

La longitud de este campo la proporciona MQ_PUT_APPL_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 28 caracteres en blanco en otros lenguajes de programación.

PutApplType (MQLONG)
PutApplTipo es el tipo de aplicación que coloca el mensaje en la cola de mensajes no entregados.

Este campo tiene el mismo significado que el campo *PutApplType* en el descriptor de mensaje MQMD (consulte "MQMD - Descriptor de mensaje" en la página 392 para obtener detalles).

Si el gestor de colas redirige el mensaje a la cola de mensajes no entregados, *PutApplType* tiene el valor MQAT_QMGR.

El valor inicial de este campo es `0`.

PutDate (MQCHAR8)
PutDate es la fecha en la que el mensaje se colocó en la cola de mensajes no entregados.

El formato utilizado para la fecha en la que el gestor de colas genera este campo es:

- AAAAMMDD

donde los caracteres representan:

AAAA
año (cuatro dígitos numéricos)

MM
mes del año (01 a 12)

DD
día del mes (01 a 31)

La hora media de Greenwich (GMT) se utiliza para los campos *PutDate* y *PutTime* , sujeto a que el reloj del sistema se establezca correctamente en GMT.

La longitud de este campo la proporciona MQ_PUT_DATE_LENGTH. El valor inicial de este campo es la serie nula en C y ocho caracteres en blanco en otros lenguajes de programación.

PutTime (MQCHAR8)
PutTime es la hora a la que se ha colocado el mensaje en la cola de mensajes no entregados.

El formato utilizado para la hora en que el gestor de colas genera este campo es:

- HHMMSSTH

donde los caracteres representan:

HH
horas (de 00 a 23)

MM
minutos (de 00 a 59)

SS
segundos (de 00 a 59; consulte la nota)

T
décimas de segundo (de 0 a 9)

H
centésimas de segundo (0 a 9)

Nota: Si el reloj del sistema está sincronizado con un estándar de tiempo muy preciso, es posible que en raras ocasiones se devuelvan 60 o 61 durante los segundos en *PutTime*. Esto sucede cuando se insertan segundos bisiestos en el estándar de tiempo global.

La hora media de Greenwich (GMT) se utiliza para los campos *PutDate* y *PutTime* , sujeto a que el reloj del sistema se establezca correctamente en GMT.

La longitud de este campo la proporciona MQ_PUT_TIME_LENGTH. El valor inicial de este campo es la serie nula en C y ocho caracteres en blanco en otros lenguajes de programación.

Razón (MQLONG)
El campo Razón identifica la razón por la que el mensaje se ha colocado en la cola de mensajes no entregados en lugar de en la cola de destino original.

Esto identifica la razón por la que el mensaje se ha colocado en la cola de mensajes no entregados en lugar de en la cola de destino original. Debe ser uno de los valores MQFB_ * o MQRC_ * (por ejemplo, MQRC_Q_FULL). Consulte la descripción del campo *Feedback* en “MQMD - Descriptor de mensaje” en la página 392 para obtener detalles de los valores MQFB_ * comunes que se pueden producir.

Si el valor está en el rango de MQFB_IMS_FIRST a MQFB_IMS_LAST, el código de error IMS real se puede determinar restando MQFB_IMS_ERROR del valor del campo *Reason* .

Algunos valores MQFB_ * sólo aparecen en este campo. Están relacionados con mensajes de repositorio, mensajes desencadenantes o mensajes de cola de transmisión que se han transferido a la cola de mensajes no entregados. Son las siguientes:

MQFB_APPL_CANNOT_BE_STARTED (X'00000109')
Una aplicación que procesa un mensaje desencadenante no puede iniciar la aplicación especificada en el campo *ApplId* del mensaje desencadenante (consulte “MQTM-Mensaje de desencadenante” en la página 577).

En z/OS, la transacción CKTI CICS es un ejemplo de una aplicación que procesa mensajes desencadenantes.

MQFB_APPL_TYPE_ERROR (X'0000010B')
Una aplicación que procesa un mensaje desencadenante no puede iniciar la aplicación porque el campo *ApplType* del mensaje desencadenante no es válido (consulte “MQTM-Mensaje de desencadenante” en la página 577).

En z/OS, la transacción CICS CKTI es un ejemplo de una aplicación que procesa mensajes desencadenantes.

MQFB_BIND_OPEN_CLUSRCVR_DEL (X'00000119')
El mensaje estaba en SYSTEM.CLUSTER.TRANSMIT.QUEUE está pensado para una cola de clúster que se ha abierto con la opción MQOO_BIND_ON_OPEN, pero el canal de clúster receptor remoto que se debe utilizar para transmitir el mensaje a la cola de destino se ha suprimido antes de que se pudiera enviar el mensaje. Debido a que se ha especificado MQOO_BIND_ON_OPEN, sólo puede utilizarse el canal seleccionado cuando se abrió la cola para transmitir el mensaje. Como este canal ya no está disponible, el mensaje se coloca en la cola de mensajes no entregados.

MQFB_NOT_A_REPOSITORY_MSG (X'00000118')
El mensaje no es un mensaje de repositorio.

MQFB_STOPPED_BY_CHAD_EXIT (X'00000115')
La salida de definición automática de canal ha detenido el mensaje.

MQFB_STOPPED_BY_MSG_EXIT (X'0000010D')
El mensaje ha sido detenido por la salida de mensajes de canal.

MQFB_TM_ERROR (X'0000010A')
El campo *Format* en MQMD especifica MQFMT_TRIGGER, pero el mensaje no empieza por una estructura MQTM válida. Por ejemplo, es posible que el captador de atención mnemotécnico de *StrucId* no sea válido, que no se reconozca *Version* o que la longitud del mensaje desencadenante sea insuficiente para contener la estructura MQTM.

En z/OS, la transacción CICS de CKTI es un ejemplo de una aplicación que procesa mensajes desencadenantes y puede generar este código de comentarios.

MQFB_XMIT_Q_MSG_ERROR (X'0000010F')
Un agente de canal de mensajes ha detectado que un mensaje de la cola de transmisión no tiene el formato correcto. El agente de canal de mensajes coloca el mensaje en la cola de mensajes no entregados utilizando este código de retorno.

El valor inicial de este campo es MQRC_NONE.

StrucId (MQCHAR4)
StrucId es el identificador de estructura.

El valor debe ser:

MQDLH_STRUC_ID
Identificador de la estructura de cabecera de mensaje no entregado.

Para el lenguaje de programación C, también se define la constante MQDLH_STRUC_ID_ARRAY; tiene el mismo valor que MQDLH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQDLH_STRUC_ID.

Versión (MQLONG)
La versión es el número de versión de la estructura.

El valor debe ser:

MQDLH_VERSION_1
Número de versión para la estructura de cabecera de mensaje no entregado.

La constante siguiente especifica el número de versión de la versión actual:

MQDLH_CURRENT_VERSION
Versión actual de la estructura de cabecera de mensaje no entregado.

El valor inicial de este campo es MQDLH_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQDLH

Tabla 496. Valores iniciales de campos en MQDLH para MQDLH

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQDLH_STRUC_ID	'DLH¬'
Version	MQDLH_VERSION_1	1
Reason	MQRC_NONE	0
DestQName	Ninguna	Serie nula o espacios en blanco
DestQMgrName	Ninguna	Serie nula o espacios en blanco

Tabla 496. Valores iniciales de campos en MQDLH para MQDLH (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
Encoding	Ninguna	0
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Espacios en blanco
PutApplType	Ninguna	0
PutApplName	Ninguna	Serie nula o espacios en blanco
PutDate	Ninguna	Serie nula o espacios en blanco
PutTime	Ninguna	Serie nula o espacios en blanco

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQDLH_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQDLH MyDLH = {MQDLH_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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).
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

MQDMHO-Suprimir opciones de manejador de mensajes

La tabla siguiente resume los campos de la estructura.

Tabla 497. Campos en MQDMHO

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Options	Opciones	Opciones

Visión general de MQDMHO

Disponibilidad: Todos los sistemas WebSphere MQ y clientes WebSphere MQ .

Finalidad: La estructura **MQDMHO** permite a las aplicaciones especificar opciones que controlan cómo se suprimen los manejadores de mensajes. La estructura es un parámetro de entrada en la llamada **MQDLTMH** .

Juego de caracteres y codificación: Los datos de **MQDMHO** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (**MQENC_NATIVE**).

Campos para MQDMHO

La estructura MQDMHO contiene los campos siguientes; los campos se describen en **orden alfabético**:

Opciones (MQLONG)

El valor debe ser:

MQDMHO_NONE

No se ha especificado ninguna opción.

Siempre es un campo de entrada. El valor inicial de este campo es **MQDMHO_NONE**.

StrucId (MQCHAR4)

Este es el identificador de estructura; el valor debe ser:

MQDMHO_STRUC_ID

Identificador de la estructura de opciones de manejador de mensajes de supresión.

Para el lenguaje de programación C, la constante **MQDMHO_STRUC_ID_ARRAY** también está definida; tiene el mismo valor que **MQDMHO_STRUC_ID**, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es **MQDMHO_STRUC_ID**.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQDMHO_VERSION_1
Version-1 suprime la estructura de opciones del manejador de mensajes.

La constante siguiente especifica el número de versión de la versión actual:

MQDMHO_CURRENT_VERSION
Versión actual de la estructura de opciones de manejador de mensajes de supresión.

Siempre es un campo de entrada. El valor inicial de este campo es **MQDMHO_VERSION_1**.

Valores iniciales y declaraciones de idioma para MQDMHO

Tabla 498. Valores iniciales de los campos en MQDMHO

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQDMHO_STRUC_ID	'DMHO'
Version	MQDMHO_VERSION_1	1
Options	MQDMHO_NONE	0

Notas:

1. En el lenguaje de programación C, la variable de macroMQDMHO_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQDMHO MyDMHO = {MQDMHO_DEFAULT};
```

Declaración C

```
typedef struct tagMQDMHO;
struct tagMQDMHO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of MQDLTMH */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

MQDMPO-Suprimir opciones de propiedad de mensaje

La tabla siguiente resume los campos de la estructura. Estructura MQDMPO de -Suprimir opciones de propiedad de mensaje

Tabla 499. Campos en MQDMPO

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Options	Opciones que controlan la acción de MQDMPO	Opciones

Visión general de MQDMPO

Disponibilidad: Todos los sistemas WebSphere MQ y clientes WebSphere MQ .

Finalidad: la estructura MQDMPO permite a las aplicaciones especificar opciones que controlan cómo se suprimen las propiedades de los mensajes. La estructura es un parámetro de entrada en la llamada MQDLTMP.

Conjunto de caracteres y codificación: Los datos de MQDMPO deben estar en el conjunto de caracteres de la aplicación y la codificación de la aplicación (MQENC_NATIVE).

Campos para MQDMPO

Suprimir estructura de opciones de propiedad de mensaje-campos

La estructura MQDMPO contiene los campos siguientes; los campos se describen en **orden alfabético**:

Opciones (MQLONG)
Suprimir estructura de opciones de propiedad de mensaje-Campo Opciones

Opciones de ubicación: Las opciones siguientes están relacionadas con la ubicación relativa de la propiedad en comparación con el cursor de la propiedad.

MQDMPO_DEL_FIRST
Suprime la primera propiedad que coincide con el nombre especificado.

MQDMPO_DEL_PROP_UNDER_CURSOR
Suprime la propiedad a la que apunta el cursor de propiedad; es decir, la propiedad que se ha consultado por última vez utilizando la opción MQIMPO_INQ_FIRST o la opción MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje. También se restablece cuando se especifica el descriptor de mensaje en el campo *MsgHandle* de la estructura MQGMO en una llamada MQGET, o estructura MQPMO en una llamada MQPUT.

Si esta opción se utiliza cuando todavía no se ha establecido el cursor de propiedad, la llamada falla con el código de terminación MQCC_FAILED y la razón MQRC_PROPERTY_NOT_AVAILABLE. Si la propiedad a la que apunta el cursor de propiedad ya se ha suprimido, la llamada también falla con el código de terminación MQCC_FAILED y la razón MQRC_PROPERTY_NOT_AVAILABLE.

Si no se requiere ninguna de las opciones, se puede utilizar la siguiente opción:

MQDMPO_NONE
No se ha especificado ninguna opción.

Este campo siempre es un campo de entrada. El valor inicial de este campo es MQDMPO_DEL_FIRST.

StrucId (MQCHAR4)
Suprimir estructura de opciones de propiedad de mensaje-Campo StrucId

Es el identificador de estructura. El valor debe ser:

MQDMPO_STRUC_ID
Identificador para la estructura de opciones de propiedad de mensaje de supresión.

Para el lenguaje de programación C, también se define la constante MQDMPO_STRUC_ID_ARRAY; tiene el mismo valor que MQDMPO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQDMPO_STRUC_ID.

Versión (MQLONG)
Suprimir estructura de opciones de propiedad de mensaje-Campo Versión

Es el número de versión de la estructura. El valor debe ser:

MQDMPO_VERSION_1
Número de versión para la estructura de opciones de suprimir propiedad de mensaje.

La constante siguiente especifica el número de versión de la versión actual:

MQDMPO_CURRENT_VERSION
Versión actual de la estructura de opciones de supresión de propiedades de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es MQDMPO_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQDMPO
Suprimir estructura de opciones de propiedad de mensaje-Valores iniciales

Tabla 500. Valores iniciales de campos en MQDPMO

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQDMPO_STRUC_ID	'DMPO'
Version	MQDMPO_VERSION_1	1
Options	Opciones que controlan la acción de MQDLTMP	MQDMPO_NONE

Tabla 500. Valores iniciales de campos en MQDPMO (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**

Notas:

1. En el lenguaje de programación C, la variable de macroMQDMPO_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQDMPO MyDMPO = {MQDMPO_DEFAULT};
```

Declaración C
Suprimir estructura de opciones de propiedad de mensaje-Declaración de lenguaje C

```
typedef struct tagMQDMPO MQDMPO;
struct tagMQDMPO {
  MQCHAR4  StrucId;        /* Structure identifier */
  MQLONG   Version;        /* Structure version number */
  MQLONG   Options;        /* Options that control the action of
                              MQDLTMP */
};
```

declaración COBOL
Suprimir estructura de opciones de propiedad de mensaje-Declaración de lenguaje COBOL

```
**   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.
```

Declaración PL/I
Suprimir estructura de opciones de propiedad de mensaje-Declaración de lenguaje PL/I

```
Dcl
 1 MQDPMO 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 */
```

Declaración High Level Assembler
Suprimir estructura de opciones de propiedad de mensaje-Declaración de lenguaje ensamblador

```
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)
```

MQEPH - Cabecera PCF incrustada

La tabla siguiente resume los campos de la estructura.

Tabla 501. Campos en MQEPH		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
StrucLength	Longitud de la estructura MQEPH más MQCFH y las estructuras de parámetros que la siguen	StrucLength
Encoding	Codificación numérica de los datos que siguen a la última estructura de parámetros PCF	Codificación
CodedCharSetId	Identificador de juego de caracteres de los datos que siguen a la última estructura de parámetros PCF	CodedCharSetId
Format	Nombre de formato de los datos que siguen a la última estructura de parámetros PCF	Formato
Flags	Distintivos	Distintivos
PCFHeader	Cabecera de formato de mandato programable (PCF)	Cabecera PCF

Visión general de MQEPH

Disponibilidad: todas las plataformas WebSphere MQ .

Finalidad: la estructura MQEPH describe los datos adicionales que están presentes en un mensaje cuando ese mensaje es un mensaje de formato de mandato programable (PCF). El campo *PCFHeader* define los parámetros PCF que siguen esta estructura y esto le permite seguir los datos del mensaje PCF con otras cabeceras.

Nombre de formato: MQFMT_EMBEDDED_PCF

Conjunto de caracteres y codificación: Los datos de MQEPH deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE.

Establezca el juego de caracteres y la codificación de MQEPH en los campos *CodedCharSetId* y *Encoding* en:

- El MQMD (si la estructura MQEPH está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQEPH (todos los demás casos).

Uso: no puede utilizar estructuras MQEPH para enviar mandatos al servidor de mandatos o a cualquier otro servidor que acepte PCF de gestor de colas.

De forma similar, el servidor de mandatos o cualquier otro servidor de aceptación de PCF de gestor de colas no generan respuestas o sucesos que contengan estructuras MQEPH.

Campos para MQEPH

La estructura MQEPH contiene los campos siguientes; los campos se describen en **orden alfabético**:

CodedCharSetId (MQLONG)

Es el identificador de juego de caracteres de los datos que siguen a la estructura MQEPH y a los parámetros PCF asociados; no se aplica a los datos de tipo carácter de la propia estructura MQEPH.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Encoding (MQLONG)

Es la codificación numérica de los datos que siguen a la estructura MQEPH y a los parámetros PCF asociados; no se aplica a los datos de tipo carácter de la propia estructura MQEPH.

El valor inicial de este campo es 0.

Flags (MQLONG)

Están disponibles los valores siguientes:

MQEPH_NONE
No se han especificado distintivos. MQEPH_NONE está definido para ayudar a la documentación del programa. No se pretende que esta constante se utilice con ninguna otra, pero como su valor es cero, tal uso no se puede detectar.

MQEPH_CCSID_EMBEDDED
El juego de caracteres de los parámetros que contienen datos de caracteres se especifica individualmente en el campo CodedCharSetId de cada estructura. El juego de caracteres de los campos StrucId y Format se define mediante el campo CodedCharSetId en la estructura de cabecera que precede a la estructura MQEPH, o mediante el campo CodedCharSetId en MQMD si MQEPH está al principio del mensaje.

El valor inicial de este campo es MQEPH_NONE.

Format (MQCHAR8)

Es el nombre de formato de los datos que siguen a la estructura MQEPH y a los parámetros PCF asociados.

El valor inicial de este campo es MQFMT_NONE.

PCFHeader (MQCFH)

Es la cabecera de formato de mandato programable (PCF), que define los parámetros PCF que siguen a la estructura MQEPH. Esto le permite seguir los datos del mensaje PCF con otras cabeceras.

La cabecera PCF se define inicialmente con los valores siguientes:

Tabla 502. Valores iniciales de los campos en MQCFH

Nombre de campo	Nombre de constante	Valor de constante
Type	MQCFT_NONE	0
StrucLength	MQCFH_STRUC_LENGH	36
Version	MQCFH_VERSION_3	3
StrucLength	Ninguna	0
Command	MQCMD_NONE	0
MsgSeqNumber	Ninguna	1
Control	MQCFC_LAST	1
CompCode	MQCC_OK	0
Reason	MQRC_NONE	0
ParameterCount	Ninguna	0

La aplicación debe cambiar Type de MQCFT_NONE a un tipo de estructura válido para el uso que está haciendo de la cabecera PCF incorporada.

StrucId (MQCHAR4)

El valor debe ser:

MQEPH_STRUC_ID
Identificador de la estructura de cabecera de distribución.

Para el lenguaje de programación C, también se define la constante MQEPH_STRUC_ID_ARRAY; tiene el mismo valor que MQDH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQEPH_STRUC_ID.

StrucLength (MQLONG)

Es la cantidad de datos que preceden a la siguiente estructura de cabecera. Esto incluye:

- Longitud de la cabecera MQEPH
- La longitud de todos los parámetros PCF que siguen a la cabecera
- Cualquier relleno en blanco que siga a estos parámetros

StrucLength debe ser un múltiplo de 4.

La parte de longitud fija de la estructura está definida por MQEPH_STRUC_LENGTH_FIXED.

El valor inicial de este campo es 68.

Versión (MQLONG)

El valor debe ser:

MQEPH_VERSION_1
Número de versión para la estructura de cabecera PCF incorporada.

La constante siguiente especifica el número de versión de la versión actual:

MQCFH_VERSION_3
Versión actual de la estructura de cabecera PCF incorporada.

El valor inicial de este campo es MQEPH_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQEPH

Tabla 503. Valores iniciales de campos en MQEPH para MQEPH

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQEPH_STRUC_ID	'EPH¬'
Version	MQEPH_VERSION_1	1
StrucLength	MQEPH_STRUC_LENGTH_FIXED	68
Encoding	Ninguna	0
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Espacios en blanco
Flags	MQEPH_NONE	0
PCFHeader	Nombres y valores tal como se definen en Tabla 502 en la página 340	0

Tabla 503. Valores iniciales de campos en MQEPH para MQEPH (continuación)

Nombre de campo	Nombre de constante	Valor de constante

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQEPH_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQEPH MyEPH = {MQEPH_DEFAULT};
```

Declaración C

```
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 */
 };
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

MQGMO-Opciones de obtención de mensajes

La tabla siguiente resume los campos de la estructura.

Tabla 504. Campos en MQGMO		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Options	Opciones que controlan la acción de MQGET	MQGMO-Campo Opciones
WaitInterval	Intervalo de espera	WaitInterval
Signal1	Señal	Signal1
Signal2	Identificador de señal	Signal2
ResolvedQName	Nombre resuelto de la cola de destino	ResolvedQName
Nota: Los campos restantes se ignoran si *Version* es menor que MQGMO_VERSION_2.		
MatchOptions	Opciones que controlan los criterios de selección utilizados para MQGET	MatchOptions
GroupStatus	Distintivo que indica si el mensaje recuperado está en un grupo	GroupStatus
SegmentStatus	Distintivo que indica si el mensaje recuperado es un segmento de un mensaje lógico	SegmentStatus
Segmentation	Distintivo que indica si se permite más segmentación para el mensaje recuperado	Segmentación
Reserved1	Reserved	Reserved1
Nota: Los campos restantes se ignoran si *Version* es menor que MQGMO_VERSION_3.		
MsgToken	Señal de mensaje	MsgToken
ReturnedLength	Longitud de los datos de mensaje devueltos (bytes)	ReturnedLength
Nota: Los campos restantes se ignoran si *Version* es menor que MQGMO_VERSION_4.		
Reserved2	Reserved	Reserved2
MsgHandle	El descriptor de un mensaje que se debe llenar con las propiedades del mensaje que se va a recuperar de la cola.	MsgHandle

Visión general de MQGMO

Disponibilidad: todas las plataformas WebSphere MQ .

Finalidad: La estructura MQGMO permite a la aplicación controlar cómo se eliminan los mensajes de las colas. La estructura es un parámetro de entrada/salida en la llamada MQGET.

Versión: La versión actual de MQGMO es MQGMO_VERSION_4. Determinados campos sólo están disponibles en determinadas versiones de MQGMO. Si necesita portar aplicaciones entre varios entornos, debe asegurarse de que la versión de MQGMO sea coherente en todos los entornos. Los campos que existen sólo en determinadas versiones de la estructura se identifican como tales en "MQGMO-Opciones de obtención de mensajes" en la página 344 y en las descripciones de campo.

Los archivos de cabecera, COPY e INCLUDE proporcionados para los lenguajes de programación soportados contienen la versión más reciente de MQGMO soportada por el entorno, pero con el valor inicial del campo *Version* establecido en MQGMO_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1 , establezca el campo *Version* en el número de versión de la versión necesaria.

Conjunto de caracteres y codificación: Los datos de MQGMO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos para MQGMO

La estructura MQGMO contiene los campos siguientes; los campos se describen en **orden alfabético**:

GroupStatus (MQCHAR)
Este distintivo indica si el mensaje recuperado está en un grupo.

Tiene uno de los siguientes valores:

MQGS_NOT_IN_GROUP
El mensaje no está en un grupo.

MQGS_MSG_IN_GROUP
El mensaje está en un grupo, pero no es el último del grupo.

MQGS_LAST_MSG_IN_GROUP
El mensaje es el último del grupo.

También es el valor devuelto si el grupo consta de un solo mensaje.

Se trata de un campo de salida. El valor inicial de este campo es MQGS_NOT_IN_GROUP. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

MatchOptions (MQLONG)

Estas opciones permiten a la aplicación elegir qué campos del parámetro *MsgDesc* utilizar para seleccionar el mensaje devuelto por la llamada MQGET. La aplicación establece las opciones necesarias en este campo y, a continuación, establece los campos correspondientes en el parámetro *MsgDesc* en los valores necesarios para esos campos. Sólo los mensajes que tienen estos valores en MQMD para el mensaje son candidatos para la recuperación utilizando ese parámetro *MsgDesc* en la llamada MQGET. Los campos para los que la opción de coincidencia correspondiente *no* se especifica se ignoran al seleccionar el mensaje que se devolverá. Si no especifica ningún criterio de selección en la llamada MQGET (es decir, *cualquier* mensaje es aceptable), establezca *MatchOptions* en MQMO_NONE.

- En z/OS, los criterios de selección que se pueden utilizar pueden estar restringidos por el tipo de índice utilizado para la cola. Consulte el atributo de cola *IndexType* para obtener más detalles.

Si especifica MQGMO_LOGICAL_ORDER, sólo determinados mensajes pueden ser devueltos por la siguiente llamada MQGET:

- Si no hay ningún grupo actual o mensaje lógico, sólo se podrán devolver los mensajes que tengan *MsgSeqNumber* igual a 1 y *Offset* igual a 0 . En esta situación, puede utilizar una o más de las siguientes opciones de coincidencia para seleccionar cuál de los mensajes elegibles se devuelve:
  - MQMO_MATCH_MSG_ID

  - MQMO_MATCH_CORREL_ID
  - MQMO_MATCH_GROUP_ID
- Si *hay* un grupo actual o un mensaje lógico, sólo el siguiente mensaje del grupo o segmento siguiente del mensaje lógico es elegible para su devolución, y esto no se puede modificar especificando las opciones MQMO_ *.

En los dos casos anteriores, puede especificar opciones de coincidencia que no se aplican, pero el valor del campo relevante en el parámetro *MsgDesc* debe coincidir con el valor del campo correspondiente en el mensaje que se va a devolver; la llamada falla con el código de razón MQRC_MATCH_OPTIONS_ERROR es que esta condición no se cumple.

MatchOptions se ignora si especifica MQGMO_MSG_UNDER_CURSOR o MQGMO_BROWSE_MSG_UNDER_CURSOR.

La obtención de mensajes basados en la propiedad de mensaje no se realiza utilizando opciones de coincidencia; para obtener más información, consulte "SelectionString (MQCHARV)" en la página 463 .

Puede especificar una o más de las siguientes opciones de coincidencia:

MQMO_MATCH_MSG_ID
El mensaje que se va a recuperar debe tener un identificador de mensaje que coincida con el valor del campo *MsgId* en el parámetro *MsgDesc* de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de correlación).

Si omite esta opción, el campo *MsgId* en el parámetro *MsgDesc* se ignora y cualquier identificador de mensaje coincidirá.

Nota: El identificador de mensaje MQMI_NONE es un valor especial que coincide con *cualquier* identificador de mensaje en el MQMD del mensaje. Por lo tanto, especificar MQMO_MATCH_MSG_ID con MQMI_NONE es lo mismo que *no* especificar MQMO_MATCH_MSG_ID.

MQMO_MATCH_CORREL_ID
El mensaje que se va a recuperar debe tener un identificador de correlación que coincida con el valor del campo *CorrelId* en el parámetro *MsgDesc* de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de mensaje).

Si omite esta opción, el campo *CorrelId* del parámetro *MsgDesc* se ignora y cualquier identificador de correlación coincidirá.

Nota: El identificador de correlación MQCI_NONE es un valor especial que coincide con *cualquier* identificador de correlación en el MQMD del mensaje. Por lo tanto, especificar MQMO_MATCH_CORREL_ID con MQCI_NONE es lo mismo que *no* especificar MQMO_MATCH_CORREL_ID.

MQMO_MATCH_GROUP_ID
El mensaje que se va a recuperar debe tener un identificador de grupo que coincida con el valor del campo *GroupId* en el parámetro *MsgDesc* de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de correlación).

Si omite esta opción, el campo *GroupId* del parámetro *MsgDesc* se ignora y cualquier identificador de grupo coincidirá.

Nota: El identificador de grupo MQGI_NONE es un valor especial que coincide con *cualquier* identificador de grupo en el MQMD del mensaje. Por lo tanto, especificar MQMO_MATCH_GROUP_ID con MQGI_NONE es lo mismo que *no* especificar MQMO_MATCH_GROUP_ID.

MQMO_MATCH_MSG_SEQ_NUMBER
El mensaje que se va a recuperar debe tener un número de secuencia de mensaje que coincida con el valor del campo *MsgSeqNumber* en el parámetro *MsgDesc* de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que pueda aplicarse (por ejemplo, el identificador de grupo).

Si omite esta opción, el campo *MsgSeqNumber* del parámetro *MsgDesc* se ignora y cualquier número de secuencia de mensaje coincidirá.

MQMO_MATCH_OFFSET
El mensaje que se va a recuperar debe tener un desplazamiento que coincida con el valor del campo *Offset* en el parámetro *MsgDesc* de la llamada MQGET. Esta coincidencia se suma a cualquier otra coincidencia que se pueda aplicar (por ejemplo, el número de secuencia de mensaje).

Si omite esta opción no se especifica, el campo *Offset* del parámetro *MsgDesc* se ignora y cualquier desplazamiento coincidirá.

- Esta opción no está soportada en z/OS.

MQMO_MATCH_MSG_TOKEN
El mensaje que se va a recuperar debe tener una señal de mensaje que coincida con el valor del campo *MsgToken* en la estructura MQGMO especificada en la llamada MQGET.

Puede especificar esta opción para todas las colas locales. Si lo especifica para una cola que tiene un *IndexType* de MQIT_MSG_TOKEN (una cola gestionada por WLM), no puede especificar ninguna otra opción de coincidencia con MQMO_MATCH_MSG_TOKEN.

No puede especificar MQMO_MATCH_MSG_TOKEN con MQGMO_WAIT o MQGMO_SET_SIGNAL. Si la aplicación desea esperar a que llegue un mensaje a una cola que tenga un *IndexType* de MQIT_MSG_TOKEN, especifique MQMO_NONE.

Si omite esta opción, el campo *MsgToken* en MQGMO se ignora y cualquier señal de mensaje coincidirá.

Si no especifica ninguna de las opciones descritas, puede utilizar la opción siguiente:

MQMO_NONE
No utilice coincidencias al seleccionar el mensaje que se va a devolver; todos los mensajes de la cola son elegibles para la recuperación (pero están sujetos al control de las opciones MQGMO_ALL_MSGS_AVAILABLE, MQGMO_ALL_SEGMENTS_AVAILABLE y MQGMO_COMPLETE_MSG).

MQMO_NONE ayuda a la documentación del programa. No está previsto que esta opción se utilice con cualquier otra opción MQMO_ *, pero como su valor es cero, no se puede detectar dicho uso.

Este es un campo de entrada. El valor inicial de este campo es MQMO_MATCH_MSG_ID con MQMO_MATCH_CORREL_ID. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

Nota: El valor inicial del campo *MatchOptions* se define para la compatibilidad con gestores de colas MQSeries anteriores. Sin embargo, al leer una serie de mensajes de una cola sin utilizar criterios de selección, este valor inicial requiere que la aplicación restablezca los campos *MsgId* y *CorrelId* en MQMI_NONE y MQCI_NONE antes de cada llamada MQGET. Evite la necesidad de restablecer *MsgId* y *CorrelId* estableciendo *Version* en MQGMO_VERSION_2y *MatchOptions* en MQMO_NONE.

MsgHandle (MQHMSG)

Si se especifica la opción MQGMO_PROPERTIES_AS_Q_DEF y el atributo de cola `PropertyControl` no está establecido en MQPROP_FORCE_MQRFH2 , este es el descriptor de contexto de un mensaje que se rellenará con las propiedades del mensaje que se está recuperando de la cola. El descriptor de contexto se crea mediante una llamada MQCRTMH. Las propiedades que ya estén asociadas con el descriptor de contexto se borrarán antes de recuperar un mensaje.

También se puede especificar el valor siguiente:
MQHM_NONE
No se ha proporcionado ningún manejador de mensajes.

No es necesario ningún descriptor de mensaje en la llamada MQGET si se proporciona un descriptor de mensaje válido y se utiliza en la salida para contener las propiedades del mensaje, el descriptor de mensaje asociado con el descriptor de mensaje se utiliza para los campos de entrada.

Si se especifica un descriptor de mensaje en la llamada MQGET, siempre tiene prioridad sobre el descriptor de mensaje asociado a un descriptor de mensaje.

Si se especifica MQGMO_PROPERTIES_FORCE_MQRFH2 , o se especifica MQGMO_PROPERTIES_AS_Q_DEF y el atributo de cola `PropertyControl` es MQPROP_FORCE_MQRFH2 , la llamada falla con el código de razón MQRC_MD_ERROR cuando no se especifica ningún parámetro de descriptor de mensaje.

Al volver de la llamada MQGET, las propiedades y el descriptor de mensaje asociados con este descriptor de mensaje se actualizan para reflejar el estado del mensaje recuperado (así como el descriptor de mensaje si se ha proporcionado uno en la llamada MQGET). A continuación, se pueden consultar las propiedades del mensaje utilizando la llamada MQINQMP.

Excepto para las extensiones de descriptor de mensaje, cuando están presentes, una propiedad que se puede consultar con la llamada MQINQMP no está contenida en los datos del mensaje; si el mensaje de la cola contenía propiedades en los datos del mensaje, estos se eliminan de los datos del mensaje antes de que se devuelvan los datos a la aplicación.

Si no se proporciona ningún descriptor de mensaje o la versión es inferior a MQGMO_VERSION_4 , debe proporcionar un descriptor de mensaje válido en la llamada MQGET. Las propiedades de mensaje (excepto las contenidas en el descriptor de mensaje) se devuelven en el asunto de datos de mensaje al valor de las opciones de propiedad en la estructura MQGMO y el atributo de cola `PropertyControl` .

Se trata de un campo de entrada siempre. El valor inicial de este campo es MQHM_NONE. Este campo se ignora si `Version` es menor que MQGMO_VERSION_4.

MsgToken (MQBYTE16)
Campo MsgToken -Estructura MQGMO. El gestor de colas utiliza este campo para identificar de forma exclusiva un mensaje.

Es una serie de bytes generada por el gestor de colas para identificar un mensaje de forma exclusiva en una cola. La señal de mensaje se genera cuando el mensaje se coloca por primera vez en el gestor de colas y permanece con el mensaje hasta que el mensaje se elimina permanentemente del gestor de colas, a menos que se reinicie el gestor de colas.

Cuando se elimina el mensaje de la cola, el *MsgToken* que ha identificado esa instancia del mensaje ya no es válido y nunca se reutiliza. Si se reinicia el gestor de colas, es posible que el *MsgToken* que ha identificado un mensaje en la cola antes del reinicio no sea válido después del reinicio. Sin embargo, el *MsgToken* nunca se reutiliza para identificar una instancia de mensaje diferente. El *MsgToken* lo genera el gestor de colas y no es visible para ninguna aplicación externa.

Cuando una llamada a MQGET devuelve un mensaje donde se proporciona un MQGMO de la versión 3 o superior, el gestor de colas devuelve el *MsgToken* que identifica el mensaje en la cola en el MQGMO. Hay una excepción a esto: cuando el mensaje se elimina de la cola fuera del punto de sincronización, es posible que el gestor de colas no devuelva un *MsgToken* porque no es útil identificar el mensaje devuelto en una llamada MQGET posterior. Las aplicaciones sólo deben utilizar *MsgToken* para hacer referencia al mensaje en las llamadas MQGET posteriores.

Si se proporciona un *MsgToken* y se especifica *MatchOption* MQMO_MATCH_MSG_TOKEN y no se especifica MQGMO_MSG_UNDER_CURSOR ni MQGMO_BROWSE_MSG_UNDER_CURSOR, sólo se puede devolver el mensaje identificado por ese *MsgToken* . La opción es válida en todas las colas locales independientemente de INDXTYPE, y en z/OS debe utilizar INDXTYPE (MSGTOKEN) sólo en las colas del Gestor de carga de trabajo (WLM).

Los demás *MatchOptions* especificados se comprueban y, si no coinciden, se devuelve MQRC_NO_MSG_AVAILABLE. Si MQGMO_BROWSE_NEXT está codificado con MQMO_MATCH_MSG_TOKEN, el mensaje identificado por *MsgToken* sólo se devuelve si está más allá del cursor para examinar para el descriptor de contexto de llamada.

Si se especifica MQGMO_MSG_UNDER_CURSOR o MQGMO_BROWSE_MSG_UNDER_CURSOR, se ignora MQMO_MATCH_MSG_TOKEN.

MQMO_MATCH_MSG_TOKEN no es válido con las siguientes opciones de obtención de mensaje:

- MQGMO_WAIT
- MQGMO_SET_SIGNAL

Para una llamada MQGET que especifique MQMO_MATCH_MSG_TOKEN, se debe proporcionar un MQGMO de la versión 3 o posterior a la llamada, de lo contrario se devuelve MQRC_INJU_GMO_VERSION.

Si *MsgToken* no es válido en este momento, se devuelve MQCC_FAILED con MQRC_NO_MSG_AVAILABLE, a menos que haya otro error.

Opciones (MQLONG)

Las opciones de **MQGMO** controlan la acción de MQGET. Puede especificar cero o más de las opciones. Si necesita más de un valor opcional:

- Añada los valores (no añada la misma constante más de una vez), o
- Combine los valores utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit).

Las combinaciones de opciones que no son válidas se anotan; todas las demás combinaciones son válidas.

Opciones de espera: Las opciones siguientes están relacionadas con la espera de que lleguen mensajes a la cola:

MQGMO_WAIT
La aplicación espera hasta que llega un mensaje adecuado. El tiempo máximo que la aplicación espera se especifica en *WaitInterval*.

Importante: No hay espera, ni retardo, si un mensaje adecuado está disponible inmediatamente.

Si se inhiben las solicitudes MQGET o las solicitudes MQGET se inhiben mientras se espera, se cancela la espera. La llamada se completa con MQCC_FAILED y el código de razón MQRC_GET_INHIBITED, independientemente de si hay mensajes adecuados en la cola.

Puede utilizar MQGMO_WAIT con las opciones MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT .

Si varias aplicaciones están esperando en la misma cola compartida, las reglas siguientes seleccionan qué aplicación se activa cuando llega un mensaje adecuado:

Tabla 505. Reglas para activar llamadas de MQGET en una cola compartida.

Número de llamadas de MQGET en espera de ser activadas		**Resultado**
Con una opción BROWSE	**Sin una opción BROWSE**[1]	
Ninguna	uno o más	Se activa una llamada MQGET sin una opción BROWSE .
uno o más	Ninguna	Todas las llamadas de MQGET con una opción BROWSE están activadas.
uno o más	uno o más	Se activa una llamada MQGET sin una opción BROWSE . El número de llamadas MQGET con una opción BROWSE que están activadas es imprevisible.

Si más de una llamada MQGET sin una opción BROWSE está esperando en la misma cola, sólo se activa una. El gestor de colas intenta dar prioridad a las llamadas en espera en el orden siguiente:

1. Solicitudes get-wait específicas que sólo pueden satisfacerse mediante determinados mensajes, por ejemplo, aquellos con un *MsgId* o *CorrelId* específico (o ambos).
2. Solicitudes de obtención de espera generales que pueden ser satisfechas por cualquier mensaje.

Nota:

- Dentro de la primera categoría, no se otorga ninguna prioridad adicional a las solicitudes get-wait más específicas. Por ejemplo, las solicitudes que especifican *MsgId* y *CorrelId*.

[1] Una llamada MQGET que especifica la opción MQGMO_LOCK se trata como una llamada de no examen.

- Dentro de cualquiera de las dos categorías, no se puede predecir qué aplicación está seleccionada. En concreto, la aplicación que más espera no es necesariamente la seleccionada.
- La longitud de vía de acceso y las consideraciones de planificación de prioridad del sistema operativo pueden significar que una aplicación en espera de una prioridad de sistema operativo inferior a la esperada recupera el mensaje.
- También puede suceder que una aplicación que no está en espera recupere el mensaje en lugar de uno que sí lo está.

En z/OS, se aplican los puntos siguientes:

- Si desea que la aplicación continúe con otro trabajo mientras espera a que llegue el mensaje, considere la posibilidad de utilizar la opción de señal (MQGMO_SET_SIGNAL) en su lugar. Sin embargo, la opción de señal es específica del entorno; las aplicaciones que se van a portar entre distintos entornos no deben utilizarla.
- Si hay más de una llamada MQGET en espera del mismo mensaje, con una combinación de opciones de espera y de señal, cada llamada en espera se considera igualmente. Es un error especificar MQGMO_SET_SIGNAL con MQGMO_WAIT. También es un error especificar esta opción con un descriptor de contexto de cola para el que hay una señal pendiente.
- Si especifica MQGMO_WAIT o MQGMO_SET_SIGNAL para una cola que tiene un *IndexType* de MQIT_MSG_TOKEN, no se permiten criterios de selección. Esto significa que:
  - Si está utilizando una version-1 MQGMO, establezca los campos *MsgId* y *CorrelId* en el MQMD especificado en la llamada MQGET en MQMI_NONE y MQCI_NONE.
  - Si está utilizando una version-2 o posterior MQGMO, establezca el campo *MatchOptions* en MQMO_NONE.
- Para una llamada MQGET en una cola compartida y la llamada es una solicitud de examen, o una obtención destructiva de un mensaje de grupo, y ni *MsgId* ni *CorrelId* deben coincidir, la llamada MQGET se vuelve a emitir cada 200 milisegundos hasta que llega un mensaje adecuado a la cola o caduca el intervalo de espera.

  Este método provoca una sobrecarga de proceso inesperada y no es un método eficaz de recuperación de mensajes cuando los mensajes se añaden con poca frecuencia. Para evitar esta sobrecarga para el caso de examen, especifique *MsgId* (si no está indexado o indexado por *MsgId*) o *CorrelId* (si está indexado por *CorrelId*) coincidente en la llamada MQGET .

MQGMO_WAIT se ignora si se especifica con MQGMO_BROWSE_MSG_UNDER_CURSOR o MQGMO_MSG_UNDER_CURSOR; no se genera ningún error.

MQGMO_NO_WAIT
La aplicación no espera si no hay ningún mensaje adecuado disponible. MQGMO_NO_WAIT es lo opuesto a MQGMO_WAIT. MQGMO_NO_WAIT está definido para ayudar a la documentación del programa. Es el valor predeterminado si no se especifica ninguno.

MQGMO_SET_SIGNAL
Utilice esta opción con los campos *Signal1* y *Signal2* . Permite a las aplicaciones continuar con otro trabajo mientras esperan a que llegue un mensaje. También permite a las aplicaciones (si hay disponibles recursos adecuados del sistema operativo) esperar a que lleguen mensajes a más de una cola.

Nota: La opción MQGMO_SET_SIGNAL es específica del entorno; no la utilice para las aplicaciones que desea portar.

En dos circunstancias, la llamada se completa de la misma forma que si no se hubiera especificado esta opción:

1. Si un mensaje disponible actualmente cumple los criterios especificados en el descriptor de mensaje.
2. Si se detecta un error de parámetro u otro error síncrono.

Si no hay ningún mensaje que cumpla los criterios especificados en el descriptor de mensaje disponible actualmente, el control vuelve a la aplicación sin esperar a que llegue

un mensaje. Los parámetros *CompCode* y *Reason* se establecen en MQCC_WARNING y MQRC_SIGNAL_REQUEST_ACCEPTED. No se han establecido otros campos de salida en el descriptor de mensaje y los parámetros de salida de la llamada MQGET . Cuando un mensaje adecuado llega más tarde, la señal se emite mediante la publicación del BCE.

A continuación, el llamante debe volver a emitir la llamada MQGET para recuperar el mensaje. La aplicación puede esperar esta señal, utilizando funciones proporcionadas por el sistema operativo.

Si el sistema operativo proporciona un mecanismo de espera múltiple, puede utilizarlo para esperar a que llegue un mensaje en cualquiera de varias colas.

Si se especifica un *WaitInterval* distinto de cero, la señal se entrega después de que caduque el intervalo de espera. El gestor de colas también puede cancelar la espera, en cuyo caso se entrega la señal.

Más de una llamada MQGET puede establecer una señal para el mismo mensaje. El orden en el que se activan las aplicaciones es el mismo que el descrito para MQGMO_WAIT.

Si más de una llamada MQGET está esperando el mismo mensaje, cada llamada en espera se considera igualmente. Las llamadas pueden incluir una mezcla de opciones de espera y de señal.

Bajo determinadas condiciones, la llamada MQGET puede recuperar un mensaje y se puede entregar una señal resultante de la llegada del mismo mensaje. Cuando se entrega una señal, una aplicación debe estar preparada para que no haya ningún mensaje disponible.

Un descriptor de contexto de cola no puede tener más de una solicitud de señal pendiente.

Esta opción no es válida con ninguna de las opciones siguientes:

- MQGMO_UNLOCK
- MQGMO_WAIT

Para una llamada MQGET en una cola compartida y la llamada es una solicitud de examen, o una obtención destructiva de un mensaje de grupo, y ni *MsgId* ni *CorrelId* deben coincidir, la señal del usuario ECB se publica MQEC_MSG_ARRIVED después de 200 milisegundos.

Esto ocurre, aunque es posible que no haya llegado un mensaje adecuado a la cola, hasta que haya caducado el intervalo de espera, cuando la cola se envía con MQEC_WAIT_INTERVAL_EXPIRED. Cuando se publica MQEC_MSG_ARRIVED , debe volver a emitir una segunda llamada MQGET para recuperar el mensaje, si hay una disponible.

Esta técnica se utiliza para asegurarse de que se le informa a tiempo de la llegada de un mensaje, pero puede aparecer como una sobrecarga de proceso inesperada cuando se compara con una secuencia de llamada similar en una cola no compartida.

No es un método eficaz de recuperación de mensajes cuando los mensajes se añaden con poca frecuencia. Para evitar esta sobrecarga para el caso de examen, especifique *MsgId* (si no está indexado o indexado por *MsgId*) o *CorrelId* (si está indexado por *CorrelId*) coincidente en la llamada MQGET .

Esta opción solo está soportada en z/OS .

MQGMO_FAIL_IF_QUIESCING
Fuerza a que la llamada MQGET falle si el gestor de colas está en estado de desactivación temporal.

En z/OS, esta opción también fuerza que la llamada MQGET falle si la conexión (para una aplicación CICS o IMS) está en estado de desactivación temporal.

Si esta opción se especifica con MQGMO_WAIT o MQGMO_SET_SIGNAL, y la espera o señal está pendiente en el momento en que el gestor de colas entra en el estado de desactivación temporal:

- La espera se cancela y la llamada devuelve el código de terminación MQCC_FAILED con el código de razón MQRC_Q_MGR_QUIESCING o MQRC_CONNECTION_QUIESCING.
- La señal se cancela con un código de terminación de señal específico del entorno.

En z/OS, la señal se completa con el código de finalización de suceso MQEC_Q_MGR_QUIESCING o MQEC_CONNECTION_QUIESCING.

Si no se especifica MQGMO_FAIL_IF_QUIESCING y el gestor de colas o la conexión entra en el estado de desactivación temporal, la espera o la señal no se cancela.

Opciones de punto de sincronización: las opciones siguientes están relacionadas con la participación de la llamada MQGET dentro de una unidad de trabajo:

MQGMO_SYNCPOINT
La solicitud es operar dentro de los protocolos normales de unidad de trabajo. El mensaje se marca como no disponible para otras aplicaciones, pero sólo se suprime de la cola cuando se confirma la unidad de trabajo. El mensaje vuelve a estar disponible si se restituye la unidad de trabajo.

Puede dejar MQGMO_SYNCPOINT y MQGMO_NO_SYNCPOINT sin establecer. En este caso, la inclusión de la solicitud de obtención en los protocolos de unidad de trabajo viene determinada por el entorno que ejecuta el gestor de colas. No lo determina el entorno que ejecuta la aplicación. En z/OS, la solicitud de obtención está dentro de una unidad de trabajo. En todos los demás entornos, la solicitud de obtención no está dentro de una unidad de trabajo.

Debido a estas diferencias, una aplicación que desea portar no debe permitir que esta opción sea la predeterminada; especifique MQGMO_SYNCPOINT o MQGMO_NO_SYNCPOINT explícitamente.

Esta opción no es válida con ninguna de las opciones siguientes:

- 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 solicitud es operar dentro de los protocolos de unidad de trabajo normales, pero *sólo* si el mensaje recuperado es persistente. Un mensaje persistente tiene el valor MQPER_PERSISTENT en el campo *Persistence* en MQMD.

- Si el mensaje es persistente, el gestor de colas procesa la llamada como si la aplicación hubiera especificado MQGMO_SYNCPOINT.
- Si el mensaje no es persistente, el gestor de colas procesa la llamada como si la aplicación hubiera especificado MQGMO_NO_SYNCPOINT.

Esta opción no es válida con ninguna de las opciones siguientes:

- 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

Esta opción está soportada en los entornos siguientes: AIX, HP-UX, z/OS, IBM i, Solaris y Linux, además de clientes MQI de WebSphere MQ conectados a estos sistemas.

MQGMO_NO_SYNCPOINT
La solicitud es operar fuera de los protocolos normales de unidad de trabajo. Si obtiene un mensaje sin una opción de examen, se suprime de la cola inmediatamente. El mensaje no puede volver a estar disponible restituir la unidad de trabajo.

Esta opción se presupone si especifica MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT.

Puede dejar MQGMO_SYNCPOINT y MQGMO_NO_SYNCPOINT sin establecer. En este caso, la inclusión de la solicitud de obtención en los protocolos de unidad de trabajo viene determinada por el entorno que ejecuta el gestor de colas. No lo determina el entorno que ejecuta la aplicación. En z/OS, la solicitud de obtención está dentro de una unidad de trabajo. En todos los demás entornos, la solicitud de obtención no está dentro de una unidad de trabajo.

Debido a estas diferencias, una aplicación que desea portar no debe permitir que esta opción sea la predeterminada; especifique MQGMO_SYNCPOINT o MQGMO_NO_SYNCPOINT explícitamente.

Esta opción no es válida con ninguna de las opciones siguientes:

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT

MQGMO_MARK_SKIP_BACKOUT
Restituir una unidad de trabajo sin restablecer en la cola el mensaje que se ha marcado con esta opción.

Esta opción solo está soportada en z/OS.

Si se especifica esta opción, también se debe especificar MQGMO_SYNCPOINT . MQGMO_MARK_SKIP_BACKOUT no es válido con ninguna de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_LOCK
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Nota: En IMS y CICS, es posible que tenga que emitir una llamada adicional de WebSphere MQ después de restituir una unidad de trabajo que contiene un mensaje marcado con MQGMO_MARK_SKIP_BACKOUT. Debe emitir una llamada WebSphere MQ antes de confirmar la nueva unidad de trabajo que contiene el mensaje marcado. La llamada puede ser cualquier llamada de WebSphere MQ que desee.

1. En IMS, si no ha aplicado IMS APAR PN60855 y está ejecutando una aplicación IMS MPP o BMP.
2. En CICS, si está ejecutando alguna aplicación.

En ambos casos, emita cualquier llamada de WebSphere MQ antes de confirmar la nueva unidad de trabajo que contiene el mensaje restituido.

Nota: Dentro de una unidad de trabajo, sólo puede haber una solicitud de obtención marcada como omisión de restitución, así como ninguna o varias solicitudes de obtención no marcadas.

Si una aplicación se restituye de una unidad de trabajo, un mensaje que se ha recuperado utilizando MQGMO_MARK_SKIP_BACKOUT no se restaura a su estado anterior. Otras actualizaciones de recursos se restituyen. El mensaje se trata como si se hubiera recuperado en una nueva unidad de trabajo iniciada por la solicitud de restitución. El mensaje se recupera sin la opción MQGMO_MARK_SKIP_BACKOUT .

MQGMO_MARK_SKIP_BACKOUT es útil si, después de que se hayan cambiado algunos recursos, se hace evidente que la unidad de trabajo no se puede completar correctamente. Si omite esta opción,

restituir la unidad de trabajo restablece el mensaje en la cola. Se vuelve a producir la misma secuencia de sucesos, cuando el mensaje se recupera a continuación.

Sin embargo, si especifica MQGMO_MARK_SKIP_BACKOUT en la llamada MQGET original, restituir la unidad de trabajo restituye las actualizaciones en los otros recursos. El mensaje se trata como si se hubiera recuperado bajo una nueva unidad de trabajo. La aplicación puede realizar el manejo de errores adecuado. Puede enviar un mensaje de informe al remitente del mensaje original o colocar el mensaje original en la cola de mensajes no entregados. A continuación, puede confirmar la nueva unidad de trabajo. La confirmación de la nueva unidad de trabajo elimina el mensaje de forma permanente de la cola original.

MQGMO_MARK_SKIP_BACKOUT marca un único mensaje físico. Si el mensaje pertenece a un grupo de mensajes, los demás mensajes del grupo no se marcan. De forma similar, si el mensaje marcado es un segmento de un mensaje lógico, los demás segmentos del mensaje lógico no se marcan.

Cualquier mensaje de un grupo se puede marcar, pero si los mensajes se recuperan utilizando MQGMO_LOGICAL_ORDER, es conveniente marcar el primer mensaje del grupo. Si la unidad de trabajo se restituye, el primer mensaje (marcado) se mueve a la nueva unidad de trabajo. El segundo y los mensajes posteriores del grupo se restablecen en la cola. Los mensajes que quedan en la cola no pueden ser recuperados por otra aplicación utilizando MQGMO_LOGICAL_ORDER. El primer mensaje del grupo ya no está en la cola. Sin embargo, la aplicación que ha realizado la copia de seguridad de la unidad de trabajo puede recuperar el segundo y los mensajes posteriores en la nueva unidad de trabajo utilizando la opción MQGMO_LOGICAL_ORDER . El primer mensaje ya se ha recuperado.

Ocasionalmente es posible que tenga que restituir la nueva unidad de trabajo. Por ejemplo, porque la cola de mensajes no entregados está llena y el mensaje no debe descartarse. Al restituir la nueva unidad de trabajo se restablece el mensaje en la cola original, lo que impide que se pierda el mensaje. Sin embargo, en esta situación el proceso no puede continuar. Después de restituir la nueva unidad de trabajo, la aplicación debe informar al operador o administrador de que hay un error irrecuperable y, a continuación, finalizar.

MQGMO_MARK_SKIP_BACKOUT sólo funciona si la unidad de trabajo que contiene la solicitud de obtención se interrumpe cuando la aplicación la restituya. Si la unidad de trabajo que contiene la solicitud de obtención se restituye porque la transacción o el sistema falla, se ignora MQGMO_MARK_SKIP_BACKOUT . Cualquier mensaje recuperado utilizando esta opción se restablece en la cola de la misma forma que los mensajes recuperados sin esta opción.

Opciones de examen: Las opciones siguientes están relacionadas con el examen de mensajes en la cola:

MQGMO_BROWSE_FIRST
Cuando se abre una cola con la opción MQOO_BROWSE, se establece un cursor para examinar, situado lógicamente antes del primer mensaje de la cola. A continuación, puede utilizar las llamadas MQGET especificando la opción MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXTo MQGMO_BROWSE_MSG_UNDER_CURSOR para recuperar mensajes de la cola de forma no destructiva. El cursor para examinar marca la posición, dentro de los mensajes de la cola, desde la que la siguiente llamada MQGET con MQGMO_BROWSE_NEXT busca un mensaje adecuado.

MQGMO_BROWSE_FIRST no es válido con ninguna de las opciones siguientes:

- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

También es un error si la cola no se ha abierto para examinar.

Una llamada MQGET con MQGMO_BROWSE_FIRST ignora la posición anterior del cursor para examinar. Se recupera el primer mensaje de la cola que cumple las condiciones especificadas en el descriptor de mensaje. El mensaje permanece en la cola y el cursor para examinar se coloca en este mensaje.

Después de esta llamada, el cursor para examinar se coloca en el mensaje que se ha devuelto. Es posible que el mensaje se elimine de la cola antes de que se emita la siguiente llamada MQGET con MQGMO_BROWSE_NEXT . En este caso, el cursor para examinar permanece en la posición de la cola que ocupaba el mensaje, aunque esa posición esté ahora vacía.

Utilice la opción MQGMO_MSG_UNDER_CURSOR con una llamada MQGET que no sea examinar, para eliminar el mensaje de la cola.

El cursor para examinar no se mueve mediante una llamada MQGET de no examinar, aunque se utilice el mismo descriptor de contexto *Hobj* . Tampoco se mueve mediante una llamada MQGET de examen que devuelve un código de terminación de MQCC_FAILED, o un código de razón de MQRC_TRUNCATED_MSG_FAILED.

Especifique la opción MQGMO_LOCK con esta opción, para bloquear el mensaje que se examina.

Puede especificar MQGMO_BROWSE_FIRST con cualquier combinación válida de las opciones MQGMO_* y MQMO_* que controlan el proceso de mensajes en grupos y segmentos de mensajes lógicos.

Si especifica MQGMO_LOGICAL_ORDER, los mensajes se examinan en orden lógico. Si omite esta opción, los mensajes se examinan en orden físico. Si especifica MQGMO_BROWSE_FIRST, puede conmutar entre el orden lógico y el orden físico. Las llamadas MQGET subsiguientes que utilizan MQGMO_BROWSE_NEXT examinan la cola en el mismo orden que la llamada más reciente que ha especificado MQGMO_BROWSE_FIRST para el descriptor de contexto de cola.

El gestor de colas retiene dos conjuntos de información de grupo y segmento para llamadas MQGET . La información de grupo y segmento para las llamadas de examen se conserva por separado de la información para las llamadas que eliminan mensajes de la cola. Si especifica MQGMO_BROWSE_FIRST, el gestor de colas ignora la información de grupo y segmento para examinar. Explora la cola como si no hubiera ningún grupo actual y ningún mensaje lógico actual. Si la llamada MQGET es satisfactoria, código de terminación MQCC_OK o MQCC_WARNING, la información de grupo y segmento para examinar se establece en la del mensaje devuelto. Si la llamada falla, la información de grupo y segmento permanece igual que antes de la llamada.

MQGMO_BROWSE_NEXT
Avance el cursor de examen hasta el siguiente mensaje de la cola que cumpla los criterios de selección especificados en la llamada MQGET . El mensaje se devuelve a la aplicación, pero permanece en la cola.

MQGMO_BROWSE_NEXT no es válido con ninguna de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

También es un error si la cola no se ha abierto para examinar.

MQGMO_BROWSE_NEXT se comporta de la misma forma que MQGMO_BROWSE_FIRST, si es la primera llamada para examinar una cola, después de que se haya abierto la cola para examinarla.

Es posible que el mensaje bajo el cursor se elimine de la cola antes de que se emita la siguiente llamada MQGET con MQGMO_BROWSE_NEXT . El cursor para examinar permanece lógicamente en la posición de la cola que ocupaba el mensaje, aunque dicha posición esté ahora vacía.

Los mensajes se almacenan en la cola de una de estas dos maneras:

- FIFO dentro de prioridad (MQMDS_PRIORITY), o
- FIFO *independientemente* de la prioridad (MQMDS_FIFO)

El atributo de cola *MsgDeliverySequence* indica qué método se aplica (consulte "Atributos para colas" en la página 816 para obtener más detalles).

Una cola puede tener un *MsgDeliverySequence* de MQMDS_PRIORITY. Llega un mensaje a la cola que tiene una prioridad más alta que la a la que apunta actualmente el cursor para examinar. En cuyo caso, el mensaje de prioridad más alta no se encuentra durante el barrido actual de la cola utilizando MQGMO_BROWSE_NEXT. Sólo se puede encontrar después de que el cursor para examinar se haya restablecido con MQGMO_BROWSE_FIRST, o volviendo a abrir la cola.

La opción MQGMO_MSG_UNDER_CURSOR se puede utilizar con una llamada MQGET sin examinar si es necesario, para eliminar el mensaje de la cola.

El cursor para examinar no se mueve mediante llamadas MQGET no examinar utilizando el mismo descriptor de contexto *Hobj* .

Especifique la opción MQGMO_LOCK con esta opción para bloquear el mensaje que se examina.

Puede especificar MQGMO_BROWSE_NEXT con cualquier combinación válida de las opciones MQGMO_* y MQMO_* que controlan el proceso de mensajes en grupos y segmentos de mensajes lógicos.

Si especifica MQGMO_LOGICAL_ORDER, los mensajes se examinan en orden lógico. Si omite esta opción, los mensajes se examinan en orden físico. Si especifica MQGMO_BROWSE_FIRST, puede conmutar entre el orden lógico y el orden físico. Las llamadas MQGET subsiguientes que utilizan MQGMO_BROWSE_NEXT examinan la cola en el mismo orden que la llamada más reciente que ha especificado MQGMO_BROWSE_FIRST para el descriptor de contexto de cola. La llamada falla con el código de razón MQRC_INCONSISTENT_BROWSE si no se cumple esta condición.

Nota: Tenga especial cuidado al utilizar una llamada MQGET para examinar más allá del final de un grupo de mensajes si no se especifica MQGMO_LOGICAL_ORDER . Por ejemplo, supongamos que el último mensaje del grupo precede al primer mensaje del grupo de la cola. Si se utiliza MQGMO_BROWSE_NEXT para examinar más allá del final del grupo, si se especifica MQMO_MATCH_MSG_SEQ_NUMBER con *MsgSeqNumber* establecido en 1 , se devuelve el primer mensaje del grupo ya examinado. Este resultado puede producirse inmediatamente, o una serie de llamadas de MQGET más tarde si hay grupos intervinientes. La misma consideración se aplica a un mensaje lógico que no está en un grupo.

La información de grupo y segmento para las llamadas de examen se conserva por separado de la información para las llamadas que eliminan mensajes de la cola.

MQGMO_BROWSE_MSG_UNDER_CURSOR
Recupere el mensaje al que apunta el cursor para examinar de forma no destructiva, independientemente de las opciones MQMO_* especificadas en el campo *MatchOptions* en MQGMO.

MQGMO_BROWSE_MSG_UNDER_CURSOR no es válido con ninguna de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

También es un error si la cola no se ha abierto para examinar.

El mensaje al que apunta el cursor para examinar es el último que se ha recuperado utilizando la opción MQGMO_BROWSE_FIRST o la opción MQGMO_BROWSE_NEXT . La llamada falla si no se ha emitido ninguna de estas llamadas para esta cola desde que se abrió. La llamada también falla si el mensaje que estaba bajo el cursor para examinar se ha recuperado de forma destructiva.

Esta llamada no cambia la posición del cursor para examinar.

La opción MQGMO_MSG_UNDER_CURSOR se puede utilizar con una llamada MQGET sin examinar, para eliminar el mensaje de la cola.

El cursor para examinar no se mueve mediante una llamada MQGET de no examinar, aunque se utilice el mismo descriptor de contexto *Hobj* . Tampoco se mueve mediante una llamada MQGET de examen que devuelve un código de terminación de MQCC_FAILED, o un código de razón de MQRC_TRUNCATED_MSG_FAILED.

Si se especifica MQGMO_BROWSE_MSG_UNDER_CURSOR con MQGMO_LOCK:

- Si ya hay un mensaje bloqueado, debe ser el que está bajo el cursor, de modo que se devuelva sin desbloquear ni bloquear de nuevo. El mensaje permanece bloqueado.
- Si no hay ningún mensaje bloqueado y hay un mensaje bajo el cursor para examinar, se bloquea y se devuelve a la aplicación. Si no hay ningún mensaje bajo el cursor para examinar, la llamada falla.

Si se especifica MQGMO_BROWSE_MSG_UNDER_CURSOR sin MQGMO_LOCK:

- Si ya hay un mensaje bloqueado, debe ser el que está bajo el cursor. El mensaje se devuelve a la aplicación y, a continuación, se desbloquea. Puesto que el mensaje está ahora desbloqueado, no hay garantía de que la misma aplicación pueda volver a examinarlo o recuperarlo de forma destructiva. Es posible que otra aplicación lo haya recuperado de forma destructiva obteniendo mensajes de la cola.
- Si no hay ningún mensaje bloqueado y hay un mensaje bajo el cursor para examinar, se devuelve a la aplicación. Si no hay ningún mensaje bajo el cursor para examinar, la llamada falla.

Si se especifica MQGMO_COMPLETE_MSG con MQGMO_BROWSE_MSG_UNDER_CURSOR, el cursor para examinar debe identificar un mensaje cuyo campo *Offset* en MQMD sea cero. Si esta condición no se cumple, la llamada falla con el código de razón MQRC_INVALID_MSG_UNDER_CURSOR.

La información de grupo y segmento para las llamadas de examen se conserva por separado de la información para las llamadas que eliminan mensajes de la cola.

MQGMO_MSG_UNDER_CURSOR
Recupere el mensaje al que apunta el cursor para examinar, independientemente de las opciones MQMO_* especificadas en el campo *MatchOptions* en MQGMO. El mensaje se elimina de la cola.

El mensaje al que apunta el cursor para examinar es el último que se ha recuperado utilizando la opción MQGMO_BROWSE_FIRST o la opción MQGMO_BROWSE_NEXT .

Si se especifica MQGMO_COMPLETE_MSG con MQGMO_MSG_UNDER_CURSOR, el cursor para examinar debe identificar un mensaje cuyo campo *Offset* en MQMD sea cero. Si esta condición no se cumple, la llamada falla con el código de razón MQRC_INVALID_MSG_UNDER_CURSOR.

Esta opción no es válida con ninguna de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_UNLOCK

También es un error si la cola no se ha abierto tanto para examinar como para entrada. Si el cursor para examinar no apunta actualmente a un mensaje recuperable, la llamada MQGET devuelve un error.

MQGMO_MARK_BROWSE_HANDLE
El mensaje devuelto por un MQGETsatisfactorio, o identificado por el *MsgToken*devuelto, está marcado. La marca es específica del descriptor de objeto utilizado en la llamada.

El mensaje no se elimina de la cola.

MQGMO_MARK_BROWSE_HANDLE sólo es válido si también se especifica una de las opciones siguientes:

- MQGMO_BROWSE_FIRST

- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

MQGMO_MARK_BROWSE_HANDLE no es válido con ninguna de las opciones siguientes:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

El mensaje permanece en este estado hasta que se produce uno de los sucesos siguientes:

- El descriptor de contexto de objeto en cuestión está cerrado, ya sea normalmente o de otro modo.
- El mensaje no está marcado para este descriptor de contexto mediante una llamada a MQGET con la opción MQGMO_UNMARK_BROWSE_HANDLE.
- El mensaje se devuelve de una llamada a MQGETdestructivo, que se completa con MQCC_OK o MQCC_WARNING. El estado del mensaje permanece modificado incluso si el MQGET se retrotrae más tarde.
- El mensaje caduca.

MQGMO_MARK_BROWSE_CO_OP
El mensaje devuelto por un MQGETsatisfactorio, o identificado por el *MsgToken*devuelto, se marca para todos los descriptores de contexto del conjunto cooperante.

La marca de nivel de cooperación se suma a cualquier marca de nivel de descriptor de contexto que se pueda haber establecido.

El mensaje no se elimina de la cola.

MQGMO_MARK_BROWSE_CO_OP sólo es válido si el descriptor de objeto utilizado ha sido devuelto por una llamada a MQOPEN que ha especificado MQOO_CO_OP. También debe especificar una de las siguientes opciones MQGMO :

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

Esta opción no es válida con ninguna de las opciones siguientes:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

Si el mensaje ya está marcado y no se ha especificado la opción MQGMO_UNMARKED_BROWSE_MSG , la llamada falla con MQCC_FAILED y el código de razón MQRC_MSG_MARKED_BROWSE_CO_OP.

El mensaje permanece en este estado hasta que se produce uno de los sucesos siguientes:

- Se cierran todos los manejadores de objetos del conjunto cooperante.
- El mensaje no está marcado para los navegadores que cooperan mediante una llamada a MQGET con la opción MQGMO_UNMARK_BROWSE_CO_OP.
- El gestor de colas desmarca automáticamente el mensaje.

- El mensaje se devuelve desde una llamada a un MQGETque no es examinar. El estado del mensaje permanece modificado incluso si el MQGET se retrotrae más tarde.
- El mensaje caduca.

MQGMO_UNMARKED_BROWSE_MSG
Una llamada a MQGET que especifica MQGMO_UNMARKED_BROWSE_MSG devuelve un mensaje que se considera no marcado para su descriptor de contexto. No devuelve un mensaje si el mensaje se ha marcado para su descriptor de contexto. Tampoco devuelve el mensaje si la cola se ha abierto mediante una llamada a MQOPEN, con la opción MQOO_CO_OP, y el mensaje ha sido marcado por un miembro del conjunto cooperante.

Esta opción no es válida con ninguna de las opciones siguientes:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

MQGMO_UNMARK_BROWSE_CO_OP
Después de una llamada a MQGET que especifica esta opción, los descriptores de contexto abiertos del conjunto de descriptores de contexto cooperantes ya no consideran que el mensaje se marque para el conjunto cooperante. El mensaje se sigue considerando marcado a nivel de descriptor de contexto si se ha marcado a nivel de descriptor de contexto antes de esta llamada.

El uso de MQGMO_UNMARK_BROWSE_CO_OP sólo es válido con un descriptor de contexto devuelto por una llamada satisfactoria a MQOPEN con la opción MQOO_CO_OP. MQGET se ejecuta correctamente aunque el mensaje no se considere marcado por el conjunto de descriptores de contexto cooperantes.

MQGMO_UNMARK_BROWSE_CO_OP no es válido en una llamada MQGET sin examinar, o con cualquiera de las opciones siguientes:

- 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
Después de una llamada a MQGET que especifica esta opción, el mensaje ubicado ya no se considera marcado por este descriptor de contexto.

La llamada se realiza correctamente incluso si el mensaje no está marcado para este descriptor de contexto.

Esta opción no es válida en una llamada MQGET sin examinar, o con cualquiera de las opciones siguientes:

- 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

Opciones de bloqueo: Las opciones siguientes están relacionadas con el bloqueo de mensajes en la cola:

MQGMO_LOCK
Bloquee el mensaje que se examina, de modo que el mensaje se vuelva invisible para cualquier otro descriptor de contexto abierto para la cola. La opción sólo se puede especificar si también se especifica una de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

Sólo se puede bloquear un mensaje para cada descriptor de contexto de cola. El mensaje puede ser un mensaje lógico o un mensaje físico:

- Si especifica MQGMO_COMPLETE_MSG, todos los segmentos de mensaje que componen el mensaje lógico se bloquean en el descriptor de contexto de cola. Todos los mensajes deben estar presentes en la cola y disponibles para su recuperación.
- Si omite MQGMO_COMPLETE_MSG, sólo se bloquea un único mensaje físico en el descriptor de contexto de cola. Si este mensaje es un segmento de un mensaje lógico, el segmento bloqueado impide que otras aplicaciones utilicen MQGMO_COMPLETE_MSG para recuperar o examinar el mensaje lógico.

El mensaje bloqueado es siempre el que está bajo el cursor para examinar. El mensaje se puede eliminar de la cola mediante una llamada MQGET posterior que especifique la opción MQGMO_MSG_UNDER_CURSOR . Otras llamadas de MQGET que utilizan el descriptor de contexto de cola también pueden eliminar el mensaje (por ejemplo, una llamada que especifica el identificador de mensaje del mensaje bloqueado).

Si la llamada devuelve el código de terminación MQCC_FAILEDo MQCC_WARNING con el código de razón MQRC_TRUNCATED_MSG_FAILED, no se bloquea ningún mensaje.

Si la aplicación no elimina el mensaje de la cola, el bloqueo se libera mediante una de las acciones siguientes:

- Emitiendo otra llamada MQGET para este descriptor de contexto, especificando MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT. El bloqueo se libera si la llamada se completa con MQCC_OK o MQCC_WARNING. El mensaje permanece bloqueado si la llamada se completa con MQCC_FAILED. No obstante, se aplican las excepciones siguientes:
  - El mensaje no se desbloquea si se devuelve MQCC_WARNING con MQRC_TRUNCATED_MSG_FAILED.
  - El mensaje se desbloquea si se devuelve MQCC_FAILED con MQRC_NO_MSG_AVAILABLE.

  Si también especifica MQGMO_LOCK, el mensaje devuelto se bloquea. Si omite MQGMO_LOCK, no hay ningún mensaje bloqueado después de la llamada.

  Si especifica MQGMO_WAITy no hay ningún mensaje disponible inmediatamente, el mensaje original se desbloquea antes del inicio de la espera.
- Emitiendo otra llamada MQGET para este descriptor de contexto, con MQGMO_BROWSE_MSG_UNDER_CURSOR, sin MQGMO_LOCK. El bloqueo se libera si la llamada se completa con MQCC_OK o MQCC_WARNING. El mensaje permanece bloqueado si la llamada se completa con MQCC_FAILED. Sin embargo, se aplica la siguiente excepción:
  - El mensaje no se desbloquea si se devuelve MQCC_WARNING con MQRC_TRUNCATED_MSG_FAILED.
- Emitiendo otra llamada MQGET para este descriptor de contexto con MQGMO_UNLOCK.

- Emisión de una llamada MQCLOSE utilizando el descriptor de contexto. El MQCLOSE puede estar implícito, provocado por la finalización de la aplicación.

No es necesaria ninguna opción MQOPEN especial para especificar MQGMO_LOCK, que no sea MQOO_BROWSE, que es necesaria para especificar una opción de examen que la acompañe.

MQGMO_LOCK no es válido con ninguna de las opciones siguientes:

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

MQGMO_LOCK no es posible cuando se utiliza un cliente IBM WebSphere MQ en HP Integrity NonStop Server a un gestor de colas z/OS cuando se coordina mediante TMF.

MQGMO_UNLOCK
El mensaje que se va a desbloquear debe haber sido bloqueado previamente por una llamada MQGET con la opción MQGMO_LOCK . Si no hay ningún mensaje bloqueado para este descriptor de contexto, la llamada se completa con MQCC_WARNING y MQRC_NO_MSG_LOCKED.

Los parámetros *MsgDesc*, *BufferLength*, *Buffer* y *DataLength* no se comprueban ni modifican si especifica MQGMO_UNLOCK. No se devuelve ningún mensaje en *Buffer*.

No es necesaria ninguna opción de apertura especial para especificar MQGMO_UNLOCK (aunque se necesita MQOO_BROWSE para emitir la solicitud de bloqueo en primer lugar).

Esta opción no es válida con ninguna opción excepto la siguiente:

- MQGMO_NO_WAIT
- MQGMO_NO_SYNCPOINT

Ambas opciones se asumen tanto si se especifican como si no.

Opciones de datos de mensaje: Las opciones siguientes están relacionadas con el proceso de los datos de mensaje cuando el mensaje se lee de la cola:

MQGMO_ACCEPT_TRUNCATED_MSG
Si el almacenamiento intermedio de mensajes es demasiado pequeño para contener el mensaje completo, permita que la llamada MQGET llene el almacenamiento intermedio. MQGET llena el almacenamiento intermedio con la mayor parte del mensaje que puede. Emite un código de finalización de aviso y completa su proceso. Esto significa que:

- Al examinar mensajes, el cursor para examinar se avanza al mensaje devuelto.
- Al eliminar mensajes, el mensaje devuelto se elimina de la cola.
- Se devuelve el código de razón MQRC_TRUNCATED_MSG_ACCEPTED si no se produce ningún otro error.

Sin esta opción, el almacenamiento intermedio se sigue llenando con la mayor cantidad de mensajes que pueda contener. Se emite un código de finalización de aviso, pero el proceso no se ha completado. Esto significa que:

- Al examinar mensajes, el cursor para examinar no está avanzado.
- Al eliminar mensajes, el mensaje no se elimina de la cola.
- Se devuelve el código de razón MQRC_TRUNCATED_MSG_FAILED si no se produce ningún otro error.

MQGMO_CONVERT
Esta opción convierte los datos de aplicación en el mensaje para que se ajusten a los valores *CodedCharSetId* y *Encoding* especificados en el parámetro *MsgDesc* en la llamada MQGET . Los datos se convierten antes de que se copien en el parámetro *Buffer* .

El campo *Format* especificado cuando se colocó el mensaje es asumido por el proceso de conversión para identificar la naturaleza de los datos en el mensaje. El gestor de colas convierte los datos de

mensaje para formatos incorporados y una salida escrita por el usuario para otros formatos. Consulte "salida de conversión de datos" en la página 888 para obtener más información sobre la salida de conversión de datos.

- Si la conversión es satisfactoria, los campos *CodedCharSetId* y *Encoding* especificados en el parámetro *MsgDesc* no se modifican al volver de la llamada MQGET .
- Si sólo falla la conversión, los datos del mensaje se devuelven sin convertir. Los campos *CodedCharSetId* y *Encoding* de *MsgDesc* se establecen en los valores del mensaje sin convertir. El código de terminación es MQCC_WARNING en este caso.

En cualquier caso, estos campos describen el identificador de juego de caracteres y la codificación de los datos de mensaje que se devuelven en el parámetro *Buffer* .

Consulte el campo *Format* descrito en "MQMD - Descriptor de mensaje" en la página 392 para obtener una lista de nombres de formato para los que el gestor de colas realiza la conversión.

Opciones de grupo y segmento: Las opciones siguientes están relacionadas con el proceso de mensajes en grupos y segmentos de mensajes lógicos. Antes de las descripciones de las opciones, a continuación se muestran algunas definiciones de términos importantes:

Mensaje físico
Un mensaje físico es la unidad de información más pequeña que se puede colocar o eliminar de una cola. A menudo se corresponde con la información especificada o recuperada en una sola llamada MQPUT, MQPUT1o MQGET . Cada mensaje físico tiene su propio descriptor de mensaje, MQMD. Normalmente, los mensajes físicos se distinguen por valores diferentes para el identificador de mensaje, el campo *MsgId* en MQMD. El gestor de colas no impone valores diferentes.

Mensaje lógico
Un mensaje lógico es una única unidad de información de aplicación. En ausencia de restricciones del sistema, un mensaje lógico es el mismo que un mensaje físico. Si los mensajes lógicos son grandes, las restricciones del sistema pueden hacer aconsejable o necesario dividir un mensaje lógico en dos o más mensajes físicos, denominados segmentos.

Un mensaje lógico que se ha segmentado consta de dos o más mensajes físicos que tienen el mismo identificador de grupo no nulo, *GroupId* campo en MQMD. Tienen el mismo número de secuencia de mensaje, *MsgSeqNumber* campo en MQMD. Los segmentos se distinguen por valores diferentes para el desplazamiento de segmento, campo *Offset* en MQMD. El desplazamiento de segmento es el desplazamiento de los datos en el mensaje físico desde el inicio de los datos en el mensaje lógico. Puesto que cada segmento es un mensaje físico, los segmentos de un mensaje lógico normalmente tienen identificadores de mensaje diferentes.

Un mensaje lógico que no se ha segmentado, pero para el que la aplicación emisora ha permitido la segmentación, también tiene un identificador de grupo no nulo. En este caso, sólo hay un mensaje físico con ese identificador de grupo si el mensaje lógico no pertenece a un grupo de mensajes. Los mensajes lógicos, para los que la aplicación emisora ha inhibido la segmentación, tienen un identificador de grupo nulo, MQGI_NONE, a menos que el mensaje lógico pertenezca a un grupo de mensajes.

Grupo de mensajes
Un grupo de mensajes es un conjunto de uno o más mensajes lógicos que tienen el mismo identificador de grupo no nulo. Los mensajes lógicos del grupo se distinguen por diferentes valores para el número de secuencia de mensaje. El número de secuencia es un entero comprendido entre 1 y n, donde n es el número de mensajes lógicos del grupo. Si uno o varios de los mensajes lógicos están segmentados, hay más de n mensajes físicos en el grupo.

MQGMO_LOGICAL_ORDER
MQGMO_LOGICAL_ORDER controla el orden en el que las llamadas MQGET sucesivas devuelven los mensajes para el descriptor de contexto de cola. La opción debe especificarse en cada llamada.

Si se especifica MQGMO_LOGICAL_ORDER para llamadas MQGET sucesivas para el mismo descriptor de contexto de cola, los mensajes de los grupos se devuelven en el orden de sus números de secuencia de mensajes. Los segmentos de mensajes lógicos se devuelven en el orden indicado por

sus desplazamientos de segmento. Este orden puede ser diferente del orden en el que se producen estos mensajes y segmentos en la cola.

Nota: La especificación de MQGMO_LOGICAL_ORDER no tiene consecuencias adversas en los mensajes que no pertenecen a grupos y que no son segmentos. En efecto, estos mensajes se tratan como si cada uno perteneciera a un grupo de mensajes que consta de un solo mensaje. Es seguro especificar MQGMO_LOGICAL_ORDER cuando se recuperan mensajes de colas que contienen una mezcla de mensajes en grupos, segmentos de mensajes y mensajes no segmentados que no están en grupos.

Para devolver los mensajes en el orden necesario, el gestor de colas conserva la información de grupo y segmento entre llamadas MQGET sucesivas. La información de grupo y segmento identifica el grupo de mensajes actual y el mensaje lógico actual para el descriptor de contexto de cola. También identifica la posición actual dentro del grupo y el mensaje lógico, y si los mensajes se están recuperando dentro de una unidad de trabajo. Puesto que el gestor de colas conserva esta información, la aplicación no necesita establecer la información de grupo y segmento antes de cada llamada MQGET . Específicamente, significa que la aplicación no necesita establecer los campos *GroupId*, *MsgSeqNumber*y *Offset* en MQMD. Sin embargo, la aplicación debe establecer la opción MQGMO_SYNCPOINT o MQGMO_NO_SYNCPOINT correctamente en cada llamada.

Cuando se abre la cola, no hay ningún grupo de mensajes actual ni ningún mensaje lógico actual. Un grupo de mensajes se convierte en el grupo de mensajes actual cuando la llamada MQGET devuelve un mensaje que tiene el distintivo MQMF_MSG_IN_GROUP . Con MQGMO_LOGICAL_ORDER especificado en llamadas sucesivas, ese grupo sigue siendo el grupo actual hasta que se devuelve un mensaje que tiene:

- MQMF_LAST_MSG_IN_GROUP sin MQMF_SEGMENT (es decir, el último mensaje lógico del grupo no está segmentado), o
- MQMF_LAST_MSG_IN_GROUP con MQMF_LAST_SEGMENT (es decir, el mensaje devuelto es el último segmento del último mensaje lógico del grupo).

Cuando se devuelve un mensaje de este tipo, el grupo de mensajes termina y, al finalizar correctamente la llamada MQGET , ya no hay un grupo actual. De forma similar, un mensaje lógico se convierte en el mensaje lógico actual cuando la llamada MQGET devuelve un mensaje que tiene el distintivo MQMF_SEGMENT . El mensaje lógico termina cuando se devuelve el mensaje que tiene el distintivo MQMF_LAST_SEGMENT .

Si no se especifica ningún criterio de selección, las llamadas MQGET sucesivas devuelven, en el orden correcto, los mensajes para el primer grupo de mensajes de la cola. A continuación, devuelven los mensajes para el segundo grupo de mensajes, y así sucesivamente, hasta que no haya más mensajes disponibles. Es posible seleccionar los grupos de mensajes concretos devueltos especificando una o más de las opciones siguientes en el campo *MatchOptions* :

- MQMO_MATCH_MSG_ID
- MQMO_MATCH_CORREL_ID
- MQMO_MATCH_GROUP_ID

Sin embargo, estas opciones sólo son efectivas cuando no hay ningún grupo de mensajes actual o mensaje lógico. Consulte el campo *MatchOptions* descrito en “MQGMO-Opciones de obtención de mensajes” en la página 344 para obtener más detalles.

La Tabla 506 en la página 364 muestra los valores de los campos *MsgId*, *CorrelId*, *GroupId*, *MsgSeqNumber*y *Offset* que el gestor de colas busca al intentar encontrar un mensaje para devolver en la llamada MQGET . Las reglas se aplican tanto a la eliminación de mensajes de la cola como a la exploración de mensajes en la cola. En la tabla, significa Sí o No:

LOG ORD
Indica si se ha especificado la opción MQGMO_LOGICAL_ORDER en la llamada.

Cur grp
Indica si existe un grupo de mensajes actual antes de la llamada.

Cur log msg
Indica si existe un mensaje lógico actual antes de la llamada.

Otras columnas
Mostrar los valores que busca el gestor de colas. Anterior indica el valor devuelto para el campo en el mensaje anterior para el descriptor de contexto de cola.

Tabla 506. Opciones MQGET relacionadas con mensajes en grupos y segmentos de mensajes lógicos

Opciones especificadas	Estado de grupo y mensaje lógico antes de la llamada		Valores que el gestor de colas busca				
LOGORD	**Cur grp**	**Cur log msg**	*MsgId*	*CorrelId*	*GroupId*	*MsgSeqNumber*	*Offset*
Sí	No	No	Controlado por *MatchOptions*	Controlado por *MatchOptions*	Controlado por *MatchOptions*	1	0
Sí	No	Sí	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	1	Desplazamiento anterior + longitud de segmento anterior
Sí	Sí	No	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	Número de secuencia anterior + 1	0
Sí	Sí	Sí	Cualquier identificador de mensaje	Cualquier identificador de correlación	Identificador de grupo anterior	Número de secuencia anterior	Desplazamiento anterior + longitud de segmento anterior
No	Cualquiera de los dos	Cualquiera de los dos	Controlado por *MatchOptions*	Controlado por *MatchOptions*	Controlado por *MatchOptions*	Controlado por *MatchOptions*	Controlado por *MatchOptions*

Si hay varios grupos de mensajes en la cola y son elegibles para la devolución, los grupos se devuelven en el orden determinado por la posición en la cola del primer segmento del primer mensaje lógico de cada grupo. Es decir, los mensajes físicos que tienen números de secuencia de mensajes de 1, y desplazamientos de 0, determinan el orden en el que se devuelven los grupos elegibles.

La opción MQGMO_LOGICAL_ORDER afecta a las unidades de trabajo como se indica a continuación:

- Si el primer mensaje lógico o segmento de un grupo se recupera dentro de una unidad de trabajo, todos los demás mensajes lógicos y segmentos del grupo deben recuperarse dentro de una unidad de trabajo, si se utiliza el mismo descriptor de contexto de cola. Sin embargo, no es necesario recuperarlos dentro de la misma unidad de trabajo. Esto permite que un grupo de mensajes que consta de muchos mensajes físicos se divida entre dos o más unidades de trabajo consecutivas para el descriptor de contexto de cola.
- Si el primer mensaje lógico o segmento de un grupo *no* se recupera dentro de una unidad de trabajo, y se utiliza el mismo descriptor de contexto de cola, ninguno de los otros mensajes lógicos y segmentos del grupo se puede recuperar dentro de una unidad de trabajo.

Si no se cumplen estas condiciones, la llamada MQGET falla con el código de razón MQRC_INCONSISTENT_UOW.

Cuando se especifica MQGMO_LOGICAL_ORDER , el MQGMO proporcionado en la llamada MQGET no debe ser menor que MQGMO_VERSION_2, y el MQMD no debe ser menor que MQMD_VERSION_2. Si no se cumple esta condición, la llamada falla con el código de razón MQRC_WRONG_GMO_VERSION o MQRC_WRONG_MD_VERSION, según corresponda.

Si MQGMO_LOGICAL_ORDER *no* se especifica para llamadas MQGET sucesivas para el descriptor de contexto de cola, los mensajes se devuelven sin tener en cuenta si pertenecen a grupos de mensajes o si son segmentos de mensajes lógicos. Esto significa que los mensajes o segmentos de un grupo o mensaje lógico determinado pueden devolverse desordenados, o mezclados con mensajes o segmentos de otros grupos o mensajes lógicos, o con mensajes que no están en grupos y no son segmentos. En esta situación, los mensajes concretos devueltos por llamadas MQGET sucesivas se controlan mediante las opciones MQMO_* especificadas en dichas llamadas (consulte el campo *MatchOptions* descrito en "MQGMO-Opciones de obtención de mensajes" en la página 344 para obtener detalles de estas opciones).

Esta es la técnica que se puede utilizar para reiniciar un grupo de mensajes o un mensaje lógico en el medio, después de que se haya producido una anomalía del sistema. Cuando se reinicia el sistema, la aplicación puede establecer los campos *GroupId*, *MsgSeqNumber*, *Offset* y *MatchOptions* en los valores adecuados y, a continuación, emitir la llamada MQGET con MQGMO_SYNCPOINT o MQGMO_NO_SYNCPOINT establecido, pero *sin* especificar MQGMO_LOGICAL_ORDER. Si esta llamada es satisfactoria, el gestor de colas conserva la información de grupo y segmento, y las llamadas MQGET subsiguientes que utilizan ese descriptor de contexto de cola pueden especificar MQGMO_LOGICAL_ORDER como normal.

La información de grupo y segmento que el gestor de colas retiene para la llamada MQGET es independiente de la información de grupo y segmento que retiene para la llamada MQPUT. Además, el gestor de colas conserva información separada para:

- Llamadas de MQGET que eliminan mensajes de la cola.
- Llamadas de MQGET que examinan mensajes en la cola.

Para cualquier descriptor de contexto de cola determinado, la aplicación puede combinar llamadas MQGET que especifiquen MQGMO_LOGICAL_ORDER con llamadas MQGET que no lo hagan. Sin embargo, tenga en cuenta los siguientes puntos:

- Si omite MQGMO_LOGICAL_ORDER, cada llamada MQGET satisfactoria hace que el gestor de colas establezca la información de grupo y segmento guardada en los valores correspondientes al mensaje devuelto; esto sustituye a la información de grupo y segmento existente retenida por el gestor de colas para el descriptor de contexto de cola. Sólo se modifica la información adecuada para la acción de la llamada (examinar o eliminar).
- Si omite MQGMO_LOGICAL_ORDER, la llamada no falla si hay un grupo de mensajes actual o un mensaje lógico; la llamada podría tener éxito con un código de terminación MQCC_WARNING . La Tabla 507 en la página 366 muestra los distintos casos que se pueden presentar. En estos casos, si el código de terminación no es MQCC_OK, el código de razón es uno de los siguientes (según corresponda):
  - MQRC_INCOMPLETE_GROUP
  - MQRC_INCOMPLETE_MSG
  - MQRC_INCONSISTENT_UOW

  Nota: El gestor de colas no comprueba la información de grupo y segmento al examinar una cola o al cerrar una cola que se ha abierto para examinar pero no para entrar; en estos casos, el código de finalización siempre es MQCC_OK (suponiendo que no haya otros errores).

Tabla 507. Resultado cuando la llamada MQGET o MQCLOSE no es coherente con la información de grupo y segmento

La llamada actual es	La llamada anterior era **MQGET** con **MQGMO_LOGICAL_ORDER**	La llamada anterior era **MQGET** sin **MQGMO_LOGICAL_ORDER**
MQGET con MQGMO_LOGICAL_ORDER	MQCC_FAILED	MQCC_FAILED
MQGET sin MQGMO_LOGICAL_ORDER	MQCC_WARNING	MQCC_OK
MQCLOSE con un grupo o mensaje lógico sin terminar	MQCC_WARNING	MQCC_OK

Las aplicaciones que desean recuperar mensajes y segmentos en orden lógico se recomiendan para especificar MQGMO_LOGICAL_ORDER, ya que esta es la opción más sencilla de utilizar. Esta opción hace que la aplicación no tenga que gestionar la información de grupo y segmento, pues el gestor de colas lo hace en su lugar. Sin embargo, es posible que las aplicaciones especializadas necesiten más control que el proporcionado por la opción MQGMO_LOGICAL_ORDER , y esto se puede lograr no especificando esa opción. A continuación, la aplicación debe asegurarse de que los campos *MsgId*, *CorrelId*, *GroupId*, *MsgSeqNumber*y *Offset* en MQMD, y las opciones MQMO_* en *MatchOptions* en MQGMO, se hayan establecido correctamente, antes de cada llamada MQGET .

Por ejemplo, una aplicación que desea *reenviar* mensajes físicos que recibe, sin tener en cuenta si estos mensajes están en grupos o segmentos de mensajes lógicos, *no* debe especificar MQGMO_LOGICAL_ORDER. En una red compleja con múltiples rutas entre los gestores de colas de emisión y recepción, los mensajes físicos pueden llegar fuera de secuencia. Al no especificar MQGMO_LOGICAL_ORDER, ni el MQPMO_LOGICAL_ORDER correspondiente en la llamada MQPUT, la aplicación de reenvío puede recuperar y reenviar cada mensaje físico tan pronto como llegue, sin tener que esperar a que llegue el siguiente en orden lógico.

Puede especificar MQGMO_LOGICAL_ORDER con cualquiera de las otras opciones de MQGMO_* y con varias de las opciones de MQMO_* en las circunstancias adecuadas (consulte más arriba).

- En z/OS, esta opción está soportada para colas privadas y compartidas, pero la cola debe tener un tipo de índice de MQIT_GROUP_ID. Para colas compartidas, el objeto CFSTRUCT con el que se correlaciona la cola debe estar en CFLEVEL (3) o CFLEVEL (4).
- En AIX, HP-UX, IBM i, Solaris, Linux, Windows, además de clientes MQI de WebSphere MQ conectados a estos sistemas, esta opción está soportada para todas las colas locales.

MQGMO_COMPLETE_MSG
Sólo la llamada MQGET puede devolver un mensaje lógico completo. Si el mensaje lógico está segmentado, el gestor de colas vuelve a ensamblar los segmentos y devuelve el mensaje lógico completo a la aplicación; el hecho de que el mensaje lógico se haya segmentado no es aparente para la aplicación que lo recupera.

Nota: Esta es la única opción que hace que el gestor de colas vuelva a ensamblar segmentos de mensajes. Si no se especifica, los segmentos se devuelven individualmente a la aplicación si están presentes en la cola (y satisfacen los otros criterios de selección especificados en la llamada MQGET). Las aplicaciones que no desean recibir segmentos individuales siempre deben especificar MQGMO_COMPLETE_MSG.

Para utilizar esta opción, la aplicación debe proporcionar un almacenamiento intermedio lo suficientemente grande como para acomodar el mensaje completo o especificar la opción MQGMO_ACCEPT_TRUNCATED_MSG .

Si la cola contiene mensajes segmentados con algunos de los segmentos que faltan (quizás porque se han retrasado en la red y todavía no han llegado), la especificación de MQGMO_COMPLETE_MSG impide la recuperación de segmentos que pertenecen a mensajes lógicos incompletos. Sin embargo, estos segmentos de mensajes siguen contribuyendo al valor del atributo de cola *CurrentQDepth* ; esto

significa que es posible que no haya mensajes lógicos recuperables, aunque *CurrentQDepth* sea mayor que cero.

Para los mensajes *persistentes* , el gestor de colas puede volver a ensamblar los segmentos sólo dentro de una unidad de trabajo:

- Si la llamada MQGET está funcionando dentro de una unidad de trabajo definida por el usuario, se utiliza dicha unidad de trabajo. Si la llamada falla durante el proceso de reensamblaje, el gestor de colas restablece en la cola los segmentos que se han eliminado durante el reensamblaje. Sin embargo, la anomalía no impide que la unidad de trabajo se confirme correctamente.
- Si la llamada está funcionando fuera de una unidad de trabajo definida por el usuario, y no existe ninguna unidad de trabajo definida por el usuario, el gestor de colas crea una unidad de trabajo para la duración de la llamada. Si la llamada es satisfactoria, el gestor de colas confirma la unidad de trabajo automáticamente (la aplicación no necesita hacerlo). Si la llamada falla, el gestor de colas restituye la unidad de trabajo.
- Si la llamada está funcionando fuera de una unidad de trabajo definida por el usuario, pero existe una unidad de trabajo definida por el usuario, el gestor de colas no puede volver a ensamblarse. Si el mensaje no necesita reensamblarse, la llamada puede seguir siendo satisfactoria. Pero si el mensaje requiere reensamblaje, la llamada falla con el código de razón MQRC_UOW_NOT_AVAILABLE.

Para los mensajes *no persistentes* , el gestor de colas no requiere que haya una unidad de trabajo disponible para realizar el reensamblaje.

Cada mensaje físico que es un segmento tiene su propio descriptor de mensaje. Para los segmentos que constituyen un único mensaje lógico, la mayoría de los campos del descriptor de mensaje son los mismos para todos los segmentos del mensaje lógico; normalmente son sólo los campos *MsgId*, *Offset*y *MsgFlags* los que difieren entre los segmentos del mensaje lógico. Sin embargo, si un segmento se coloca en una cola de mensajes no entregados en un gestor de colas intermedio, el manejador DLQ recupera el mensaje especificando la opción MQGMO_CONVERT , y esto puede hacer que se cambie el juego de caracteres o la codificación del segmento. Si el manejador DLQ envía correctamente el segmento en su camino, es posible que el segmento tenga un juego de caracteres o una codificación que difiera de los otros segmentos del mensaje lógico cuando el segmento llegue al gestor de colas de destino.

Un mensaje lógico que consta de segmentos en los que los campos *CodedCharSetId* y *Encoding* difieren no puede ser reensamblado por el gestor de colas en un único mensaje lógico. En su lugar, el gestor de colas vuelve a ensamblar y devuelve los primeros segmentos consecutivos al principio del mensaje lógico que tienen los mismos identificadores de juego de caracteres y codificaciones, y la llamada MQGET se completa con el código de terminación MQCC_WARNING y el código de razón MQRC_INCONSISTENT_CCSIDS o MQRC_INCONSISTENT_ENCODINGS, según corresponda. Esto sucede independientemente de si se especifica MQGMO_CONVERT . Para recuperar los segmentos restantes, la aplicación debe volver a emitir la llamada MQGET sin la opción MQGMO_COMPLETE_MSG , recuperando los segmentos uno por uno. MQGMO_LOGICAL_ORDER se puede utilizar para recuperar los segmentos restantes en orden.

Una aplicación que coloca segmentos también puede establecer otros campos en el descriptor de mensaje en valores que difieren entre segmentos. Sin embargo, no hay ninguna ventaja al hacerlo si la aplicación receptora utiliza MQGMO_COMPLETE_MSG para recuperar el mensaje lógico. Cuando el gestor de colas vuelve a ensamblar un mensaje lógico, devuelve en el descriptor de mensaje los valores del descriptor de mensaje para el *primer* segmento; la única excepción es el campo *MsgFlags* , que el gestor de colas establece para indicar que el mensaje reensamblado es el único segmento.

Si se especifica MQGMO_COMPLETE_MSG para un mensaje de informe, el gestor de colas realiza un proceso especial. El gestor de colas comprueba la cola para ver si todos los mensajes de informe de ese tipo de informe relacionados con los distintos segmentos del mensaje lógico están presentes en la cola. Si lo son, se pueden recuperar como un solo mensaje especificando MQGMO_COMPLETE_MSG. Para que esto sea posible, los mensajes de informe deben ser generados por un gestor de colas o un MCA que dé soporte a la segmentación, o la aplicación de origen debe solicitar al menos

100 bytes de datos de mensaje (es decir, se deben especificar las opciones MQRO_*_WITH_DATA o MQRO_*_WITH_FULL_DATA adecuadas). Si hay menos de la cantidad completa de datos de aplicación para un segmento, los bytes que faltan se sustituyen por nulos en el mensaje de informe devuelto.

Si se especifica MQGMO_COMPLETE_MSG con MQGMO_MSG_UNDER_CURSOR o MQGMO_BROWSE_MSG_UNDER_CURSOR, el cursor para examinar debe estar situado en un mensaje cuyo campo *Offset* en MQMD tenga un valor de 0. Si esta condición no se cumple, la llamada falla con el código de razón MQRC_INVALID_MSG_UNDER_CURSOR.

MQGMO_COMPLETE_MSG implica MQGMO_ALL_SEGMENTS_AVAILABLE, que por lo tanto no es necesario especificar.

MQGMO_COMPLETE_MSG se puede especificar con cualquiera de las otras opciones MQGMO_* aparte de MQGMO_SYNCPOINT_IF_PERSISTENT, y con cualquiera de las opciones MQMO_* aparte de MQMO_MATCH_OFFSET.

- En z/OS, esta opción está soportada para colas privadas y compartidas, pero la cola debe tener un tipo de índice de MQIT_GROUP_ID. Para colas compartidas, el objeto CFSTRUCT con el que la correlación de colas debe estar en CFLEVEL (3) o CFLEVEL (4).
- En AIX, HP-UX, IBM i, Solaris, Linux, Windows, además de clientes MQI de WebSphere MQ conectados a estos sistemas, esta opción está soportada para todas las colas locales.

MQGMO_ALL_MSGS_AVAILABLE

Los mensajes de un grupo pasan a estar disponibles para su recuperación sólo cuando están disponibles *todos* los mensajes del grupo. Si la cola contiene grupos de mensajes con algunos de los mensajes que faltan (quizás porque se han retrasado en la red y todavía no han llegado), especificar MQGMO_ALL_MSGS_AVAILABLE impide la recuperación de mensajes que pertenecen a grupos incompletos. Sin embargo, estos mensajes siguen contribuyendo al valor del atributo de cola *CurrentQDepth* ; esto significa que es posible que no haya grupos de mensajes recuperables, aunque *CurrentQDepth* sea mayor que cero. Si no hay otros mensajes recuperables, se devuelve el código de razón MQRC_NO_MSG_AVAILABLE después de que haya caducado el intervalo de espera especificado (si lo hay).

El proceso de MQGMO_ALL_MSGS_AVAILABLE depende de si también se especifica MQGMO_LOGICAL_ORDER :

- Si se especifican ambas opciones, MQGMO_ALL_MSGS_AVAILABLE tiene un efecto *sólo* cuando no hay ningún grupo o mensaje lógico actual. Si *hay* un grupo actual o un mensaje lógico, se ignora MQGMO_ALL_MSGS_AVAILABLE . Esto significa que MQGMO_ALL_MSGS_AVAILABLE puede permanecer activo al procesar mensajes en orden lógico.
- Si se especifica MQGMO_ALL_MSGS_AVAILABLE sin MQGMO_LOGICAL_ORDER, MQGMO_ALL_MSGS_AVAILABLE *siempre* tiene un efecto. Esto significa que la opción debe desactivarse después de que el primer mensaje del grupo se haya eliminado de la cola, para poder eliminar los mensajes restantes del grupo.

La finalización satisfactoria de una llamada MQGET especificando MQGMO_ALL_MSGS_AVAILABLE significa que en el momento en que se emitió la llamada MQGET , todos los mensajes del grupo estaban en la cola. Sin embargo, tenga en cuenta que otras aplicaciones todavía pueden eliminar mensajes del grupo (el grupo no está bloqueado para la aplicación que recupera el primer mensaje del grupo).

Si omite esta opción, los mensajes que pertenecen a grupos se pueden recuperar incluso cuando el grupo está incompleto.

MQGMO_ALL_MSGS_AVAILABLE implica MQGMO_ALL_SEGMENTS_AVAILABLE, que por lo tanto no es necesario especificar.

MQGMO_ALL_MSGS_AVAILABLE se puede especificar con cualquiera de las otras opciones de MQGMO_* y con cualquiera de las opciones de MQMO_* .

- En z/OS, esta opción está soportada para colas privadas y compartidas, pero la cola debe tener un tipo de índice de MQIT_GROUP_ID. Para colas compartidas, el objeto CFSTRUCT con el que la correlación de colas debe estar en CFLEVEL (3) o CFLEVEL (4).
- En AIX, HP-UX, IBM i, Solaris, Linux, Windows, además de clientes MQI de WebSphere MQ conectados a estos sistemas, esta opción está soportada para todas las colas locales.

MQGMO_ALL_SEGMENTS_AVAILABLE
Los segmentos de un mensaje lógico pasan a estar disponibles para su recuperación sólo cuando están disponibles *todos* los segmentos del mensaje lógico. Si la cola contiene mensajes segmentados con algunos de los segmentos que faltan (quizás porque se han retrasado en la red y todavía no han llegado), la especificación de MQGMO_ALL_SEGMENTS_AVAILABLE impide la recuperación de segmentos que pertenecen a mensajes lógicos incompletos. Sin embargo, estos segmentos siguen contribuyendo al valor del atributo de cola *CurrentQDepth* ; esto significa que es posible que no haya mensajes lógicos recuperables, aunque *CurrentQDepth* sea mayor que cero. Si no hay otros mensajes recuperables, se devuelve el código de razón MQRC_NO_MSG_AVAILABLE después de que haya caducado el intervalo de espera especificado (si lo hay).

El proceso de MQGMO_ALL_SEGMENTS_AVAILABLE depende de si también se especifica MQGMO_LOGICAL_ORDER :

- Si se especifican ambas opciones, MQGMO_ALL_SEGMENTS_AVAILABLE tiene un efecto *sólo* cuando no hay ningún mensaje lógico actual. Si *es* un mensaje lógico actual, MQGMO_ALL_SEGMENTS_AVAILABLE se ignora. Esto significa que MQGMO_ALL_SEGMENTS_AVAILABLE puede permanecer activo al procesar mensajes en orden lógico.
- Si se especifica MQGMO_ALL_SEGMENTS_AVAILABLE sin MQGMO_LOGICAL_ORDER, MQGMO_ALL_SEGMENTS_AVAILABLE *siempre* tiene un efecto. Esto significa que la opción debe desactivarse después de que el primer segmento del mensaje lógico se haya eliminado de la cola, para poder eliminar los segmentos restantes del mensaje lógico.

Si no se especifica esta opción, los segmentos de mensaje se pueden recuperar incluso cuando el mensaje lógico está incompleto.

Aunque tanto MQGMO_COMPLETE_MSG como MQGMO_ALL_SEGMENTS_AVAILABLE requieren que todos los segmentos estén disponibles antes de que se pueda recuperar cualquiera de ellos, el primero devuelve el mensaje completo, mientras que el segundo permite que los segmentos se recuperen uno por uno.

Si se especifica MQGMO_ALL_SEGMENTS_AVAILABLE para un mensaje de informe, el gestor de colas comprueba la cola para ver si hay al menos un mensaje de informe para cada uno de los segmentos que componen el mensaje lógico completo. Si existe, se cumple la condición MQGMO_ALL_SEGMENTS_AVAILABLE . Sin embargo, el gestor de colas no comprueba el *tipo* de los mensajes de informe presentes, por lo que puede haber una combinación de tipos de informe en los mensajes de informe relacionados con los segmentos del mensaje lógico. Como resultado, el éxito de MQGMO_ALL_SEGMENTS_AVAILABLE no implica que MQGMO_COMPLETE_MSG tenga éxito. Si *hay* una combinación de tipos de informe presentes para los segmentos de un mensaje lógico determinado, estos mensajes de informe se deben recuperar uno por uno.

Puede especificar MQGMO_ALL_SEGMENTS_AVAILABLE con cualquiera de las otras opciones de MQGMO_* y con cualquiera de las opciones de MQMO_* .

- En z/OS, esta opción está soportada para colas privadas y compartidas, pero la cola debe tener un tipo de índice de MQIT_GROUP_ID. Para colas compartidas, el objeto CFSTRUCT con el que la correlación de colas debe estar en CFLEVEL (3) o CFLEVEL (4).
- En AIX, HP-UX, IBM i, Solaris, Linux, Windows, además de clientes MQI de WebSphere MQ conectados a estos sistemas, esta opción está soportada para todas las colas locales.

Opciones de propiedad: Las opciones siguientes hacen relación a las propiedades del mensaje:

MQGMO_PROPERTIES_AS_Q_DEF

Las propiedades del mensaje, excepto las contenidas en el descriptor de mensaje (o extensión), deben representarse tal como se define en el atributo de cola *PropertyControl* . Si se proporciona un *MsgHandle* , esta opción se ignora y las propiedades del mensaje están disponibles a través de *MsgHandle*, a menos que el valor del atributo de cola *PropertyControl* sea MQPROP_FORCE_MQRFH2.

Se trata de la acción predeterminada si no se especifican opciones de propiedad.

MQGMO_PROPERTIES_IN_HANDLE

Las propiedades del mensaje deben estar disponibles a través de *MsgHandle*. Si no se proporciona ningún manejador de mensajes, la llamada fallará con la razón MQRC_HMSG_ERROR.

Nota: Si el mensaje lo lee más adelante una aplicación que no crea un manejador de mensajes, el gestor de colas coloca las propiedades de mensaje en una estructura MQRFH2 . Es posible que encuentre que la presencia de una cabecera MQRFH2 inesperada interrumpe el comportamiento de una aplicación existente.

MQGMO_NO_PROPERTIES

No se recuperarán las propiedades del mensaje, excepto las contenidas en el descriptor de mensaje (o extensión). Si se proporciona un *MsgHandle* , se ignorará.

MQGMO_PROPERTIES_FORCE_MQRFH2

Las propiedades del mensaje, excepto las contenidas en el descriptor de mensaje (o extensión), deben representarse utilizando cabeceras MQRFH2 . Esto proporciona compatibilidad con versiones anteriores para aplicaciones que esperan recuperar propiedades pero que no pueden cambiarse para utilizar manejadores de mensajes. Si se proporciona *MsgHandle*, se hará caso omiso del mismo.

MQGMO_PROPERTIES_COMPATIBILITY
Si el mensaje contiene una propiedad con el prefijo **"mcd."**, **"jms."**, **"usr."**o **"mqext."**, todas las propiedades del mensaje se entregan a la aplicación en una cabecera MQRFH2 . De lo contrario, todas las propiedades del mensaje, excepto las que se encuentran en el descriptor de mensaje (o extensión), se descartan y dejan de estar accesibles para la aplicación.

Opción predeterminada: Si ninguna de las opciones descritas es necesaria, se puede utilizar la opción siguiente:

MQGMO_NONE
Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados. MQGMO_NONE ayuda a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

El valor inicial del campo *Options* es MQGMO_NO_WAIT más MQGMO_PROPERTIES_AS_Q_DEF.

Reserved1 (MQCHAR)

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

Reserved2 (MQLONG)

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco. Este campo se ignora si *Version* es menor que **MQGMO_VERSION_4**.

ResolvedQName (MQCHAR48)

Es un campo de salida que el gestor de colas establece en el nombre local de la cola de la que se ha recuperado el mensaje, tal como se ha definido en el gestor de colas local. Esto es diferente del nombre utilizado para abrir la cola si:

- Se ha abierto una cola alias (en cuyo caso, se devuelve el nombre de la cola local a la que se ha resuelto el alias), o
- Se ha abierto una cola modelo (en cuyo caso, se devuelve el nombre de la cola local dinámica).

La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ReturnedLength (MQLONG)

Es un campo de salida que el gestor de colas establece en la longitud, en bytes, de los datos de mensaje devueltos por la llamada MQGET en el parámetro *Buffer* . Si el gestor de colas no da soporte a esta prestación, *ReturnedLength* se establece en el valor MQRL_UNDEFINED.

Cuando los mensajes se convierten entre codificaciones o juegos de caracteres, los datos del mensaje a veces pueden cambiar de tamaño. A la devolución de la llamada MQGET:

- Si *ReturnedLength no* es MQRL_UNDEFINED, *ReturnedLength*proporciona el número de bytes de datos de mensaje devueltos.
- Si *ReturnedLength* tiene el valor MQRL_UNDEFINED, el número de bytes de datos de mensaje devueltos normalmente lo proporciona el menor de *BufferLength* y *DataLength*, pero puede ser *menor que* si la llamada MQGET se completa con el código de razón MQRC_TRUNCATED_MSG_ACCEPTED. Si esto sucede, los bytes insignificantes del parámetro *Buffer* se establecen en nulos.

Se define el siguiente valor especial:

MQRL_UNDEFINED
No se ha definido la longitud de los datos devueltos.

En z/OS, el valor devuelto para el campo *ReturnedLength* es siempre MQRL_UNDEFINED.

El valor inicial de este campo es MQRL_UNDEFINED. Este campo se ignora si *Version* es menor que MQGMO_VERSION_3.

Segmentación (MQCHAR)

Es un distintivo que indica si se permite una segmentación adicional para el mensaje recuperado. Tiene uno de los siguientes valores:

MQSEG_INHIBIDO
Segmentación no permitida.

MQSEG_PERMITIDO
Segmentación permitida.

En z/OS, el gestor de colas siempre establece este campo en MQSEG_INHIBIDO.

Se trata de un campo de salida. El valor inicial de este campo es MQSEG_INHIBIDO. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

SegmentStatus (MQCHAR)

Es un distintivo que indica si el mensaje recuperado es un segmento de un mensaje lógico. Tiene uno de los siguientes valores:

MQSS_NO_A_SEGMENTO
El mensaje no es un segmento.

SEGMENTO_MQSS
El mensaje es un segmento, pero no es el último segmento del mensaje lógico.

MQSS_LAST_SEGMENT
El mensaje es el último segmento del mensaje lógico.

También es el valor devuelto si el mensaje lógico consta de un solo segmento.

En z/OS, el gestor de colas siempre establece este campo en MQSS_NOT_A_SEGMENT.

Se trata de un campo de salida. El valor inicial de este campo es MQSS_NOT_A_SEGMENT. Este campo se ignora si *Version* es menor que MQGMO_VERSION_2.

Signal1 (MQLONG)

Es un campo de entrada que sólo se utiliza junto con la opción MQGMO_SET_SIGNAL; identifica una señal que se debe entregar cuando hay un mensaje disponible.

Nota: El tipo de datos y el uso de este campo están determinados por el entorno; por este motivo, las aplicaciones que desee portar entre distintos entornos no deben utilizar señales.

- En z/OS, este campo debe contener la dirección de un bloque de control de sucesos (ECB). La aplicación debe borrar el ECB antes de emitir la llamada MQGET. El almacenamiento que contiene el ECB no debe liberarse hasta que se cierre la cola. El gestor de colas publica el BCE con uno de los códigos de terminación de señal descritos. Estos códigos de terminación se establecen en los bits 2 a 31 del ECB, el área definida en la macro de correlación de z/OS IHAECB para un código de terminación de usuario.
- En todos los demás entornos, se trata de un campo reservado; su valor no es significativo.

Los códigos de terminación de señal son:

MQEC_MSG_LLEGADO
Ha llegado un mensaje adecuado a la cola. Este mensaje no se ha reservado para el llamante; se debe emitir una segunda solicitud MQGET, pero otra aplicación podría recuperar el mensaje antes de realizar la segunda solicitud.

MQEC_WAIT_INTERVAL_CADUCADO
El *WaitInterval* especificado ha caducado sin que llegue un mensaje adecuado.

MQEC_WAIT_CANCELADO
La espera se ha cancelado por una razón indeterminada (como la terminación del gestor de colas o la inhabilitación de la cola). Vuelva a emitir la solicitud si desea realizar un diagnóstico adicional.

MQEC_Q_MGR_QUIESCING
La espera se ha cancelado porque el gestor de colas ha entrado en el estado de desactivación temporal (se ha especificado MQGMO_FAIL_IF_QUIESCING en la llamada MQGET).

MQEC_CONNECTION_QUIESCING
La espera se ha cancelado porque la conexión ha entrado en el estado de desactivación temporal (se ha especificado MQGMO_FAIL_IF_QUIESCING en la llamada MQGET).

El valor inicial de este campo lo determina el entorno:

- En z/OS, el valor inicial es el puntero nulo.
- En todos los demás entornos, el valor inicial es 0.

Signal2 (MQLONG)

Se trata de un campo de entrada que sólo se utiliza junto con la opción MQGMO_SET_SIGNAL. Es un campo reservado; su valor no es significativo.

El valor inicial de este campo es 0.

StrucId (MQCHAR4)

Es el identificador de estructura. El valor debe ser:

MQGMO_STRUC_ID
Identificador de la estructura de opciones de obtención de mensaje.

Para el lenguaje de programación C, también se define la constante MQGMO_STRUC_ID_ARRAY; tiene el mismo valor que MQGMO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQGMO_STRUC_ID.

Versión (MQLONG)
La versión es el número de versión de la estructura.

El valor debe ser uno de los siguientes:

MQGMO_VERSION_1
Estructura de opciones de obtención de mensaje Version-1 .

Esta versión está soportada en todos los entornos.

MQGMO_VERSION_2
Estructura de opciones de obtención de mensaje Version-2 .

Esta versión está soportada en todos los entornos.

MQGMO_VERSION_3
Estructura de opciones de obtención de Version-3 .

Esta versión está soportada en todos los entornos.

MQGMO_VERSION_4
Estructura de opciones de obtención de mensaje Version-4 .

Esta versión está soportada en todos los entornos.

Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQGMO_CURRENT_VERSION
Versión actual de la estructura de opciones de obtención de mensajes.

Siempre es un campo de entrada. El valor inicial de este campo es MQGMO_VERSION_1.

WaitInterval (MQLONG)

Es el tiempo aproximado, expresado en milisegundos, que la llamada MQGET espera a que llegue un mensaje adecuado (es decir, un mensaje que cumpla los criterios de selección especificados en el parámetro *MsgDesc* de la llamada MQGET.

.

Importante: No hay espera, ni retardo, si un mensaje adecuado está disponible inmediatamente.

Consulte el campo *MsgId* descrito en "MQMD - Descriptor de mensaje" en la página 392 para obtener más detalles). Si no ha llegado ningún mensaje adecuado después de que haya transcurrido este tiempo, la llamada se completa con MQCC_FAILED y el código de razón MQRC_NO_MSG_AVAILABLE.

En z/OS, el periodo de tiempo que la llamada MQGET espera realmente se ve afectado por las consideraciones de carga y planificación de trabajo del sistema y puede variar entre el valor especificado para *WaitInterval* y aproximadamente 250 milisegundos mayor que *WaitInterval*.

WaitInterval se utiliza junto con la opción MQGMO_WAIT o MQGMO_SET_SIGNAL. Se ignora si no se especifica ninguno de ellos. Si se especifica uno de estos valores, *WaitInterval* debe ser mayor o igual que cero, o el siguiente valor especial:

MQWI_UNLIMITED
Intervalo de espera ilimitado.

El valor inicial de este campo es 0.

Valores iniciales y declaraciones de lenguaje para MQGMO

Tabla 508. Valores iniciales de campos en MQGMO para MQGMO

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQGMO_STRUC_ID	'GMO¬'
Version	MQGMO_VERSION_1	1
Options	MQGMO_NO_WAIT	0

Tabla 508. Valores iniciales de campos en MQGMO para MQGMO (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
WaitInterval	Ninguna	0
Signal1	Ninguna	Puntero nulo en z/OS; de lo contrario, 0
Signal2	Ninguna	0
ResolvedQName	Ninguna	Serie nula o espacios en blanco
MatchOptions	MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID	3
GroupStatus	MQGS_NOT_IN_GROUP	'¬'
SegmentStatus	MQSS_NO_A_SEGMENTO	'¬'
Segmentation	MQSEG_INHIBIDO	'¬'
Reserved1	Ninguna	'¬'
MsgToken	MQMTOK_NONE	Nulos
ReturnedLength	MQRL_UNDEFINED	-1
Reserved2	Ninguna	'¬'
MsgHandle	MQHM_NONE	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQGMO_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQGMO MyGMO = {MQGMO_DEFAULT};
```

Declaración C

```
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 */
};
```

- En z/OS, el campo *Signal1* se declara como PMQLONG.

declaración COBOL

```
**   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.
```

- En z/OS, el campo *Signal1* se declara como POINTER.

Declaración PL/I

```
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 */
```

- En z/OS, el campo *Signal1* se declara como `pointer`.

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

MQIIH-Cabecera de información de IMS

La tabla siguiente resume los campos de la estructura.

Tabla 509. Campos en MQIIH

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId

Tabla 509. Campos en MQIIH (continuación)		
Campo	**Descripción**	**Tema**
Version	Número de versión de la estructura	Versión
StrucLength	Longitud de la estructura MQIIH	StrucLength
Encoding	Reserved	Codificación
CodedCharSetId	Reserved	CodedCharSetId
Format	Nombre del formato MQ de los datos que siguen a MQIIH	Formato
Flags	Distintivos	Distintivos
LTermOverride	Alteración temporal del terminal lógico	LTermOverride
MFSMapName	Nombre de la correlación de servicios del formato del mensaje	MFSMapName
ReplyToFormat	Nombre del formato MQ del mensaje de respuesta	ReplyToFormat
Authenticator	Contraseña o passticket de RACF™	Authenticator
TranInstanceId	Identificador de instancia de transacción	TranInstanceId
TranState	Estado de la transacción	TranState
CommitMode	Modalidad de confirmación	CommitMode
SecurityScope	Ámbito de seguridad	SecurityScope
Reserved	Reserved	Reserved

Visión general de MQIIH

Disponibilidad: Todos los sistemas WebSphere MQ y clientes WebSphere MQ .

Finalidad: la estructura MQIIH describe la información que debe estar presente al principio de un mensaje enviado al puente IMS a través de WebSphere MQ for z/OS.

Nombre de formato: MQFMT_IMS.

Conjunto de caracteres y codificación: las condiciones especiales se aplican al juego de caracteres y la codificación utilizados para la estructura MQIIH y los datos de mensaje de aplicación:

- Las aplicaciones que se conectan al gestor de colas propietario de la cola puente IMS deben proporcionar una estructura MQIIH que esté en el juego de caracteres y la codificación del gestor de colas. Esto se debe a que la conversión de datos de la estructura MQIIH no se realiza en este caso.
- Las aplicaciones que se conectan a otros gestores de colas pueden proporcionar una estructura MQIIH que esté en cualquiera de los conjuntos de caracteres y codificaciones soportados; el agente de canal de mensajes receptor conectado al gestor de colas propietario de la cola puente IMS convierte la MQIIH.
- Los datos del mensaje de aplicación que siguen a la estructura MQIIH deben estar en el mismo juego de caracteres y codificación que la estructura MQIIH. No utilice los campos *CodedCharSetId* y *Encoding* en la estructura MQIIH para especificar el juego de caracteres y la codificación de los datos del mensaje de aplicación.

  Debe proporcionar una salida de conversión de datos para convertir los datos del mensaje de aplicación si los datos no son uno de los formatos incorporados soportados por el gestor de colas.

Campos para MQIIH

La estructura MQIIH contiene los campos siguientes; los campos se describen en **orden alfabético**:

Autenticador (MQCHAR8)

Esta es la contraseña de RACF o PassTicket. Es opcional; si se especifica, se utiliza con el ID de usuario en el contexto de seguridad MQMD para crear un UTOKEN que se envía a IMS para proporcionar un contexto de seguridad. Si no se especifica, el ID de usuario se utiliza sin verificación. Esto depende del valor de los conmutadores RACF , que pueden requerir la presencia de un autenticador.

Esto se ignora si el primer byte está en blanco o es nulo. Se puede utilizar el siguiente valor especial:

MQIAUT_NONE
Sin autenticación.

Para el lenguaje de programación C, la constante MQIAUT_NONE_ARRAY también está definida; tiene el mismo valor que MQIAUT_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_AUTHENTICATOR_LENGTH. El valor inicial de este campo es MQIAUT_NONE.

CodedCharSetId (MQLONG)

Se trata de un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

El ID de juego de caracteres para las estructuras soportadas que siguen a una estructura MQIIH es el mismo que el de la propia estructura MQIIH y se toma de cualquier cabecera MQ anterior.

CommitMode (MQCHAR)

Esta es la modalidad de confirmación de IMS . Consulte la publicación *OTMA Reference* para obtener más información sobre las modalidades de confirmación de IMS . El valor debe ser uno de los siguientes:

MQICM_COMMIT_THEN_SEND
Confirme y, a continuación, envíe.

Esta modalidad implica una doble cola de salida, pero tiempos de ocupación de región más cortos. Las transacciones de vía de acceso rápida y conversacional no se pueden ejecutar con esta modalidad.

MQICM_SEND_THEN_COMMIT
Enviar y, a continuación, confirmar.

Cualquier transacción IMS iniciada como resultado de un mpde de confirmación de MQICM_SEND_THEN_COMMIT se ejecuta en modalidad RESPONSE independientemente de cómo se haya definido la transacción en la definición del sistema IMS (parámetro MSGTYPE en la macro TRANSACT). Esto también se aplica a las transacciones iniciadas mediante un conmutador de transacción.

El valor inicial de este campo es MQICM_COMMIT_THEN_SEND.

Encoding (MQLONG)

Se trata de un campo reservado; su valor no es significativo. El valor inicial de este campo es 0.

La codificación para las estructuras soportadas que siguen a una estructura MQIIH es la misma que la de la propia estructura MQIIH y se toma de cualquier cabecera MQ anterior.

Flags (MQLONG)

El valor de distintivos debe ser:

MQIIH_NONE
Sin distintivos.

MQIIH_PASS_EXPIRATION
El mensaje de respuesta contiene:

- Las mismas opciones de informe de caducidad que el mensaje de solicitud

- El tiempo de caducidad restante del mensaje de solicitud sin realizar ningún ajuste para el tiempo de proceso del puente

Si no se establece este valor, la hora de caducidad se establece en *unlimited*.

MQIIH_REPLY_FORMAT_NONE
Establece MQIIH.Format de la respuesta a MQFMT_NONE.

MQIIH_IGNORE_PURG
Establece el indicador TMAMIPRG en el prefijo OTMA, que solicita que OTMA ignore las llamadas PURG en el TP PCB para las transacciones CM0 .

MQIIH_CM0_REQUEST_RESPONSE
Para transacciones de modalidad de confirmación 0 (CM0), este distintivo establece el indicador TMAMHRSP en el prefijo OTMA. El establecimiento de este indicador solicita que OTMA/IMS genere un mensaje DFS2082 RESPONSE MODE TRANSACTION TERMINADO SIN RESPUESTA cuando el programa de aplicación IMS original no responde al IOPCB ni conmuta el mensaje a otra transacción.

El valor inicial de este campo es MQIIH_NONE.

Format (MQCHAR8)

Especifica el nombre de formato MQ de los datos que siguen a la estructura MQIIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

La longitud de este campo la proporciona MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

LTermOverride (MQCHAR8)

La alteración temporal de terminal lógico, situada en el campo PCB de E/S. Es opcional; si no se especifica, se utiliza el nombre TPIPE. Se ignora si el primer byte está en blanco o es nulo.

La longitud de este campo la proporciona MQ_LTERM_OVERRIDE_LENGTH. El valor inicial de este campo es de 8 caracteres en blanco.

MFSMapName (MQCHAR8)

El nombre de correlación de servicios de formato de mensaje, situado en el campo PCB de E/S. Es opcional. En la entrada representa el MID, en la salida representa el MOD. Se ignora si el primer byte está en blanco o es nulo.

La longitud de este campo la proporciona MQ_MFS_MAP_NAME_LENGTH. El valor inicial de este campo es de 8 caracteres en blanco.

Formato ReplyTo(MQCHAR8)

Es el nombre de formato MQ del mensaje de respuesta que se envía en respuesta al mensaje actual. La longitud de este campo la proporciona MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

Para convertir los datos en el mensaje de respuesta utilizando MQGMO_CONVERT, especifique MQIIH.replyToFormat= MQFMT_STRING o MQIIH.replyToFormat= MQFMT_IMS_VAR_STRING. Para obtener una explicación del uso de estos campos, consulte “Format (MQCHAR8)” en la página 408.

Si se utiliza el valor predeterminado (MQIIH.replyToFormat= MQFMT_NONE) en el mensaje de solicitud y el mensaje de respuesta se recupera utilizando MQGMO_CONVERT, no se realiza ninguna conversión de datos.

Reservado (MQCHAR)

Es un campo reservado; debe estar en blanco.

SecurityScope (MQCHAR)

Esto indica el proceso de seguridad de IMS necesario. Los valores siguientes están definidos:

MQISS_CHECK
Comprobar ámbito de seguridad: se crea un ACEE en la región de control, pero no en la región dependiente.

MQISS_FULL
Ámbito de seguridad completo: un ACEE almacenado en memoria caché se crea en la región de control y un ACEE no almacenado en memoria caché se crea en la región dependiente. Si utiliza MQISS_FULL, asegúrese de que el ID de usuario para el que se crea el ACEE tenga acceso a los recursos utilizados en la región dependiente.

Si no se especifica MQISS_CHECK ni MQISS_FULL para este campo, se presupone MQISS_CHECK.

El valor inicial de este campo es MQISS_CHECK.

StrucId (MQCHAR4)

Es el identificador de estructura. El valor debe ser:

MQIIH_STRUC_ID
Identificador de la estructura de cabecera de información de IMS .

Para el lenguaje de programación C, también se define la constante MQIIH_STRUC_ID_ARRAY; tiene el mismo valor que MQIIH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQIIH_STRUC_ID.

StrucLength (MQLONG)

Es la longitud de la estructura MQIIH. El valor debe ser:

MQIIH_LENGTH_1
Longitud de la estructura de cabecera de información de IMS .

El valor inicial de este campo es MQIIH_LENGTH_1.

TranInstanceId (MQBYTE16)

Es el identificador de instancia de transacción. Este campo lo utilizan los mensajes de salida de IMS, por lo que se ignora en la primera entrada. Si establece *TranState* en MQITS_IN_CONVERSATION, debe proporcionarse en la siguiente entrada, y en todas las entradas posteriores, para permitir que IMS correlacione los mensajes con la conversación correcta. Puede utilizar el siguiente valor especial:

MQITII_NONE
No hay ningún identificador de instancia de transacción.

Para el lenguaje de programación C, también se define la constante MQITII_NONE_ARRAY; tiene el mismo valor que MQITII_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_TRAN_INSTANCE_ID_LENGTH. El valor inicial de este campo es MQITII_NONE.

TranState (MQCHAR)

Indica el estado de conversación de IMS . Esto se ignora en la primera entrada porque no existe ninguna conversación. En entradas posteriores, indica si una conversación está activa o no. En la salida, se establece mediante IMS. El valor debe ser uno de los siguientes:

MQITS_EN_CONVERSACIÓN
En conversación.

MQITS_NO_EN_CONVERSACIÓN
No en conversación.

MQITS_ARCHITECTED
Devolver datos de estado de transacción en formato de arquitectura.

Este valor sólo se utiliza con el mandato IMS `/DISPLAY TRAN` . Devuelve los datos de estado de transacción en el formato de arquitectura IMS en lugar del formato de caracteres.

El valor inicial de este campo es MQITS_NOT_IN_CONVERSATION.

Versión (MQLONG)

Es el número de versión de la estructura. El valor debe ser:

MQIIH_VERSION_1
Número de versión para la estructura de cabecera de información de IMS .

La constante siguiente especifica el número de versión de la versión actual:

MQIIH_CURRENT_VERSION
Versión actual de la estructura de cabecera de información de IMS .

El valor inicial de este campo es MQIIH_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQIIH

Tabla 510. Valores iniciales de los campos en MQIIH para MQIIH

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQIIH_STRUC_ID	'IIH?'
Version	MQIIH_VERSION_1	1
StrucLength	MQIIH_LENGTH_1	84
Encoding	Ninguna	0
CodedCharSetId	Ninguna	0
Format	MQFMT_NONE	Espacios en blanco
Flags	MQIIH_NONE	0
LTermOverride	Ninguna	Espacios en blanco
MFSMapName	Ninguna	Espacios en blanco
ReplyToFormat	MQFMT_NONE	Espacios en blanco
Authenticator	MQIAUT_NONE	Espacios en blanco
TranInstanceId	MQITII_NONE	Nulos
TranState	MQITS_NO_EN_CONVERSACIÓN	'?'
CommitMode	MQICM_COMMIT_THEN_SEND	'0'
SecurityScope	MQISS_CHECK	'C'
Reserved	Ninguna	'?'

Notas:

1. ¿El símbolo? representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQIIH_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQIIH MyIIH = {MQIIH_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

MQIMPO-Consultar opciones de propiedad de mensaje

La tabla siguiente resume los campos de la estructura. Estructura MQIMPO de -consultar opciones de propiedad de mensaje

Tabla 511. Campos en MQIMPO

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Options	Opciones que controlan la acción de MQINQMP	Opciones
RequestedEncoding	Codificación en la que se va a convertir la propiedad consultado	RequestedEncoding
RequestedCCSID	Juego de caracteres de la propiedad consultado	RequestedCCSID

Tabla 511. Campos en MQIMPO (continuación)		
Campo	**Descripción**	**Tema**
ReturnedEncoding	Codificación del valor devuelto	ReturnedEncoding
ReturnedCCSID	Juego de caracteres de valor devuelto	ReturnedCCSID
Reserved1	Reservado, campo	ReturnedCCSID
ReturnedName	Nombre de la propiedad consultado	ReturnedName
TypeString	Representación de serie del tipo de datos de la propiedad	TypeString

Visión general de MQIMPO

La estructura de opciones de consulta de propiedades de mensaje.

Disponibilidad: Todos los sistemas WebSphere MQ y clientes WebSphere MQ .

Finalidad: la estructura MQIMPO permite a las aplicaciones especificar opciones que controlan cómo se consultan las propiedades de los mensajes. La estructura es un parámetro de entrada en la llamada MQINQMP.

Juego de caracteres y codificación: Los datos de MQIMPO deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (MQENC_NATIVE).

Campos para MQIMPO

Consultar estructura de opciones de propiedad de mensaje-campos

La estructura MQIMPO contiene los campos siguientes; los campos se describen en **orden alfabético**:

Opciones (MQLONG)
Estructura de opciones de propiedad de mensaje de consulta-Campo Opciones

Las opciones siguientes controlan la acción de MQINQMP. Puede especificar una o más de estas opciones y, si necesita más de una, los valores pueden ser:

- Sumado (no añadir la misma constante más de una vez), o
- Combinado utilizando la operación OR bit a bit (si el lenguaje de programación da soporte a operaciones de bits).

Las combinaciones de opciones que no son válidas se anotan; todas las demás combinaciones son válidas.

Opciones de datos de valor: Las opciones siguientes están relacionadas con el proceso de los datos de valor cuando se recupera la propiedad del mensaje.

MQIM_CONVERT_VALUE
Esta opción solicita que el valor de la propiedad se convierta para que se ajuste a los valores *RequestedCCSID* y *RequestedEncoding* especificados antes de que la llamada MQINQMP devuelva el valor de la propiedad en el área *Value* .

- Si la conversión se realiza correctamente, los campos *ReturnedCCSID* y *ReturnedEncoding* se establecen en los mismos que *RequestedCCSID* y *RequestedEncoding* al volver de la llamada MQINQMP.
- Si la conversión falla, pero la llamada MQINQMP de lo contrario se completa sin error, el valor de propiedad se devuelve sin convertir.

  Si la propiedad es una serie, los campos *ReturnedCCSID* y *ReturnedEncoding* se establecen en el juego de caracteres y la codificación de la serie no convertida.

El código de terminación es MQCC_WARNING en este caso, con el código de razón MQRC_PROP_VALUE_NOT_CONVERTED. El cursor de propiedad se ha avanzado a la propiedad devuelta.

Si el valor de propiedad se expande durante la conversión y supera el tamaño del parámetro *Value* , el valor se devuelve sin convertir, con el código de terminación MQCC_FAILED; el código de razón se establece en MQRC_PROPERTY_VALUE_TOO_BIG.

El parámetro *DataLength* de la llamada MQINQMP devuelve la longitud a la que se habría convertido el valor de propiedad, para permitir que la aplicación determine el tamaño del almacenamiento intermedio necesario para acomodar el valor de propiedad convertido. El cursor de propiedad no se ha modificado.

Esta opción también solicita que:

- Si el nombre de propiedad contiene un comodín, y
- El campo *ReturnedName* se inicializa con una dirección o desplazamiento para el nombre devuelto,

entonces el nombre devuelto se convierte para ajustarse a los valores *RequestedCCSID* y *RequestedEncoding* .

- Si la conversión es satisfactoria, el campo *VSCCSID* de *ReturnedName* y la codificación del nombre devuelto se establecen en el valor de entrada de *RequestedCCSID* y *RequestedEncoding* .
- Si la conversión falla, pero de lo contrario la llamada MQINQMP se completa sin error ni aviso, el nombre devuelto no se convierte. El código de terminación es MQCC_WARNING en este caso, con el código de razón MQRC_PROP_NAME_NOT_CONVERT.

  El cursor de propiedad se ha avanzado a la propiedad devuelta. Se devuelve MQRC_PROP_VALUE_NOT_CONVERTED si no se convierten tanto el valor como el nombre.

Si el nombre devuelto se expande durante la conversión y supera el tamaño del campo *VSBufsize* de *RequestedName*, la serie devuelta se deja sin convertir, con el código de terminación MQCC_FAILED y el código de razón se establece en MQRC_PROPERTY_NAME_TOO_BIG.

El campo *VSLength* de la estructura MQCHARV devuelve la longitud a la que se habría convertido el valor de propiedad, para permitir que la aplicación determine el tamaño del almacenamiento intermedio necesario para acomodar el valor de propiedad convertido. El cursor de propiedad no se ha modificado.

MQIM_CONVERT_TYPE

Esta opción solicita que el valor de la propiedad se convierta de su tipo de datos actual en el tipo de datos especificado en el parámetro *Type* de la llamada MQINQMP.

- Si la conversión es satisfactoria, el parámetro *Type* no se modifica al devolver la llamada MQINQMP.
- Si la conversión falla, pero de lo contrario la llamada MQINQMP se completa sin errores, la llamada falla con la razón MQRC_PROP_CONV_NOT_SUPPORTED. El cursor de propiedad no se ha modificado.

Si la conversión del tipo de datos hace que el valor se expanda durante la conversión y el valor convertido supera el tamaño del parámetro *Value* , el valor se devuelve sin convertir, con el código de terminación MQCC_FAILED y el código de razón se establece en MQRC_PROPERTY_VALUE_TOO_BIG.

El parámetro *DataLength* de la llamada MQINQMP devuelve la longitud a la que se habría convertido el valor de propiedad, para permitir que la aplicación determine el tamaño del almacenamiento intermedio necesario para acomodar el valor de propiedad convertido. El cursor de propiedad no se ha modificado.

Si el valor del parámetro *Type* de la llamada MQINQMP no es válido, la llamada falla con la razón MQRC_PROPERTY_TYPE_ERROR.

Si la conversión de tipo de datos solicitada no está soportada, la llamada falla con la razón MQRC_PROP_CONV_NOT_SUPPORTED. Se da soporte a las siguientes conversiones de tipos de datos:

Tipo de datos de propiedad	Tipos de datos de destino soportados
MQTYPE_BOOLEAN	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	Ninguna

Las normas generales que rigen las conversiones soportadas son las siguientes:

- Los valores de propiedad numéricos se pueden convertir de un tipo de datos a otro, siempre que no se pierdan datos durante la conversión.

  Por ejemplo, el valor de una propiedad con el tipo de datos MQTYPE_INT32 se puede convertir en un valor con el tipo de datos MQTYPE_INT64, pero no se puede convertir en un valor con el tipo de datos MQTYPE_INT16.
- Un valor de propiedad de cualquier tipo de datos se puede convertir a una serie.
- Un valor de propiedad de serie se puede convertir a cualquier otro tipo de datos, siempre que la serie tenga el formato correcto para la conversión. Si una aplicación intenta convertir un valor de propiedad de serie que no tiene el formato correcto, WebSphere MQ devuelve el código de razón MQRC_PROP_NUMBER_FORMAT_ERROR.
- Si una aplicación intenta una conversión que no está soportada, WebSphere MQ devuelve el código de razón MQRC_PROP_CONV_NOT_SUPPORTED.

Las reglas específicas para convertir un valor de propiedad de un tipo de datos a otro son las siguientes:

- Al convertir un valor de propiedad MQTYPE_BOOLEAN en una serie, el valor TRUE se convierte a la serie "TRUE" y el valor false se convierte a la serie "FALSE".
- Al convertir un valor de propiedad MQTYPE_BOOLEAN en un tipo de datos numérico, el valor TRUE se convierte en uno y el valor FALSE se convierte en cero.
- Al convertir un valor de propiedad de serie a un valor MQTYPE_BOOLEAN, la serie "TRUE", o "1", se convierte a TRUE, y la serie "FALSE", o "0", se convierte a FALSE.

  Tenga en cuenta que los términos "TRUE" y "FALSE" no distinguen entre mayúsculas y minúsculas.

  Cualquier otra serie no se puede convertir; WebSphere MQ devuelve el código de razón MQRC_PROP_NUMBER_FORMAT_ERROR.
- Al convertir un valor de propiedad de serie a un valor con el tipo de datos MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32 o MQTYPE_INT64, la serie debe tener el formato siguiente:

  ```
  [blanks][sign]digits
  ```

  El significado de los componentes de la serie es el siguiente:

blanks
Caracteres en blanco iniciales opcionales

sign
Un carácter opcional de signo más (+) o signo menos (-).

digits
Una secuencia continua de caracteres de dígitos (0-9). Al menos, debe estar presente un carácter de dígito.

Después de la secuencia de caracteres de dígitos, la serie puede contener otros caracteres que no son caracteres de dígitos, pero la conversión se detiene tan pronto como se llega al primero de estos caracteres. Se da por sentado que la serie representa un entero decimal.

WebSphere MQ devuelve el código de razón MQRC_PROP_NUMBER_FORMAT_ERROR si la serie no tiene el formato correcto.

- Al convertir un valor de propiedad de serie a un valor con el tipo de datos MQTYPE_FLOAT32 o MQTYPE_FLOAT64, la serie debe tener el formato siguiente:

```
[blanks][sign]digits[.digits][e_char[e_sign]e_digits]
```

El significado de los componentes de la serie es el siguiente:

blanks
Caracteres en blanco iniciales opcionales

sign
Un carácter opcional de signo más (+) o signo menos (-).

digits
Una secuencia continua de caracteres de dígitos (0-9). Al menos, debe estar presente un carácter de dígito.

e_char
Un carácter exponente, que es "E" o "e".

e_sign
Un carácter de signo más (+) o signo menos (-) opcional para el exponente.

e_digits
Una secuencia contigua de caracteres de dígitos (0-9) para el exponente. Al menos, debe estar presente un carácter de dígito, si la serie contiene un carácter de exponente.

Detrás de la secuencia de caracteres de dígitos, o los caracteres opcionales que representan un exponente, la serie puede contener otros caracteres que no son caracteres de dígitos, pero la conversión se detiene tan pronto como se llega al primero de estos caracteres. Se da por supuesto que la serie representa un número de coma flotante decimal con un exponente que es una potencia de 10.

WebSphere MQ devuelve el código de razón MQRC_PROP_NUMBER_FORMAT_ERROR si la serie no tiene el formato correcto.

- Al convertir un valor de propiedad numérico en una serie, el valor se convierte a la representación de serie del valor como un número decimal, no la serie que contiene el carácter ASCII para ese valor. Por ejemplo, el entero 65 se convierte a la serie "65", no a la serie "A".
- Al convertir un valor de propiedad de serie de bytes en una serie, cada byte se convierte en los dos caracteres hexadecimales que representan el byte. Por ejemplo, la matriz de bytes {0xF1, 0x12, 0x00, 0xFF} se convierte a la serie "F11200FF".

MQIM_QUERY_LENGTH
Consulte el tipo y la longitud del valor de propiedad. La longitud se devuelve en el parámetro *DataLength* de la llamada MQINQMP. El valor de propiedad no se devuelve.

Si se especifica un almacenamiento intermedio *ReturnedName* , el campo *VSLength* de la estructura MQCHARV se rellena con la longitud del nombre de propiedad. El nombre de propiedad no se devuelve.

Opciones de iteración: Las opciones siguientes están relacionadas con la iteración sobre propiedades, utilizando un nombre con un carácter comodín

MQIM_INQ_PRIMERO
Consultar la primera propiedad que coincide con el nombre especificado. Después de esta llamada, se establece un cursor en la propiedad que se devuelve.

Este es el valor predeterminado.

La opción MQIMPO_INQ_PROP_UNDER_CURSOR se puede utilizar posteriormente con una llamada MQINQMP, si es necesario, para volver a consultar la misma propiedad.

Tenga en cuenta que sólo hay un cursor de propiedad; por lo tanto, si el nombre de propiedad, especificado en la llamada MQINQMP, cambia el cursor se restablece.

Esta opción no es válida con ninguna de las opciones siguientes:

MQIM_INQ_NEXT
MQIM_INQ_PROP_UNDER_CURSOR

MQIM_INQ_NEXT
Consulta la siguiente propiedad que coincide con el nombre especificado, continuando la búsqueda desde el cursor de propiedad. El cursor está avanzado a la propiedad que se devuelve.

Si esta es la primera llamada MQINQMP para el nombre especificado, se devuelve la primera propiedad que coincide con el nombre especificado.

La opción MQIMPO_INQ_PROP_UNDER_CURSOR se puede utilizar posteriormente con una llamada MQINQMP si es necesario, para volver a consultar la misma propiedad.

Si la propiedad bajo el cursor se ha suprimido, MQINQMP devuelve la siguiente propiedad coincidente después de la que se ha suprimido.

Si se añade una propiedad que coincide con el comodín, mientras una iteración está en curso, es posible que la propiedad se devuelva o no durante la finalización de la iteración. La propiedad se devuelve una vez que la iteración se reinicia utilizando MQIMPO_INQ_FIRST.

Una propiedad que coincide con el comodín que se ha suprimido, mientras la iteración estaba en curso, no se devuelve después de su supresión.

Esta opción no es válida con ninguna de las opciones siguientes:

MQIM_INQ_PRIMERO
MQIM_INQ_PROP_UNDER_CURSOR

MQIM_INQ_PROP_UNDER_CURSOR
Recupere el valor de la propiedad a la que apunta el cursor de propiedad. La propiedad a la que apunta el cursor de propiedad es la que se ha consultado por última vez, utilizando la opción MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje, cuando se especifica el descriptor de mensaje en el campo *MsgHandle* del MQGMO en una llamada MQGET, o cuando se especifica el descriptor de mensaje en los campos *OriginalMsgHandle* o *NewMsgHandle* de la estructura MQPMO en una llamada MQPUT.

Si esta opción se utiliza cuando el cursor de propiedad todavía no se ha establecido, o si la propiedad a la que apunta el cursor de propiedad se ha suprimido, la llamada falla con el código de terminación MQCC_FAILED y la razón MQRC_PROPERTY_NOT_AVAILABLE.

Esta opción no es válida con ninguna de las opciones siguientes:

MQIM_INQ_PRIMERO
MQIM_INQ_NEXT

Si no se requiere ninguna de las opciones descritas anteriormente, se puede utilizar la opción siguiente:

MQIMPO_NONE
Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

MQIMPO_NONE ayuda a la documentación del programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Siempre es un campo de entrada. El valor inicial de este campo es MQIMPO_INQ_FIRST.

RequestedCCSID (MQLONG)
Consultar estructura de opciones de propiedad de mensaje-Campo RequestedCCSID

El juego de caracteres en el que se va a convertir el valor de propiedad consultado si el valor es una serie de caracteres. También es el juego de caracteres en el que se convertirá *ReturnedName* cuando se especifique MQIMPO_CONVERT_VALUE o MQIMPO_CONVERT_TYPE.

El valor inicial de este campo es MQCCSI_APPL.

RequestedEncoding (MQLONG)
Consultar estructura de opciones de propiedad de mensaje-Campo RequestedEncoding

Esta es la codificación en la que se convertirá el valor de propiedad consultado cuando se especifique MQIMPO_CONVERT_VALUE o MQIMPO_CONVERT_TYPE.

El valor inicial de este campo es MQENC_NATIVE.

Reserved1 (MQCHAR)

Este es un campo reservado. El valor inicial de este campo es un carácter en blanco (campo de 4 bytes).

ReturnedCCSID (MQLONG)
Consultar estructura de opciones de propiedad de mensaje-Campo ReturnedCCSID

En la salida, es el juego de caracteres del valor devuelto si el parámetro *Type* de la llamada MQINQMP es MQTYPE_STRING.

Si se especifica la opción MQIMPO_CONVERT_VALUE y la conversión se ha realizado correctamente, el campo *ReturnedCCSID* , en el momento de la devolución, es el mismo valor que el valor que se ha pasado.

El valor inicial de este campo es cero.

ReturnedEncoding (MQLONG)
Estructura de opciones de propiedad de mensaje de consulta-Campo ReturnedEncoding

En la salida, es la codificación del valor devuelto.

Si se especifica la opción MQIMPO_CONVERT_VALUE y la conversión se ha realizado correctamente, el campo *ReturnedEncoding* , en el momento de la devolución, es el mismo valor que el valor que se ha pasado.

El valor inicial de este campo es MQENC_NATIVE.

ReturnedName (MQCHARV)
Consultar estructura de opciones de propiedad de mensaje-Campo ReturnedName

El nombre real de la propiedad consultado.

En la entrada, se puede pasar un almacenamiento intermedio de serie utilizando el campo *VSPtr* o *VSOffset* de la estructura MQCHARV . La longitud del almacenamiento intermedio de serie se especifica utilizando el campo *VSBufsize* de la estructura MQCHARV.

Al volver de la llamada MQINQMP, el almacenamiento intermedio de serie se completa con el nombre de la propiedad que se ha consultado, siempre que el almacenamiento intermedio de serie sea lo suficientemente largo como para contener completamente el nombre. El campo *VSLength* de la estructura MQCHARV se rellena con la longitud del nombre de propiedad. El campo *VSCCSID* de la estructura MQCHARV se rellena para indicar el juego de caracteres del nombre devuelto, si la conversión del nombre ha fallado o no.

Es un campo de entrada/salida. El valor inicial de este campo es MQCHARV_DEFAULT.

StrucId (MQCHAR4)
Estructura de opciones de propiedad de mensaje de consulta-Campo StrucId

Es el identificador de estructura. El valor debe ser:

MQIM_ID_ESTRUCTURA
Identificador de la estructura de opciones de propiedad de mensaje de consulta.

Para el lenguaje de programación C, también se define la constante MQIMPO_STRUC_ID_ARRAY; tiene el mismo valor que MQIMPO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQIMPO_STRUC_ID.

TypeString (MQCHAR8)
Consultar estructura de opciones de propiedad de mensaje-Campo TypeString

Una representación de serie del tipo de datos de la propiedad.

Si la propiedad se ha especificado en una cabecera MQRFH2 y no se reconoce el atributo MQRFH2 `dt` , este campo se puede utilizar para determinar el tipo de datos de la propiedad. *TypeString* se devuelve en el juego de caracteres codificado 1208 (UTF-8), y son los primeros ocho bytes del valor del atributo `dt` de la propiedad que no se ha podido reconocer

Siempre es un campo de salida. El valor inicial de este campo es la serie nula en el lenguaje de programación C y 8 caracteres en blanco en otros lenguajes de programación.

Versión (MQLONG)
Consultar estructura de opciones de propiedad de mensaje-Campo Versión

Es el número de versión de la estructura. El valor debe ser:

MQIMPO_VERSION_1
Número de versión para la estructura de opciones de propiedad de mensaje de consulta.

La constante siguiente especifica el número de versión de la versión actual:

MQIM_VERSIÓN_ACTUAL
Versión actual de la estructura de opciones de propiedad de mensaje de consulta.

Siempre es un campo de entrada. El valor inicial de este campo es MQIMPO_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQIMPO
Estructura de opciones de propiedad de mensaje de consulta-Valores iniciales

Tabla 512. Valores iniciales de campos en MQIPMO

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQIM_ID_ESTRUCTURA	'IMPO'
Version	MQIMPO_VERSION_1	1

Tabla 512. Valores iniciales de campos en MQIPMO (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
Options	MQIM_INQ_PRIMERO	
RequestedEncoding	MQENC_NATIVE	
RequestedCCSID	MQCCSI_APPL	
ReturnedEncoding	MQENC_NATIVE	
ReturnedCCSID	0	
Reserved1	0	
ReturnedName	MQCHARV_PREDETERMINADO	
TypeString	Serie nula o espacios en blanco	

Notas:

1. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
2. En el lenguaje de programación C, la variable de macroMQIMPO_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQIMPO MyIMPO = {MQIMPO_DEFAULT};
```

Declaración C
Consultar estructura de opciones de propiedad de mensaje-Declaración de lenguaje C

```
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 */
};
```

declaración COBOL
Estructura de opciones de propiedad de mensaje de consulta-Declaración de lenguaje COBOL

```
**   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.
```

Declaración PL/I
Consultar estructura de opciones de propiedad de mensaje-Declaración de lenguaje PL/I

```
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 */
```

Declaración High Level Assembler
Estructura de opciones de propiedad de mensaje de consulta-Declaración de lenguaje ensamblador

```
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)
```

MQMD - Descriptor de mensaje

La tabla siguiente resume los campos de la estructura.

Tabla 513. Campos en MQMD		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Report	Opciones para mensajes de informe	Informe
MsgType	Tipo de mensaje	MsgType
Expiry	Duración del mensaje	MQMD-Campo de caducidad
Feedback	Código de comentario o razón	MQMD-Campo de comentarios
Encoding	Codificación numérica de datos de mensaje	Codificación
CodedCharSetId	Identificador de juego de caracteres de datos de mensaje	CodedCharSetId
Format	Nombre de formato de datos de mensaje	Formato
Priority	Prioridad de mensaje	Priority
Persistence	Persistencia de los mensajes	Persistencia
MsgId	Identificador del mensaje	MQMD-Campo MsgId
CorrelId	Identificador de correlación	CorrelId
BackoutCount	Contador de restitución	BackoutCount
ReplyToQ	Nombre de la cola de respuestas	ReplyToQ
ReplyToQMgr	Nombre del gestor de colas de respuestas	GestorColasRespuesta
UserIdentifier	Identificador de usuario	UserIdentifier
AccountingToken	Señal de contabilidad	AccountingToken
ApplIdentityData	Datos de aplicación relacionados con la identidad	ApplIdentityData
PutApplType	Tipo de aplicación que coloca el mensaje	PutApplType
PutApplName	Nombre de la aplicación que ha transferido el mensaje	PutApplName
PutDate	Fecha cuando se colocó el mensaje	PutDate
PutTime	Hora cuando se colocó el mensaje	PutTime
ApplOriginData	Datos de aplicación relacionados con el origen	ApplOriginData
Nota: Los campos restantes se ignoran si *Version* es menor que MQMD_VERSION_2.		
GroupId	Identificador de grupo	GroupId
MsgSeqNumber	Número de secuencia del mensaje lógico en el grupo	MsgSeqNumber
Offset	Desplazamiento de datos en el mensaje físico desde el inicio del mensaje lógico	desplazamiento
MsgFlags	Distintivos de mensajes	MQMD - Campo MsgFlags

Tabla 513. Campos en MQMD (continuación)		
Campo	**Descripción**	**Tema**
OriginalLength	Longitud del mensaje original	OriginalLength

Visión general de MQMD

Disponibilidad: todos los sistemas WebSphere MQ , además de los clientes MQI de WebSphere MQ conectados a estos sistemas.

Finalidad: la estructura MQMD contiene la información de control que acompaña a los datos de aplicación cuando un mensaje viaja entre las aplicaciones de envío y recepción. La estructura es un parámetro de entrada/salida en las llamadas MQGET, MQPUT y MQPUT1 .

Versión: La versión actual de MQMD es MQMD_VERSION_2. Las aplicaciones que están pensadas para ser portables entre varios entornos deben asegurarse de que la versión necesaria de MQMD esté soportada en todos los entornos afectados. Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones siguientes.

Los archivos de cabecera, COPY e INCLUDE proporcionados para los lenguajes de programación soportados contienen la versión más reciente de MQMD soportada por el entorno, pero con el valor inicial del campo *Version* establecido en MQMD_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1 , la aplicación debe establecer el campo *Version* en el número de versión de la versión necesaria.

Hay disponible una declaración para la estructura version-1 con el nombre MQMD1.

Juego de caracteres y codificación: los datos de MQMD deben estar en el juego de caracteres y la codificación del gestor de colas local; los proporciona el atributo de gestor de colas *CodedCharSetId* y MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de WebSphere MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Si los gestores de colas emisores y receptores utilizan conjuntos de caracteres o codificaciones diferentes, los datos de MQMD se convierten automáticamente. No es necesario que la aplicación convierta MQMD.

Utilización de distintas versiones de MQMD: un MQMD de version-2 es equivalente a utilizar un MQMD de version-1 y a prefijar los datos del mensaje con una estructura MQMDE. Sin embargo, si todos los campos de la estructura MQMDE tienen sus valores predeterminados, se puede omitir MQMDE. Un MQMD version-1 más MQMDE se utilizan tal como se describe:

- En las llamadas MQPUT y MQPUT1 , si la aplicación proporciona un MQMD de version-1 , la aplicación puede opcionalmente añadir un prefijo a los datos de mensaje con un MQMDE, estableciendo el campo *Format* de MQMD en MQFMT_MD_EXTENSION para indicar que hay un MQMDE presente. Si la aplicación no proporciona un MQMDE, el gestor de colas asume los valores predeterminados para los campos de MQMDE.

  Nota: Varios de los campos que existen en el MQMD version-2 pero no en el MQMD version-1 son campos de entrada/salida en las llamadas MQPUT y MQPUT1 . Sin embargo, el gestor de colas *no* devuelve ningún valor en los campos equivalentes de MQMDE en la salida de las llamadas MQPUT y MQPUT1 ; si la aplicación requiere estos valores de salida, debe utilizar un MQMD version-2 .
- En la llamada MQGET, si la aplicación proporciona un MQMD version-1 , el gestor de colas prefija el mensaje devuelto con un MQMDE, pero sólo si uno o varios de los campos de MQMDE tienen un valor no predeterminado. El campo *Format* de MQMD tendrá el valor MQFMT_MD_EXTENSION para indicar que hay un MQMDE presente.

Los valores predeterminados que el gestor de colas utiliza para los campos de MQMDE son los mismos que los valores iniciales de dichos campos, que se muestran en la Tabla 518 en la página 449.

Cuando un mensaje está en una cola de transmisión, algunos de los campos de MQMD se establecen en valores concretos; consulte “MQXQH-Cabecera de cola de transmisión” en la página 596 para obtener más detalles.

Contexto de mensaje: determinados campos de MQMD contienen el contexto de mensaje. Hay dos tipos de contexto de mensaje: *contexto de identidad* y *contexto de origen*. Por norma, se utiliza para:

- El contexto de identidad está relacionado con la aplicación que *originalmente* colocó el mensaje
- El contexto de origen está relacionado con la aplicación que *más recientemente* ha colocado el mensaje.

Estas dos aplicaciones pueden ser la misma aplicación, pero también pueden ser aplicaciones diferentes (por ejemplo, cuando se reenvía un mensaje de una aplicación a otra).

Aunque la identidad y el contexto de origen normalmente tienen los significados descritos, el contenido de ambos tipos de campos de contexto en MQMD depende de las opciones MQPMO_ * _CONTEXT que se especifican cuando se coloca el mensaje. Como resultado, el contexto de identidad no está necesariamente relacionado con la aplicación que colocó originalmente el mensaje, y el contexto de origen no está necesariamente relacionado con la aplicación que colocó el mensaje más recientemente; depende del diseño de la suite de aplicaciones.

El agente de canal de mensajes (MCA) nunca altera el contexto del mensaje. Los MCA que reciben mensajes de gestores de colas remotos utilizan la opción de contexto MQPMO_SET_ALL_CONTEXT en la llamada MQPUT o MQPUT1 . Esto permite al MCA receptor conservar exactamente el contexto de mensaje que ha viajado con el mensaje del MCA emisor. Sin embargo, el resultado es que el contexto de origen no está relacionado con ninguno de los MCA que han enviado y recibido el mensaje. El contexto de origen hace referencia a una aplicación anterior que ha colocado el mensaje. Si todas las aplicaciones intermedias han pasado el contexto de mensaje, el contexto de origen hace referencia a la propia aplicación de origen.

En las descripciones, los campos de contexto se describen como si se utilizaran como se ha descrito anteriormente. Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Campos para MQMD

La estructura MQMD contiene los campos siguientes; los campos se describen en **orden alfabético**:

Tabla 514. Campos en MQMD

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Report	Opciones para mensajes de informe	Informe
MsgType	Tipo de mensaje	MsgType
Expiry	Duración del mensaje	MQMD-Campo de caducidad
Feedback	Código de comentario o razón	MQMD-Campo de comentarios
Encoding	Codificación numérica de datos de mensaje	Codificación
CodedCharSetId	Identificador de juego de caracteres de datos de mensaje	CodedCharSetId
Format	Nombre de formato de datos de mensaje	Formato
Priority	Prioridad de mensaje	Priority
Persistence	Persistencia de los mensajes	Persistencia
MsgId	Identificador del mensaje	MQMD-Campo MsgId
CorrelId	Identificador de correlación	CorrelId

Tabla 514. Campos en MQMD (continuación)		
Campo	**Descripción**	**Tema**
BackoutCount	Contador de restitución	BackoutCount
ReplyToQ	Nombre de la cola de respuestas	ReplyToQ
ReplyToQMgr	Nombre del gestor de colas de respuestas	GestorColasRespuesta
UserIdentifier	Identificador de usuario	UserIdentifier
AccountingToken	Señal de contabilidad	AccountingToken
ApplIdentityData	Datos de aplicación relacionados con la identidad	ApplIdentityData
PutApplType	Tipo de aplicación que coloca el mensaje	PutApplType
PutApplName	Nombre de la aplicación que ha transferido el mensaje	PutApplName
PutDate	Fecha cuando se colocó el mensaje	PutDate
PutTime	Hora cuando se colocó el mensaje	PutTime
ApplOriginData	Datos de aplicación relacionados con el origen	ApplOriginData
Nota: Los campos restantes se ignoran si *Version* es menor que MQMD_VERSION_2.		
GroupId	Identificador de grupo	GroupId
MsgSeqNumber	Número de secuencia del mensaje lógico en el grupo	MsgSeqNumber
Offset	Desplazamiento de datos en el mensaje físico desde el inicio del mensaje lógico	desplazamiento
MsgFlags	Distintivos de mensajes	MQMD - Campo MsgFlags
OriginalLength	Longitud del mensaje original	OriginalLength

AccountingToken (MQBYTE32)

Es la señal de contabilidad, parte del **contexto de identidad** del mensaje. Para obtener más información sobre el contexto de mensaje, consulte "Visión general de MQMD" en la página 394; consulte también Contexto de mensaje.

AccountingToken permite a una aplicación cobrar adecuadamente por el trabajo realizado como resultado del mensaje. El gestor de colas trata esta información como una serie de bits y no comprueba su contenido.

El gestor de colas genera esta información como se indica a continuación:

- El primer byte del campo se establece en la longitud de la información de contabilidad presente en los bytes siguientes; esta longitud está en el rango de cero a 30, y se almacena en el primer byte como un entero binario.
- El segundo y los bytes subsiguientes (según lo especificado por el campo de longitud) se establecen en la información de contabilidad adecuada para el entorno.
  - En z/OS , la información de contabilidad se establece en:
    - Para el lote z/OS , la información de contabilidad de la tarjeta JES JOB o de una sentencia ACCT de JES en la tarjeta EXEC (los separadores de coma se cambian a X'FF '). Esta información se trunca, si es necesario, a 31 bytes.
    - Para TSO, el número de cuenta del usuario.

- Para CICS, el identificador de unidad de trabajo de LU 6.2 (UEPUOWDS) (26 bytes).
- Para IMS, el nombre PSB de 8 caracteres concatenado con la señal de recuperación IMS de 16 caracteres.

– En IBM i, la información de contabilidad se establece en el código de contabilidad del trabajo.
– En sistemas UNIX , la información de contabilidad se establece en el identificador de usuario numérico, en caracteres ASCII.
– En Windows, la información de contabilidad se establece en un identificador de seguridad (SID) de Windows en un formato comprimido. El SID identifica de forma exclusiva el identificador de usuario almacenado en el campo *UserIdentifier* . Cuando el SID se almacena en el campo *AccountingToken* , se omite la autoridad de identificador de 6 bytes (ubicada en el tercer y subsiguiente bytes del SID). Por ejemplo, si el SID de Windows tiene una longitud de 28 bytes, se almacenan 22 bytes de información de SID en el campo *AccountingToken* .

- El último byte (byte 32) del campo de contabilidad se establece en el tipo de señal de contabilidad (en este caso MQACTT_NT_SECURITY_ID, x '0b'):

  MQACTT_CICS_LUOW_ID
  Identificador LUOW de CICS .

  MQACTT_NT_SECURITY_ID
  Identificador de seguridad de Windows .

  MQACTT_OS400_ACCOUNT_TOKEN
  Señal de contabilidad de IBM i .

  MQACTT_UNIX_NUMERIC_ID
  Identificador numérico de sistemas UNIX .

  MQACTT_USER
  Señal de contabilidad definida por el usuario.

  MQACTT_UNKNOWN
  Tipo de señal de contabilidad desconocido.

  El tipo de señal de contabilidad se establece en un valor explícito sólo en los entornos siguientes: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más clientes MQI de WebSphere MQ conectados a estos sistemas. En otros entornos, el tipo de señal de contabilidad se establece en el valor MQACTT_UNKNOWN. En estos entornos, utilice el campo *PutApplType* para deducir el tipo de señal de contabilidad recibida.

- Todos los demás bytes se establecen en cero binario.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT en el parámetro *PutMsgOpts* . Si no se especifica MQPMO_SET_IDENTITY_CONTEXT ni MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje .

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1 , este campo contiene el *AccountingToken* que se ha transmitido con el mensaje si se ha colocado en una cola. Este será el valor de *AccountingToken* que se conserva con el mensaje si se conserva (consulte la descripción de MQPMO_RETAIN en“Opciones de MQPMO (MQLONG)” en la página 482 para obtener más detalles sobre las publicaciones retenidas), pero no se utiliza como *AccountingToken* cuando el mensaje se envía como una publicación a los suscriptores, ya que proporcionan un valor para alterar temporalmente *AccountingToken* en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo es totalmente binario cero.

Es un campo de salida para la llamada MQGET.

Este campo no está sujeto a ninguna conversión basada en el juego de caracteres del gestor de colas; el campo se trata como una serie de bits y no como una serie de caracteres.

El gestor de colas no hace nada con la información de este campo. La aplicación debe interpretar la información si desea utilizarla con fines contables.

Puede utilizar el siguiente valor especial para el campo *AccountingToken* :

MQACT_NONE
No se ha especificado ninguna señal de contabilidad.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, la constante MQACT_NONE_ARRAY también está definida; tiene el mismo valor que MQACT_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_ACCOUNTING_TOKEN_LENGTH. El valor inicial de este campo es MQACT_NONE.

Datos de ApplIdentity(MQCHAR32)

Forma parte del **contexto de identidad** del mensaje. Para obtener más información sobre el contexto de mensaje, consulte "Visión general de MQMD" en la página 394 y Contexto de mensaje .

ApplIdentityData es información definida por la suite de aplicaciones y se puede utilizar para proporcionar información adicional sobre el mensaje o su originador. El gestor de colas trata esta información como datos de tipo carácter, pero no define su formato. Cuando el gestor de colas genera esta información, está totalmente en blanco.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT en el parámetro *PutMsgOpts* . Si hay un carácter nulo, el gestor de colas convierte el valor nulo y los caracteres siguientes en espacios en blanco. Si no se especifica MQPMO_SET_IDENTITY_CONTEXT ni MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida. Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1 , este campo contiene el *ApplIdentityData* que se ha transmitido con el mensaje si se ha colocado en una cola. Este será el valor de *ApplIdentityData* que se mantiene con el mensaje si se conserva (consulte la descripción de MQPMO_RETAIN para obtener más detalles sobre las publicaciones retenidas) pero no se utiliza como *ApplIdentityData* cuando el mensaje se envía como publicación a los suscriptores porque proporcionan un valor para alterar temporalmente *ApplIdentityData* en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo está totalmente en blanco.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona MQ_APPL_IDENTITY_DATA_LENGTH. El valor inicial de este campo es la serie nula en C y 32 caracteres en blanco en otros lenguajes de programación.

Datos de ApplOrigin(MQCHAR4)

Forma parte del **contexto de origen** del mensaje. Para obtener más información sobre el contexto de mensaje, consulte "Visión general de MQMD" en la página 394 y Contexto de mensaje .

ApplOriginData es información definida por la suite de aplicaciones que se puede utilizar para proporcionar información adicional sobre el origen del mensaje. Por ejemplo, las aplicaciones que se ejecutan con autorización de usuario adecuada podrían establecerla para indicar si los datos de identidad son de confianza.

El gestor de colas trata esta información como datos de tipo carácter, pero no define su formato. Cuando el gestor de colas genera esta información, está totalmente en blanco.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_ALL_CONTEXT en el parámetro *PutMsgOpts* . Se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona MQ_APPL_ORIGIN_DATA_LENGTH. El valor inicial de este campo es la serie nula en C y 4 caracteres en blanco en otros lenguajes de programación.

Cuando se publica el mensaje, aunque se ha establecido *ApplOriginData* , está en blanco en la suscripción que recibe.

BackoutCount (MQLONG)

Es un recuento del número de veces que la llamada MQGET ha devuelto previamente el mensaje como parte de una unidad de trabajo y, posteriormente, se ha restituido. Ayuda a la aplicación a detectar errores de proceso que se basan en el contenido del mensaje. El recuento excluye las llamadas MQGET que especifican cualquiera de las opciones MQGMO_BROWSE_ *.

La precisión de este recuento se ve afectada por el atributo de cola *HardenGetBackout* ; consulte "Atributos para colas" en la página 816.

En z/OS, un valor de 255 significa que el mensaje se ha restituido 255 o más veces; el valor devuelto nunca es mayor que 255.

Es un campo de salida para la llamada MQGET. Se ignora para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es 0.

CodedCharSetId (MQLONG)
Este campo especifica el identificador de juego de caracteres de los datos de caracteres dentro del cuerpo del mensaje.

Nota: Los datos de tipo carácter en MQMD y las otras estructuras de datos de MQ que son parámetros en las llamadas deben estar en el juego de caracteres del gestor de colas. Esto lo define el atributo *CodedCharSetId* del gestor de colas; consulte "Atributos para el gestor de colas" en la página 780 para obtener detalles de este atributo.

Si este campo se establece en MQCCSI_Q_MGR al llamar a MQGET con MQGMO_CONVERT en las opciones, el comportamiento es diferente entre las aplicaciones cliente y servidor. Para las aplicaciones de servidor, la página de códigos utilizada para la conversión de caracteres es el *CodedCharSetId* del gestor de colas; para las aplicaciones cliente, la página de códigos utilizada para la conversión de caracteres es la página de códigos del entorno local actual.

Para las aplicaciones cliente, se rellena MQCCSI_Q_MGR, basándose en el entorno local del cliente en lugar del del gestor de colas. La excepción a esa regla es cuando coloca un mensaje en una cola de puente IMS ; lo que se devuelve, en el campo *CodedCharSetId* de MQMD, es el CCSID del gestor de colas.

No debe utilizar el siguiente valor especial:

MQCCSI_APPL
Esto da como resultado un valor incorrecto en el campo *CodedCharSetId* del MQMD y provoca un código de retorno de MQRC_SOURCE_CCSID_ERROR (o MQRC_FORMAT_ERROR para z/OS) cuando se recibe el mensaje utilizando la llamada MQGET con la opción MQGMO_CONVERT.

Puede utilizar los siguientes valores especiales:

MQCCSI_Q_MGR
Los datos de caracteres del mensaje están en el juego de caracteres del gestor de colas.

En las llamadas MQPUT y MQPUT1 , el gestor de colas cambia este valor en el MQMD que se envía con el mensaje al identificador de juego de caracteres verdadero del gestor de colas. Como resultado, la llamada MQGET nunca devuelve el valor MQCCSI_Q_MGR.

MQCCSI_PREDETERMINADO
El *CodedCharSetId* de los datos en el campo *String* está definido por el campo *CodedCharSetId* en la estructura de cabecera que precede a la estructura MQCFH, o por el campo *CodedCharSetId* en el MQMD si el MQCFH está al principio del mensaje.

MQCCSI_INHERIT
Los datos de caracteres del mensaje están en el mismo juego de caracteres que esta estructura; este es el juego de caracteres del gestor de colas. (Sólo para MQMD, MQCCSI_INHERIT tiene el mismo significado que MQCCSI_Q_MGR).

El gestor de colas cambia este valor en el MQMD que se envía con el mensaje al identificador de juego de caracteres real de MQMD. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

No utilice MQCCSI_INHERIT si el valor del campo *PutApplType* en MQMD es MQAT_BROKER.

MQCCSI_EMBEDDED
Los datos de tipo carácter del mensaje están en un juego de caracteres con el identificador contenido en los propios datos del mensaje. Puede haber cualquier número de identificadores de juego de caracteres incluidos en los datos del mensaje, que se aplican a diferentes partes de los datos. Este valor debe utilizarse para mensajes PCF (con un formato de MQFMT_ADMIN, MQFMT_EVENT o MQFMT_PCF) que contengan datos en una mezcla de juegos de caracteres. Cada estructura MQCFST, MQCFSL y MQCFSF contenida en el mensaje PCF debe tener especificado un identificador de juego de caracteres explícito y no MQCCSI_DEFAULT.

Si un mensaje con el formato MQFMT_EMBEDDED_PCF va a contener datos en una mezcla de juegos de caracteres, no utilice MQCCSI_EMBEDDED. En su lugar, establezca MQEPH_CCSID_EMBEDDED en el campo Distintivos de la estructura MQEPH. Esto es equivalente a establecer MQCCSI_EMBEDDED en la estructura anterior. Cada estructura MQCFST, MQCFSL y MQCFSF contenida en el mensaje PCF debe tener especificado un identificador de juego de caracteres explícito y no MQCCSI_DEFAULT. Para obtener más información sobre la estructura MQEPH, consulte "MQEPH - Cabecera PCF incrustada" en la página 338.

Especifique este valor sólo en las llamadas MQPUT y MQPUT1 . Si se especifica en la llamada MQGET, impide la conversión del mensaje.

En las llamadas MQPUT y MQPUT1 , el gestor de colas cambia los valores MQCCSI_Q_MGR y MQCCSI_INHERIT en el MQMD que se envía con el mensaje tal como se ha descrito anteriormente, pero no cambia el MQMD especificado en la llamada MQPUT o MQPUT1 . No se realiza ninguna otra comprobación en el valor especificado.

Las aplicaciones que recuperan mensajes deben comparar este campo con el valor que espera la aplicación; si los valores difieren, es posible que la aplicación tenga que convertir datos de tipo carácter en el mensaje.

Si especifica la opción MQGMO_CONVERT en la llamada MQGET, este campo es un campo de entrada/salida. El valor especificado por la aplicación es el identificador del juego de caracteres codificado al que convertir los datos del mensaje si es necesario. Si la conversión es satisfactoria o innecesaria, el valor no se modifica (excepto que el valor MQCCSI_Q_MGR o MQCCSI_INHERIT se convierte en el valor real). Si la conversión no es satisfactoria, el valor después de la llamada MQGET representa el identificador de juego de caracteres codificado del mensaje no convertido que se devuelve a la aplicación.

De lo contrario, es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQCCSI_Q_MGR.

CorrelId (MQBYTE24)
El campo CorrelId es una propiedad de la cabecera de mensaje que se puede utilizar para identificar un mensaje específico o un grupo de mensajes.

Es una serie de bytes que la aplicación puede utilizar para relacionar un mensaje con otro, o para relacionar el mensaje con otro trabajo que la aplicación está realizando. El identificador de correlación es una propiedad permanente del mensaje y persiste en los reinicios del gestor de colas. Puesto que el identificador de correlación es una serie de bytes y no una serie de caracteres, el identificador de correlación *no* se convierte entre juegos de caracteres cuando el mensaje fluye de un gestor de colas a otro.

Para las llamadas MQPUT y MQPUT1 , la aplicación puede especificar cualquier valor. El gestor de colas transmite este valor con el mensaje y lo entrega a la aplicación que emite la solicitud get para el mensaje.

Si la aplicación especifica MQPMO_NEW_CORREL_ID, el gestor de colas genera un identificador de correlación exclusivo que se envía con el mensaje y también se devuelve a la aplicación emisora en la salida de la llamada MQPUT o MQPUT1 .

Un identificador de correlación generado por el gestor de colas consta de un identificador de producto de 3 bytes (AMQ o CSQ en ASCII o EBCDIC), seguido de un byte reservado y una implementación específica del producto de una serie exclusiva. En WebSphere MQ esta serie de implementación específica del producto contiene los primeros 12 caracteres del nombre del gestor de colas y un valor derivado del reloj del sistema. Por lo tanto, todos los gestores de colas que pueden intercomunicarse deben tener nombres que difieran en los primeros 12 caracteres para asegurarse de que los identificadores de mensaje son exclusivos. La capacidad de generar una serie exclusiva también depende de que el reloj del sistema no se cambie hacia atrás. Para eliminar la posibilidad de que un identificador de mensaje generado por el gestor de colas duplique uno generado por la aplicación, la aplicación debe evitar generar identificadores con caracteres iniciales en el rango de A a I en ASCII o EBCDIC (de X'41 'a X'49' y de X'C1'a X'C9'). Sin embargo, no se impide que la aplicación genere identificadores con caracteres iniciales en estos rangos.

Este identificador de correlación generado se conserva con el mensaje si se conserva y se utiliza como identificador de correlación cuando el mensaje se envía como publicación a los suscriptores que especifican MQCI_NONE en el campo de ID SubCorrelen el MQSD pasado en la llamada MQSUB. Consulte Opciones de MQPMO para obtener más detalles sobre las publicaciones retenidas.

Cuando el gestor de colas o un agente de canal de mensajes genera un mensaje de informe, establece el campo *CorrelId* de la forma especificada por el campo *Report* del mensaje original, MQRO_COPY_MSG_ID_TO_CORREL_ID o MQRO_PASS_CORREL_ID. Las aplicaciones que generan mensajes de informe también deben hacerlo.

Para la llamada MQGET, *CorrelId* es uno de los cinco campos que se pueden utilizar para seleccionar un mensaje determinado para recuperarlo de la cola. Consulte la descripción del campo *MsgId* para obtener detalles sobre cómo especificar valores para este campo.

Especificar MQCI_NONE como identificador de correlación tiene el mismo efecto que *no* especificar MQMO_MATCH_CORREL_ID, es decir, *cualquier* identificador de correlación coincidirá.

Si se especifica la opción MQGMO_MSG_UNDER_CURSOR en el parámetro *GetMsgOpts* de la llamada MQGET, este campo se ignora.

Al volver de una llamada MQGET, el campo *CorrelId* se establece en el identificador de correlación del mensaje devuelto (si lo hay).

Se pueden utilizar los siguientes valores especiales:

MQCI_NONE
No se ha especificado ningún identificador de correlación.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQCI_NONE_ARRAY; tiene el mismo valor que MQCI_NONE, pero es una matriz de caracteres en lugar de una serie.

MQCI_SESIÓN_NUEVA
El mensaje es el inicio de una nueva sesión.

El puente CICS reconoce este valor como que indica el inicio de una nueva sesión, es decir, el inicio de una nueva secuencia de mensajes.

Para el lenguaje de programación C, la constante MQCI_NEW_SESSION_ARRAY también está definida; tiene el mismo valor que MQCI_NEW_SESSION, pero es una matriz de caracteres en lugar de una serie.

Para la llamada MQGET, es un campo de entrada/salida. Para las llamadas MQPUT y MQPUT1 , este es un campo de entrada si se especifica MQPMO_NEW_CORREL_ID *no* y un campo de salida si se especifica MQPMO_NEW_CORREL_ID *es* . La longitud de este campo la proporciona MQ_CORREL_ID_LENGTH. El valor inicial de este campo es MQCI_NONE.

Nota:

No puede pasar el identificador de correlación de una publicación en una jerarquía. El campo lo utiliza el gestor de colas.

Encoding (MQLONG)

Especifica la codificación numérica de los datos numéricos del mensaje; no se aplica a los datos numéricos de la propia estructura MQMD. La codificación numérica define la representación utilizada para enteros binarios, enteros decimales empaquetados y números de coma flotante.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. El gestor de colas no comprueba que el campo sea válido. Se define el siguiente valor especial:

MQENC_NATIVE
La codificación es el valor predeterminado para el lenguaje de programación y la máquina en la que se ejecuta la aplicación.

Nota: El valor de esta constante depende del lenguaje de programación y del entorno. Por este motivo, las aplicaciones deben compilarse utilizando los archivos de cabecera, macro, COPY o INCLUDE adecuados para el entorno en el que se ejecutará la aplicación.

Las aplicaciones que colocan mensajes suelen especificar MQENC_NATIVE. Las aplicaciones que recuperan mensajes deben comparar este campo con el valor MQENC_NATIVE; si los valores difieren, es posible que la aplicación tenga que convertir datos numéricos en el mensaje. Utilice la opción MQGMO_CONVERT para solicitar al gestor de colas que convierta el mensaje como parte del proceso de la llamada MQGET. Consulte "Codificaciones de máquina" en la página 881 para obtener detalles sobre cómo se construye el campo *Encoding* .

Si especifica la opción MQGMO_CONVERT en la llamada MQGET, este campo es un campo de entrada/salida. El valor especificado por la aplicación es la codificación a la que convertir los datos del mensaje si es necesario. Si la conversión es satisfactoria o innecesaria, el valor no se modifica. Si la conversión no es satisfactoria, el valor después de la llamada MQGET representa la codificación del mensaje no convertido que se devuelve a la aplicación.

En otros casos, es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQENC_NATIVE.

Expiry (MQLONG)

Es un periodo de tiempo expresado en décimas de segundo, establecido por la aplicación que coloca el mensaje. El mensaje se debe poder seleccionar para descartarlo si no se ha suprimido de la cola de destino antes de que transcurra este período de tiempo.

El valor se reduce para reflejar el tiempo que el mensaje pasa en la cola de destino, y también en cualquier cola de transmisión intermedia si la transferencia es a una cola remota. También pueden disminuir los agentes de canal de mensajes para reflejar los tiempos de transmisión, si estos son significativos. Del mismo modo, una aplicación que reenvía este mensaje a otra cola puede disminuir el valor si es necesario, si ha retenido el mensaje durante un tiempo significativo. Sin embargo, la hora de caducidad se trata como aproximada y no es necesario reducir el valor para reflejar intervalos de tiempo pequeños.

Cuando una aplicación recupera el mensaje utilizando la llamada MQGET, el campo *Expiry* representa la cantidad de tiempo de caducidad original que todavía permanece.

Una vez transcurrido el tiempo de caducidad de un mensaje, éste pasa a ser elegible para ser descartado por el gestor de colas. El mensaje se descarta cuando se produce una llamada MQGET de examinar o no examinar que habría devuelto el mensaje si no hubiera caducado. Por ejemplo, una llamada MQGET sin examinar con el campo *MatchOptions* en MQGMO establecido en MQMO_NONE leyendo de una cola ordenada FIFO descarta todos los mensajes caducados hasta el primer mensaje no caducado. Con una cola ordenada de prioridad, la misma llamada descartará los mensajes caducados de prioridad más alta y los mensajes de prioridad igual que han llegado a la cola antes del primer mensaje no caducado.

Un mensaje que ha caducado nunca se devuelve a una aplicación (mediante una llamada MQGET de examinar o no examinar), por lo que el valor del campo *Expiry* del descriptor de mensaje después de una llamada MQGET satisfactoria es mayor que cero o el valor especial MQEI_UNLIMITED.

Si un mensaje se coloca en una cola remota, el mensaje puede caducar (y descartarse) mientras está en una cola de transmisión intermedia, antes de que el mensaje llegue a la cola de destino.

Se genera un informe cuando se descarta un mensaje caducado, si el mensaje ha especificado una de las opciones de informe MQRO_EXPIRATION_ *. Si no se especifica ninguna de estas opciones, no se genera ningún informe de este tipo; se presupone que el mensaje ya no es relevante después de este periodo de tiempo (quizás porque un mensaje posterior lo ha reemplazado).

Para un mensaje colocado en un punto de sincronización, el intervalo de caducidad empieza en el momento en que se coloca el mensaje, no en el momento en que se confirma el punto de sincronización. Es posible que el intervalo de caducidad pueda pasar antes de que se confirme el punto de sincronización. En este caso, el mensaje se descartará en algún momento después de la operación de confirmación y el mensaje no se devolverá a una aplicación en respuesta a una operación MQGET.

Cualquier otro programa que descarte mensajes basándose en la hora de caducidad también debe enviar un mensaje de informe adecuado si se ha solicitado uno.

Nota:

1. Si se coloca un mensaje con una hora *Expiry* de cero o un número mayor que 999 999 999, la llamada MQPUT o MQPUT1 falla con el código de razón MQRC_EXPIRY_ERROR; en este caso no se genera ningún mensaje de informe.
2. Debido a que un mensaje con un tiempo de caducidad que ha transcurrido puede que no se descarte hasta más adelante, es posible que haya mensajes en una cola que hayan pasado su tiempo de caducidad y que, por lo tanto, no sean elegibles para la recuperación. No obstante, estos mensajes cuentan para el número de mensajes en la cola para todos los fines, incluido el desencadenamiento de profundidad.
3. Se genera un informe de caducidad, si se solicita, cuando se descarta el mensaje, no cuando pasa a ser elegible para descartarlo.
4. Descartar un mensaje caducado y generar un informe de caducidad si se solicita, nunca forman parte de la unidad de trabajo de la aplicación, aunque el mensaje se haya planificado para descartarlo como resultado de una llamada MQGET que opera dentro de una unidad de trabajo.
5. Si una llamada MQGET recupera un mensaje casi caducado dentro de una unidad de trabajo y posteriormente se restituye la unidad de trabajo, es posible que el mensaje sea apto para ser descartado antes de que se pueda recuperar de nuevo.
6. Si un mensaje casi caducado está bloqueado por una llamada MQGET con MQGMO_LOCK, es posible que el mensaje sea apto para ser descartado antes de que pueda ser recuperado por una llamada MQGET con MQGMO_MSG_UNDER_CURSOR; el código de razón MQRC_NO_MSG_UNDER_CURSOR se devuelve en esta llamada MQGET posterior si esto sucede.
7. Cuando se recupera un mensaje de solicitud con un tiempo de caducidad mayor que cero, la aplicación puede realizar una de las acciones siguientes cuando envía el mensaje de respuesta:
   - Copie el tiempo de caducidad restante del mensaje de solicitud en el mensaje de respuesta.
   - Establezca la hora de caducidad en el mensaje de respuesta en un valor explícito mayor que cero.
   - Establezca la hora de caducidad en el mensaje de respuesta en MQEI_UNLIMITED.

   La acción a realizar depende del diseño de la aplicación. Sin embargo, la acción predeterminada para transferir mensajes a una cola de mensajes no entregados (undelivered-message) debe ser conservar el tiempo de caducidad restante del mensaje y continuar decrementándolo.
8. Los mensajes desencadenantes siempre se generan con MQEI_UNLIMITED.
9. Un mensaje (normalmente en una cola de transmisión) que tiene un nombre *Format* de MQFMT_XMIT_Q_HEADER tiene un segundo descriptor de mensaje dentro de MQXQH. Por lo tanto, tiene dos campos *Expiry* asociados. En este caso deben señalarse los siguientes puntos adicionales:
   - Cuando una aplicación coloca un mensaje en una cola remota, el gestor de colas coloca el mensaje inicialmente en una cola de transmisión local y añade un prefijo a los datos del mensaje de aplicación con una estructura MQXQH. El gestor de colas establece los valores de los dos campos *Expiry* para que sean los mismos que los especificados por la aplicación.

Si una aplicación coloca un mensaje directamente en una cola de transmisión local, los datos del mensaje ya deben empezar con una estructura MQXQH y el nombre de formato debe ser MQFMT_XMIT_Q_HEADER. En este caso, no es necesario que la aplicación establezca los valores de estos dos campos *Expiry* para que sean iguales. (El gestor de colas comprueba que el campo *Expiry* de MQXQH contiene un valor válido y que los datos del mensaje son lo suficientemente largos para incluirlo). Para una aplicación que puede escribir directamente en la cola de transmisión, la aplicación tiene que crear una cabecera de cola de transmisión con el descriptor de mensaje incorporado. Sin embargo, si el valor de caducidad del descriptor de mensaje grabado en la cola de transmisión no es coherente con el valor del descriptor de mensaje incorporado, se produce un rechazo de error de caducidad.

- Cuando un mensaje con un nombre *Format* de MQFMT_XMIT_Q_HEADER se recupera de una cola (ya sea una cola normal o de transmisión), el gestor de colas disminuye *ambos* estos *Expiry* campos con el tiempo empleado en esperar en la cola. No se genera ningún error si los datos del mensaje no son lo suficientemente largos para incluir el campo *Expiry* en MQXQH.
- El gestor de colas utiliza el campo *Expiry* en el descriptor de mensaje independiente (es decir, no el del descriptor de mensaje incorporado en la estructura MQXQH) para probar si el mensaje es apto para descartarlo.
- Si los valores iniciales de los dos campos *Expiry* son diferentes, la hora *Expiry* en el descriptor de mensaje separado cuando se recupera el mensaje puede ser mayor que cero (por lo que el mensaje no es elegible para descartarlo), mientras que ha transcurrido la hora según el campo *Expiry* en MQXQH. En este caso, el campo *Expiry* en MQXQH se establece en cero.

10. El tiempo de caducidad de un mensaje de respuesta devuelto desde el puente IMS es ilimitado a menos que se establezca MQIIH_PASS_EXPIRATION en el campo Distintivos de MQIIH. Consulte Distintivos para obtener más información.

Se reconoce el siguiente valor especial:

MQEI_UNLIMITED
El mensaje tiene un tiempo de caducidad ilimitado.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQEI_UNLIMITED.

Feedback (MQLONG)
El campo Comentarios se utiliza con un mensaje de tipo MQMT_REPORT para indicar la naturaleza del informe y sólo es significativo con ese tipo de mensaje.

El campo puede contener uno de los valores MQFB_ * o uno de los valores MQRC_ *. Los códigos de comentarios se agrupan de la forma siguiente:

MQFB_NONE
No se han proporcionado comentarios.

MQFB_SISTEMA_PRIMERO
Valor más bajo para los comentarios generados por el sistema.

MQFB_SYSTEM_LAST
Valor más alto para comentarios generados por el sistema.

El rango de códigos de comentarios generados por el sistema MQFB_SYSTEM_FIRST a MQFB_SYSTEM_LAST incluye los códigos de comentarios generales listados en este tema (MQFB_ *), y también los códigos de razón (MQRC_ *) que se pueden producir cuando el mensaje no se puede colocar en la cola de destino.

MQFB_APPL_FIRST
Valor más bajo para los comentarios generados por la aplicación.

MQFB_APPL_LAST
Valor más alto para los comentarios generados por la aplicación.

Las aplicaciones que generan mensajes de informe no deben utilizar códigos de comentarios en el rango del sistema (que no sean MQFB_QUIT), a menos que deseen simular mensajes de informe generados por el gestor de colas o el agente de canal de mensajes.

En las llamadas MQPUT o MQPUT1 , el valor especificado debe ser MQFB_NONE o estar dentro del rango del sistema o del rango de aplicaciones. Esto se comprueba independientemente del valor de *MsgType*.

Códigos de comentarios generales:

MQFB_COA
Confirmación de llegada a la cola de destino (consulte MQRO_COA).

MQFB_COD
Confirmación de entrega a la aplicación receptora (consulte MQRO_COD).

MQFB_EXPIRATION
El mensaje se ha descartado porque no se había eliminado de la cola de destino antes de que hubiera transcurrido su tiempo de caducidad.

MQFB_PAN
Notificación de acción positiva (consulte MQRO_PAN).

MQFB_NAN
Notificación de acción negativa (consulte MQRO_NAN).

MQFB_QUIT
Finalizar aplicación.

Esto lo puede utilizar un programa de planificación de carga de trabajo para controlar el número de instancias de un programa de aplicación que se están ejecutando. El envío de un mensaje MQMT_REPORT con este código de retorno a una instancia del programa de aplicación indica a dicha instancia que debe detener el proceso. Sin embargo, el cumplimiento de este convenio es un asunto de la aplicación; el gestor de colas no lo impone.

Códigos de comentarios de canal:

MQFB_CHANNEL_COMPLETED
Un canal ha finalizado normalmente.

MQFB_CHANNEL_FAIL
Un canal ha finalizado de forma anómala y pasa al estado STOPPED.

MQFB_CHANNEL_FAIL_RETRY
Un canal ha finalizado de forma anómala y entra en estado RETRY.

IMS-bridge feedback codes

Estos códigos se utilizan cuando se recibe un código de detección IMS-OTMA inesperado. El código de detección o, cuando el código de detección es 0x1A , el código de razón asociado con ese código de detección, se indica en *Comentarios*.

1. Para los códigos *Feedback* en el rango MQFB_IMS_FIRST (300) a MQFB_IMS_LAST (399), se ha recibido un código de detección distinto de 0x1A . El *código de detección* lo proporciona la expresión (*Feedback* - MQFB_IMS_FIRST+1)
2. Para los códigos *Feedback* en el rango MQFB_IMS_NACK_1A_REASON_FIRST (600) a MQFB_IMS_NACK_1A_REASON_LAST (855), se ha recibido un código de detección de 0x1A . El *código de razón* asociado con el código de detección lo proporciona la expresión (*Feedback* - MQFB_IMS_NACK_1A_REASON_FIRST)

El significado de los códigos de detección IMS-OTMA y los códigos de razón correspondientes se describen en la publicación *Open Transaction Manager Access Guide and Reference*.

El puente IMS puede generar los siguientes códigos de comentarios:

MQFB_DATA_LENGTH_ZERO
Una longitud de segmento era cero en los datos de aplicación del mensaje.

MQFB_DATA_LENGTH_NEGATIVO
Una longitud de segmento era negativa en los datos de aplicación del mensaje.

MQFB_DATA_LENGTH_TOO_BIG
Una longitud de segmento era demasiado grande en los datos de aplicación del mensaje.

MQFB_BUFFER_OVERFLOW
El valor de uno de los campos de longitud haría que los datos desbordaran el almacenamiento intermedio de mensajes.

MQFB_LENGTH_OFF_BY_ONE
El valor de uno de los campos de longitud era 1 byte demasiado corto.

MQFB_IIH_ERROR
El campo *Format* en MQMD especifica MQFMT_IMS, pero el mensaje no empieza por una estructura MQIIH válida.

MQFB_NOT_AUTHORIZED_FOR_IMS
El ID de usuario contenido en el descriptor de mensaje MQMD, o la contraseña contenida en el campo *Authenticator* de la estructura MQIIH, ha fallado la validación realizada por el puente IMS . Como resultado, el mensaje no se ha pasado a IMS.

MQFB_IMS_ERROR
IMSha devuelto un error inesperado. Consulte el registro de errores de WebSphere MQ en el sistema en el que reside el puente IMS para obtener más información sobre el error.

MQFB_IMS_FIRST
Cuando el código de detección de IMS-OTMA no es 0x1A, los códigos de comentarios generados por IMSestán en el rango de MQFB_IMS_FIRST (300) a MQFB_IMS_LAST (399). El propio código de detección IMS-OTMA es *Feedback* menos MQFB_IMS_ERROR.

MQFB_IMS_LAST
Valor más alto para los comentarios generados por IMScuando el código de detección no es 0x1A.

MQFB_IMS_NACK_1A_REASON_FIRST
Cuando el código de detección es 0x1A, los códigos de comentarios generados por IMSestán en el rango de MQFB_IMS_NACK_1A_REASON_FIRST (600) a MQFB_IMS_NACK_1A_REASON_LAST (855).

MQFB_IMS_NACK_1A_REASON_LAST
Valor más alto para los comentarios generados por IMScuando el código de detección es 0x1A

Códigos de comentarios de puente CICS: el puente CICS puede generar los siguientes códigos de comentarios:

MQFB_CICS_APPL_ABENDED
El programa de aplicación especificado en el mensaje ha finalizado de forma anómala. Este código de retorno sólo aparece en el campo *Reason* de la estructura MQDLH.

MQFB_CICS_APPL_NOT_STARTED
El EXEC CICS LINK para el programa de aplicación especificado en el mensaje ha fallado. Este código de retorno sólo aparece en el campo *Reason* de la estructura MQDLH.

MQFB_CICS_BRIDGE_FAILURE
El puente CICS ha terminado de forma anómala sin completar el proceso normal de errores.

MQFB_CICS_CCSID_ERROR
Identificador de juego de caracteres no válido.

MQFB_CICS_CIH_ERROR
Falta la estructura de cabecera de información de CICS o no es válida.

MQFB_CICS_COMMAREA_ERROR
La longitud de COMMAREA CICS no es válida.

MQFB_CICS_CORREL_ID_ERROR
Identificador de correlación no válido.

MQFB_CICS_DLQ_ERROR
La tarea de puente CICS no ha podido copiar una respuesta a esta solicitud en la cola de mensajes no entregados. La solicitud se ha restituido.

MQFB_CICS_ENCODING_ERROR
Codificación no válida.

MQFB_CICS_INTERNAL_ERROR
El puente CICS ha encontrado un error inesperado.

Este código de retorno sólo aparece en el campo *Reason* de la estructura MQDLH.

MQFB_CICS_NOT_AUTHORIZED
Identificador de usuario no autorizado o contraseña no válida.

Este código de retorno sólo aparece en el campo *Reason* de la estructura MQDLH.

MQFB_CICS_UOW_BACKED_OUT
La unidad de trabajo fue restituido, por una de las siguientes razones:

- Se ha detectado una anomalía al procesar otra solicitud dentro de la misma unidad de trabajo.
- Se ha producido una terminación anómala de CICS mientras la unidad de trabajo estaba en curso.

MQFB_CICS_UOW_ERROR
Campo de control de unidad de trabajo *UOWControl* no válido.

Códigos de retorno de mensaje de ruta de rastreo:

MQFB_ACTIVITY
Se utiliza con el formato MQFMT_EMBEDDED_PCF para permitir la opción de datos de usuario después de los informes de actividad.

MQFB_MAX_ACTIVITIES
Se devuelve cuando se descarta el mensaje de ruta de rastreo porque el número de actividades en las que se ha implicado el mensaje supera el límite máximo de actividades.

MQFB_NO_REENVIADO
Se devuelve cuando se descarta el mensaje de ruta de rastreo porque está a punto de enviarse a un gestor de colas remoto que no soporta mensajes de ruta de rastreo.

MQFB_NO_ENTREGADO
Se devuelve cuando se descarta el mensaje de ruta de rastreo porque está a punto de colocarse en una cola local.

MQFB_UNSUPPORTED_FORWARDING
Se devuelve cuando se descarta el mensaje de ruta de rastreo porque un valor del parámetro de reenvío no se reconoce y está en la máscara de bits rechazada.

MQFB_UNSUPPORTED_DELIVERY
Se devuelve cuando se descarta el mensaje de ruta de rastreo porque un valor del parámetro de entrega no se reconoce y está en la máscara de bits rechazada.

Códigos de razón deWebSphere MQ: para mensajes de informe de excepción, *Feedback* contiene un código de razón de WebSphere MQ . Entre los posibles códigos de razón están:

MQRC_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida
(2051, X'803 ') Llamadas de colocación inhibidas para la cola.

MQRC_Q_FULL
(2053, X'805 ') La cola ya contiene el número máximo de mensajes.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808 ') No hay espacio disponible en disco para la cola.

MQRC_PERSISTENT_NOT_ALLOWED
(2048, X'800 ') La cola no admite mensajes persistentes.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') Longitud de mensaje mayor que el máximo para el gestor de colas.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Longitud de mensaje mayor que el máximo para la cola.

Para obtener una lista completa de códigos de razón, consulte:

- Para WebSphere MQ para z/OS, consulte Códigos de razón de API.
- Para todas las demás plataformas, consulte Códigos de terminación y razón de la API.

.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQFB_NONE.

Format (MQCHAR8)

Es un nombre que el remitente del mensaje utiliza para indicar al destinatario la naturaleza de los datos del mensaje. Los caracteres que estén en el juego de caracteres del gestor de colas se pueden especificar para el nombre, pero debe restringir el nombre a lo siguiente:

- Mayúsculas de la A a la Z
- Dígitos numéricos del 0 al 9

Si se utilizan otros caracteres, es posible que no sea posible convertir el nombre entre los juegos de caracteres de los gestores de colas emisor y receptor.

Rellene el nombre con espacios en blanco hasta la longitud del campo, o utilice un carácter nulo para terminar el nombre antes del final del campo; el nulo y los caracteres subsiguientes se tratan como blancos. No especifique un nombre con espacios en blanco iniciales o intercalados. Para la llamada MQGET, el gestor de colas devuelve el nombre rellenado con espacios en blanco a la longitud del campo.

El gestor de colas no comprueba que el nombre cumpla con las recomendaciones descritas anteriormente.

Los nombres que empiezan por MQ en mayúsculas, minúsculas y combinación de mayúsculas y minúsculas tienen significados definidos por el gestor de colas; no utilice nombres que empiecen por estas letras para sus propios formatos.Los formatos incorporados del gestor de colas son:

MQFMT_NONE
La naturaleza de los datos no está definida: los datos no se pueden convertir cuando el mensaje se recupera de una cola utilizando la opción MQGMO_CONVERT.

Si especifica MQGMO_CONVERT en la llamada MQGET, y el juego de caracteres o la codificación de datos del mensaje difiere del especificado en el parámetro *MsgDesc* , el mensaje se devuelve con los siguientes códigos de terminación y razón (suponiendo que no haya otros errores):

- Código de terminación MQCC_WARNING y código de razón MQRC_FORMAT_ERROR si los datos MQFMT_NONE están al principio del mensaje.
- Código de terminación MQCC_OK y código de razón MQRC_NONE si los datos MQFMT_NONE están al final del mensaje (es decir, precedidos por una o más estructuras de cabecera MQ). Las estructuras de cabecera MQ se convierten al juego de caracteres solicitado y a la codificación en este caso.

Para el lenguaje de programación C, la constante MQFMT_NONE_ARRAY también está definida; tiene el mismo valor que MQFMT_NONE, pero es una matriz de caracteres en lugar de una serie.

MQFMT_ADMIN
El mensaje es un mensaje de petición o respuesta de servidor de mandatos en formato de mandato programable (PCF). Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET. Consulte Utilización de formatos de mandatos programables para obtener más información sobre la utilización de mensajes de formato de mandatos programables.

Para el lenguaje de programación C, también se define la constante MQFMT_ADMIN_ARRAY; tiene el mismo valor que MQFMT_ADMIN, pero es una matriz de caracteres en lugar de una serie.

MQFMT_CICS
Los datos del mensaje empiezan con la cabecera de información MQCIH CICS , seguida de los datos de la aplicación. El nombre de formato de los datos de aplicación se proporciona mediante el campo *Format* en la estructura MQCIH.

En z/OS, especifique la opción MQGMO_CONVERT en la llamada MQGET para convertir mensajes que tengan el formato MQFMT_CICS.

Para el lenguaje de programación C, la constante MQFMT_CICS_ARRAY también está definida; tiene el mismo valor que MQFMT_CICS, pero es una matriz de caracteres en lugar de una serie.

MQFMT_COMMAND_1
El mensaje es un mensaje de respuesta de servidor de mandatos MQSC que contiene el recuento de objetos, el código de terminación y el código de razón. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante MQFMT_COMMAND_1_ARRAY ; tiene el mismo valor que MQFMT_COMMAND_1, pero es una matriz de caracteres en lugar de una serie.

MQFMT_COMMAND_2
El mensaje es un mensaje de respuesta del servidor de mandatos MQSC que contiene información sobre los objetos solicitados. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante MQFMT_COMMAND_2_ARRAY ; tiene el mismo valor que MQFMT_COMMAND_2, pero es una matriz de caracteres en lugar de una serie.

MQFMT_DEAD_LETTER_HEADER
Los datos del mensaje empiezan por la cabecera MQDLH de mensaje no entregado. Los datos del mensaje original siguen inmediatamente la estructura MQDLH. El nombre de formato de los datos de mensaje originales lo proporciona el campo *Format* en la estructura MQDLH; consulte "MQDLH-Cabecera de mensaje no entregado" en la página 326 para obtener detalles de esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Los informes COA y COD no se generan para los mensajes que tienen un *Format* de MQFMT_DEAD_LETTER_HEADER.

Para el lenguaje de programación C, también se define la constante MQFMT_DEAD_LETTER_HEADER_ARRAY; tiene el mismo valor que MQFMT_DEAD_LETTER_HEADER, pero es una matriz de caracteres en lugar de una serie.

MQFMT_DIST_HEADER
Los datos del mensaje empiezan con la cabecera de lista de distribución MQDH; esto incluye las matrices de registros MQOR y MQPMR. La cabecera de lista de distribución puede ir seguida de datos adicionales. El formato de los datos adicionales (si los hay) se proporciona mediante el campo *Format* en la estructura MQDH; consulte "MQDH - Cabecera de distribución" en la página 319 para obtener detalles de esta estructura. Los mensajes con el formato MQFMT_DIST_HEADER se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Este formato está soportado en los entornos siguientes: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más WebSphere MQ clientes MQI conectados a estos sistemas.

Para el lenguaje de programación C, también se define la constante MQFMT_DIST_HEADER_ARRAY; tiene el mismo valor que MQFMT_DIST_HEADER, pero es una matriz de caracteres en lugar de una serie.

MQFMT_EMBEDDED_PCF
Formato para un mensaje de ruta de rastreo, siempre que el valor del mandato PCF se establezca en MQCMD_TRACE_ROUTE. El uso de este formato permite que los datos de usuario se envíen junto con el mensaje de ruta de rastreo, siempre que sus aplicaciones puedan hacer frente a los parámetros PCF anteriores.

La cabecera PCF **debe** ser la primera cabecera, o el mensaje no se tratará como un mensaje de ruta de rastreo. Esto significa que el mensaje no puede estar en un grupo y que los mensajes de ruta de rastreo no se pueden segmentar. Si se envía un mensaje de ruta de rastreo en un grupo, el mensaje se rechaza con el código de razón MQRC_MSG_NOT_ALLOWED_IN_GROUP.

Tenga en cuenta que MQFMT_ADMIN también se puede utilizar para el formato de un mensaje de ruta de rastreo, pero en este caso no se pueden enviar datos de usuario junto con el mensaje de ruta de rastreo.

MQFMT_EVENT
El mensaje es un mensaje de suceso de MQ que notifica un suceso que se ha producido. Los mensajes de sucesos tienen la misma estructura que los mandatos programables; consulte Mensajes de mandatos PCF para obtener más información sobre esta estructura y Supervisión de sucesos para obtener información sobre sucesos.

Los mensajes de suceso Version-1 se pueden convertir en todos los entornos si se especifica la opción MQGMO_CONVERT en la llamada MQGET. Los mensajes de suceso Version-2 sólo se pueden convertir en z/OS.

Para el lenguaje de programación C, la constante MQFMT_EVENT_ARRAY también está definida; tiene el mismo valor que MQFMT_EVENT, pero es una matriz de caracteres en lugar de una serie.

MQFMT_IMS
Los datos del mensaje empiezan con la cabecera de información MQIIH IMS , seguida de los datos de la aplicación. El nombre de formato de los datos de aplicación se proporciona mediante el campo *Format* en la estructura MQIIH.

Para obtener detalles sobre cómo se maneja la estructura MQIIH cuando se utiliza MQGET con MQGMO_CONVERT, consulte “Format (MQCHAR8)” en la página 379 y “Formato ReplyTo(MQCHAR8)” en la página 379.

Para el lenguaje de programación C, la constante MQFMT_IMS_ARRAY también está definida; tiene el mismo valor que MQFMT_IMS, pero es una matriz de caracteres en lugar de una serie.

MQFMT_IMS_VAR_STRING
El mensaje es una serie de variable IMS , que es una serie con el formato `llzzccc`, donde:

ll
es un campo de longitud de 2 bytes que especifica la longitud total del elemento de serie variable IMS . Esta longitud es igual a la longitud de `ll` (2 bytes), más la longitud de `zz` (2 bytes), más la longitud de la propia serie de caracteres. `ll` es un entero binario de 2 bytes en la codificación especificada por el campo *Encoding* .

zz
es un campo de 2 bytes que contiene distintivos que son significativos para IMS. `zz` es una serie de bytes que consta de dos campos MQBYTE y se transmite sin cambiar de remitente a destinatario (es decir, `zz` no está sujeto a ninguna conversión).

ccc
es una serie de caracteres de longitud variable que contiene `ll-4` caracteres. `ccc` está en el juego de caracteres especificado por el campo *CodedCharSetId* .

En z/OS, los datos del mensaje pueden constar de una secuencia de series de variables IMS a tope, con cada serie con el formato `llzzccc`. No debe haber bytes omitidos entre series de variables IMS sucesivas. Esto significa que si la primera serie tiene una longitud impar, la segunda serie estará desalineada, es decir, no empezará en un límite que sea un múltiplo de dos. Tenga cuidado al construir estas series en máquinas que requieren alineación de tipos de datos elementales.

Utilice la opción MQGMO_CONVERT en la llamada MQGET para convertir mensajes que tengan el formato MQFMT_IMS_VAR_STRING.

Para el lenguaje de programación C, también se define la constante MQFMT_IMS_VAR_STRING_ARRAY; tiene el mismo valor que MQFMT_IMS_VAR_STRING, pero es una matriz de caracteres en lugar de una serie.

MQFMT_MD_EXTENSION
Los datos del mensaje empiezan con la extensión de descriptor de mensaje MQMDE y, opcionalmente, van seguidos de otros datos (normalmente, los datos del mensaje de aplicación). El nombre de formato, el juego de caracteres y la codificación de los datos que siguen a MQMDE se proporcionan mediante los campos *Format*, *CodedCharSetId*y *Encoding* en MQMDE. Consulte "MQMDE- Extensión de descriptor de mensaje" en la página 445 para obtener detalles de esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, la constante MQFMT_MD_EXTENSION_ARRAY también está definida; tiene el mismo valor que MQFMT_MD_EXTENSION, pero es una matriz de caracteres en lugar de una serie.

MQFMT_PCF
El mensaje es un mensaje definido por el usuario que se ajusta a la estructura de un mensaje de formato de mandato programable (PCF). Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET. Consulte Utilización de formatos de mandatos programables para obtener más información sobre la utilización de mensajes de formato de mandatos programables.

Para el lenguaje de programación C, la constante MQFMT_PCF_ARRAY también está definida; tiene el mismo valor que MQFMT_PCF, pero es una matriz de caracteres en lugar de una serie.

MQFMT_REF_MSG_HEADER
Los datos del mensaje empiezan con la cabecera de mensaje de referencia MQRMH y, opcionalmente, van seguidos de otros datos. El nombre de formato, el juego de caracteres y la codificación de los datos se proporcionan mediante los campos *Format*, *CodedCharSetId*y *Encoding* en MQRMH. Consulte "MQRMH - Cabecera de mensaje de referencia" en la página 524 para obtener detalles de esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Este formato está soportado en los entornos siguientes: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más WebSphere MQ clientes MQI conectados a estos sistemas.

Para el lenguaje de programación C, también se define la constante MQFMT_REF_MSG_HEADER_ARRAY; tiene el mismo valor que MQFMT_REF_MSG_HEADER, pero es una matriz de caracteres en lugar de una serie.

MQFMT_RF_HEADER
Los datos del mensaje empiezan con las reglas y la cabecera de formato MQRFH, y opcionalmente van seguidos de otros datos. El nombre de formato, el juego de caracteres y la codificación de los datos (si los hay) se proporcionan mediante los campos *Format*, *CodedCharSetId*y *Encoding* en MQRFH. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante MQFMT_RF_HEADER_ARRAY; tiene el mismo valor que MQFMT_RF_HEADER, pero es una matriz de caracteres en lugar de una serie.

MQFMT_RF_HEADER_2
Los datos del mensaje empiezan con las reglas version-2 y la cabecera de formato MQRFH2, y opcionalmente van seguidos de otros datos. El nombre de formato, el juego de caracteres y la codificación de los datos opcionales (si los hay) se proporcionan mediante los campos *Format*, *CodedCharSetId*y *Encoding* en MQRFH2. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante MQFMT_RF_HEADER_2_ARRAY ; tiene el mismo valor que MQFMT_RF_HEADER_2, pero es una matriz de caracteres en lugar de una serie.

MQFMT_STRING
Los datos del mensaje de aplicación pueden ser una serie SBCS (juego de caracteres de un solo byte) o una serie DBCS (juego de caracteres de doble byte). Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, también se define la constante MQFMT_STRING_ARRAY; tiene el mismo valor que MQFMT_STRING, pero es una matriz de caracteres en lugar de una serie.

MQFMT_TRIGGER
El mensaje es un mensaje desencadenante, descrito por la estructura MQTM; consulte "MQTM-Mensaje de desencadenante" en la página 577 para obtener detalles de esta estructura. Los mensajes de este formato se pueden convertir si se especifica la opción MQGMO_CONVERT en la llamada MQGET.

Para el lenguaje de programación C, la constante MQFMT_TRIGGER_ARRAY también está definida; tiene el mismo valor que MQFMT_TRIGGER, pero es una matriz de caracteres en lugar de una serie.

MQFMT_WORK_INFO_HEADER
Los datos del mensaje empiezan con la cabecera de información de trabajo MQWIH, que va seguida de los datos de la aplicación. El nombre de formato de los datos de aplicación se proporciona mediante el campo *Format* en la estructura MQWIH.

En z/OS, especifique la opción MQGMO_CONVERT en la llamada MQGET para convertir los *datos de usuario* en mensajes que tengan el formato MQFMT_WORK_INFO_HEADER. Sin embargo, la propia estructura MQWIH siempre se devuelve en el juego de caracteres y la codificación del gestor de colas (es decir, la estructura MQWIH se convierte tanto si se especifica la opción MQGMO_CONVERT como si no).

Para el lenguaje de programación C, también se define la constante MQFMT_WORK_INFO_HEADER_ARRAY; tiene el mismo valor que MQFMT_WORK_INFO_HEADER, pero es una matriz de caracteres en lugar de una serie.

MQFMT_XMIT_Q_HEADER
Los datos del mensaje empiezan por la cabecera de cola de transmisión MQXQH. Los datos del mensaje original siguen inmediatamente a la estructura MQXQH. El nombre de formato de los datos de mensaje originales lo proporciona el campo *Format* de la estructura MQMD, que forma parte de la cabecera de cola de transmisión MQXQH. Consulte "MQXQH-Cabecera de cola de transmisión" en la página 596 para obtener detalles de esta estructura.

Los informes COA y COD no se generan para los mensajes que tienen un *Format* de MQFMT_XMIT_Q_HEADER.

Para el lenguaje de programación C, también se define la constante MQFMT_XMIT_Q_HEADER_ARRAY; tiene el mismo valor que MQFMT_XMIT_Q_HEADER, pero es una matriz de caracteres en lugar de una serie.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . La longitud de este campo la proporciona MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

GroupId (MQBYTE24)

Es una serie de bytes que se utiliza para identificar el grupo de mensajes o el mensaje lógico al que pertenece el mensaje físico. *GroupId* también se utiliza si se permite la segmentación para el mensaje. En todos estos casos, *GroupId* tiene un valor no nulo y uno o varios de los distintivos siguientes se establecen en el campo *MsgFlags* :

- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- SEGMENTO MQMF_SEGMENT
- MQMF_LAST_SEGMENT
- MQMF_SEGMENTATION_ALLOWED

Si no se establece ninguno de estos distintivos, *GroupId* tiene el valor nulo especial MQGI_NONE.

La aplicación no necesita establecer este campo en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica MQPMO_LOGICAL_ORDER.

- En la llamada MQGET, MQMO_MATCH_GROUP_ID *no* se ha especificado.

Estas son las formas recomendadas de utilizar estas llamadas para mensajes que no son mensajes de informe. Sin embargo, si la aplicación requiere más control, o la llamada es MQPUT1, la aplicación debe asegurarse de que *GroupId* esté establecido en un valor adecuado.

Los grupos de mensajes y segmentos sólo se pueden procesar correctamente si el identificador de grupo es exclusivo. Por este motivo, las aplicaciones *no deben generar sus propios identificadores de grupo*; en su lugar, las aplicaciones deben realizar una de las acciones siguientes:

- Si se especifica MQPMO_LOGICAL_ORDER, el gestor de colas genera automáticamente un identificador de grupo exclusivo para el primer mensaje del grupo o segmento del mensaje lógico, y utiliza dicho identificador de grupo para los mensajes restantes del grupo o segmentos del mensaje lógico, por lo que la aplicación no necesita realizar ninguna acción especial. Este es el procedimiento recomendado.
- Si *no* se especifica MQPMO_LOGICAL_ORDER, la aplicación debe solicitar al gestor de colas que genere el identificador de grupo, estableciendo *GroupId* en MQGI_NONE en la primera llamada MQPUT o MQPUT1 para un mensaje del grupo o segmento del mensaje lógico. El identificador de grupo devuelto por el gestor de colas en la salida de esa llamada debe utilizarse entonces para los mensajes restantes en el grupo o segmentos del mensaje lógico. Si un grupo de mensajes contiene mensajes segmentados, se debe utilizar el mismo identificador de grupo para todos los segmentos y mensajes del grupo.

  Cuando no se especifica MQPMO_LOGICAL_ORDER, los mensajes de grupos y segmentos de mensajes lógicos se pueden poner en cualquier orden (por ejemplo, en orden inverso), pero el identificador de grupo debe ser asignado por la llamada *first* MQPUT o MQPUT1 que se emite para cualquiera de estos mensajes.

En la entrada a las llamadas MQPUT y MQPUT1 , el gestor de colas utiliza el valor descrito en Orden físico en una cola. En la salida de las llamadas MQPUT y MQPUT1 , el gestor de colas establece este campo en el valor que se ha enviado con el mensaje si el objeto abierto es una sola cola y no una lista de distribución, pero lo deja sin modificar si el objeto abierto es una lista de distribución. En este último caso, si la aplicación necesita conocer los identificadores de grupo generados, la aplicación debe proporcionar registros MQPMR que contengan el campo *GroupId* .

En la entrada a la llamada MQGET, el gestor de colas utiliza el valor descrito en Tabla 506 en la página 364. En la salida de la llamada MQGET, el gestor de colas establece este campo en el valor del mensaje recuperado.

Se define el siguiente valor especial:

MQGI_NONE
No se ha especificado ningún identificador de grupo.

El valor es cero binario para la longitud del campo. Es el valor que se utiliza para los mensajes que no están en grupos, no en segmentos de mensajes lógicos y para los que no se permite la segmentación.

Para el lenguaje de programación C, la constante MQGI_NONE_ARRAY también está definida; tiene el mismo valor que MQGI_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_GROUP_ID_LENGTH. El valor inicial de este campo es MQGI_NONE. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

MsgFlags (MQLONG)
Los MsgFlags son distintivos que especifican atributos del mensaje o controlan su proceso.

Los MsgFlags se dividen en las categorías siguientes:

- Distintivos de segmentación
- Distintivos de estado

Distintivos de segmentación: cuando un mensaje es demasiado grande para una cola, normalmente falla un intento de colocar el mensaje en la cola. La segmentación es una técnica mediante la cual el gestor de colas o la aplicación divide el mensaje en partes más pequeñas denominadas segmentos, y coloca cada segmento en la cola como un mensaje físico independiente. La aplicación que recupera el mensaje

puede recuperar los segmentos uno por uno o solicitar al gestor de colas que vuelva a ensamblar los segmentos en un único mensaje devuelto por la llamada MQGET. Esto último se consigue especificando la opción MQGMO_COMPLETE_MSG en la llamada MQGET y proporcionando un almacenamiento intermedio lo suficientemente grande como para acomodar el mensaje completo. (Consulte "MQGMO-Opciones de obtención de mensajes" en la página 344 para obtener detalles de la opción MQGMO_COMPLETE_MSG.) Un mensaje se puede segmentar en el gestor de colas emisor, en un gestor de colas intermedio o en el gestor de colas de destino.

Puede especificar una de las siguientes opciones para controlar la segmentación de un mensaje:

MQMF_SEGMENTATION_INHIBIDO
Esta opción impide que el gestor de colas divida el mensaje en segmentos. Si se especifica para un mensaje que ya es un segmento, esta opción impide que el segmento se divida en segmentos más pequeños.

El valor de este distintivo es cero binario. Éste es el valor predeterminado.

MQMF_SEGMENTATION_ALLOWED
Esta opción permite que el gestor de colas divida el mensaje en segmentos. Si se especifica para un mensaje que ya es un segmento, esta opción permite que el segmento se divida en segmentos más pequeños. MQMF_SEGMENTATION_ALLOWED se puede establecer sin establecer MQMF_SEGMENT o MQMF_LAST_SEGMENT.

- En z/OS, el gestor de colas no da soporte a la segmentación de mensajes. Si un mensaje es demasiado grande para la cola, la llamada MQPUT o MQPUT1 falla con el código de razón MQRC_MSG_TOO_BIG_FOR_Q. Sin embargo, todavía se puede especificar la opción MQMF_SEGMENTATION_ALLOWED y permite segmentar el mensaje en un gestor de colas remoto.

Cuando el gestor de colas segmenta un mensaje, el gestor de colas activa el distintivo MQMF_SEGMENT en la copia del MQMD que se envía con cada segmento, pero no altera los valores de estos distintivos en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1 . Para el último segmento del mensaje lógico, el gestor de colas también activa el distintivo MQMF_LAST_SEGMENT en el MQMD que se envía con el segmento.

Nota: Tenga cuidado al colocar mensajes con MQMF_SEGMENTATION_ALLOWED pero sin MQPMO_LOGICAL_ORDER. Si el mensaje es:

- No es un segmento, y
- No en un grupo, y
- No se reenvía,

la aplicación debe restablecer el campo *GroupId* en MQGI_NONE antes de *cada llamada* MQPUT o MQPUT1 , para que el gestor de colas pueda generar un identificador de grupo exclusivo para cada mensaje. Si esto no se hace, los mensajes no relacionados pueden tener el mismo identificador de grupo, lo que puede llevar a un proceso incorrecto posteriormente. Consulte las descripciones del campo *GroupId* y la opción MQPMO_LOGICAL_ORDER para obtener más información sobre cuándo restablecer el campo *GroupId* .

El gestor de colas divide los mensajes en segmentos según sea necesario para que los segmentos (más los datos de cabecera necesarios) quepan en la cola. Sin embargo, existe un límite inferior para el tamaño de un segmento generado por el gestor de colas, y sólo el último segmento creado a partir de un mensaje puede ser menor que este límite (el límite inferior para el tamaño de un segmento generado por la aplicación es de un byte). Los segmentos generados por el gestor de colas pueden tener una longitud desigual. El gestor de colas procesa el mensaje de la forma siguiente:

- Los formatos definidos por el usuario se dividen en límites que son múltiplos de 16 bytes; el gestor de colas no genera segmentos que sean menores de 16 bytes (que no sean el último segmento).
- Los formatos incorporados distintos de MQFMT_STRING se dividen en puntos adecuados a la naturaleza de los datos presentes. Sin embargo, el gestor de colas nunca divide un mensaje en medio de una estructura de cabecera de WebSphere MQ . Esto significa que el gestor de colas no puede dividir más un segmento que contenga una única estructura de cabecera MQ y, como resultado, el tamaño de segmento mínimo posible para ese mensaje es mayor que 16 bytes.

El segundo segmento o segmento posterior generado por el gestor de colas empieza por uno de los siguientes:

  - Una estructura de cabecera MQ
  - El inicio de los datos del mensaje de aplicación
  - Parte del camino a través de los datos del mensaje de aplicación

- MQFMT_STRING se divide sin tener en cuenta la naturaleza de los datos presentes (SBCS, DBCS o SBCS/DBCS mixto). Cuando la serie es DBCS o SBCS/DBCS mixto, esto puede dar como resultado segmentos que no se pueden convertir de un juego de caracteres a otro. El gestor de colas nunca divide los mensajes MQFMT_STRING en segmentos menores de 16 bytes (que no sean el último segmento).
- El gestor de colas establece los campos *Format*, *CodedCharSetId*y *Encoding* en el MQMD de cada segmento para describir correctamente los datos presentes en el *inicio* del segmento; el nombre de formato es el nombre de un formato incorporado o el nombre de un formato definido por el usuario.
- Se modifica el campo *Report* en el MQMD de segmentos con *Offset* mayor que cero. Para cada tipo de informe, si la opción de informe es MQRO_ * _WITH_DATA, pero el segmento no puede contener ninguno de los primeros 100 bytes de datos de usuario (es decir, los datos que siguen a las estructuras de cabecera de WebSphere MQ que pueden estar presentes), la opción de informe se cambia a MQRO_ *.

El gestor de colas sigue las reglas anteriores, pero de lo contrario divide los mensajes de forma imprevisible; no realice suposiciones sobre dónde se divide un mensaje.

Para los mensajes *persistentes* , el gestor de colas sólo puede realizar la segmentación dentro de una unidad de trabajo:

- Si la llamada MQPUT o MQPUT1 está funcionando dentro de una unidad de trabajo definida por el usuario, se utiliza dicha unidad de trabajo. Si la llamada falla durante el proceso de segmentación, el gestor de colas elimina los segmentos que se colocaron en la cola como resultado de la llamada anómala. Sin embargo, la anomalía no impide que la unidad de trabajo se confirme correctamente.
- Si la llamada está funcionando fuera de una unidad de trabajo definida por el usuario y no existe ninguna unidad de trabajo definida por el usuario, el gestor de colas crea una unidad de trabajo sólo para la duración de la llamada. Si la llamada es satisfactoria, el gestor de colas confirma la unidad de trabajo automáticamente. Si la llamada falla, el gestor de colas restituye la unidad de trabajo.
- Si la llamada está funcionando fuera de una unidad de trabajo definida por el usuario, pero existe una unidad de trabajo definida por el usuario, el gestor de colas no puede realizar la segmentación. Si el mensaje no requiere segmentación, la llamada puede seguir siendo satisfactoria. Pero si el mensaje requiere segmentación, la llamada falla con el código de razón MQRC_UOW_NOT_AVAILABLE.

Para los mensajes *no persistentes* , el gestor de colas no requiere que haya una unidad de trabajo disponible para realizar la segmentación.

Tenga especial cuidado al convertir datos en mensajes que se pueden segmentar:

- Si la aplicación receptora convierte datos en la llamada MQGET y especifica la opción MQGMO_COMPLETE_MSG, a la salida de conversión de datos se le pasa el mensaje completo para que la salida lo convierta y el hecho de que el mensaje se haya segmentado es evidente para la salida.
- Si la aplicación receptora recupera un segmento a la vez, se invoca la salida de conversión de datos para convertir un segmento a la vez. Por lo tanto, la salida debe convertir los datos de un segmento independientemente de los datos de cualquiera de los otros segmentos.

  Si la naturaleza de los datos del mensaje es tal que la segmentación arbitraria de los datos en límites de 16 bytes puede dar como resultado segmentos que la salida no puede convertir, o el formato es MQFMT_STRING y el juego de caracteres es DBCS o SBCS/DBCS mixto, la aplicación emisora debe crear y colocar los segmentos, especificando MQMF_SEGMENTATION_INHIBIDO para suprimir la segmentación adicional. De esta forma, la aplicación emisora puede asegurarse de que

cada segmento contiene información suficiente para permitir que la salida de conversión de datos convierta el segmento correctamente.

- Si se especifica la conversión del emisor para un agente de canal de mensajes (MCA) de envío, el MCA sólo convierte los mensajes que no son segmentos de mensajes lógicos; el MCA nunca intenta convertir los mensajes que son segmentos.

Este distintivo es un distintivo de entrada en las llamadas MQPUT y MQPUT1 y un distintivo de salida en la llamada MQGET. En la última llamada, el gestor de colas también repite el valor del distintivo en el campo *Segmentation* en MQGMO.

El valor inicial de este distintivo es MQMF_SEGMENTATION_INHIBIDO.

Distintivos de estado: son distintivos que indican si el mensaje físico pertenece a un grupo de mensajes, es un segmento de un mensaje lógico, ambos o ninguno. Se pueden especificar uno o más de los siguientes valores en la llamada MQPUT o MQPUT1 , o la llamada MQGET los puede devolver:

MQMF_MSG_IN_GROUP
El mensaje es miembro de un grupo.

MQMF_LAST_MSG_IN_GROUP
El mensaje es el último mensaje lógico de un grupo.

Si se establece este distintivo, el gestor de colas activa MQMF_MSG_IN_GROUP en la copia de MQMD que se envía con el mensaje, pero no altera los valores de estos distintivos en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1 .

Es válido que un grupo conste de un solo mensaje lógico. Si este es el caso, se establece MQMF_LAST_MSG_IN_GROUP, pero el campo *MsgSeqNumber* tiene el valor uno.

SEGMENTO MQMF_SEGMENT
El mensaje es un segmento de un mensaje lógico.

Cuando se especifica MQMF_SEGMENT sin MQMF_LAST_SEGMENT, la longitud de los datos de mensaje de aplicación en el segmento (*excluyendo* las longitudes de las estructuras de cabecera de WebSphere MQ que pueden estar presentes) debe ser como mínimo una. Si la longitud es cero, la llamada MQPUT o MQPUT1 falla con el código de razón MQRC_SEGMENT_LENGTH_ZERO.

En z/OS, esta opción no está soportada si el mensaje se está colocando en una cola que tiene un tipo de índice de MQIT_GROUP_ID.

MQMF_LAST_SEGMENT
El mensaje es el último segmento de un mensaje lógico.

Si se establece este distintivo, el gestor de colas activa MQMF_SEGMENT en la copia de MQMD que se envía con el mensaje, pero no altera los valores de estos distintivos en el MQMD proporcionado por la aplicación en la llamada MQPUT o MQPUT1 .

Un mensaje lógico sólo puede constar de un segmento. Si es así, se establece MQMF_LAST_SEGMENT, pero el campo *Offset* tiene el valor cero.

Cuando se especifica MQMF_LAST_SEGMENT, la longitud de los datos del mensaje de aplicación en el segmento (*excluyendo* las longitudes de las estructuras de cabecera que pueden estar presentes) puede ser cero.

En z/OS, esta opción no está soportada si el mensaje se está colocando en una cola que tiene un tipo de índice de MQIT_GROUP_ID.

La aplicación debe asegurarse de que estos distintivos se establecen correctamente al transferir mensajes. Si se especifica MQPMO_LOGICAL_ORDER, o se ha especificado en la llamada MQPUT anterior para el descriptor de contexto de cola, los valores de los distintivos deben ser coherentes con la información de grupo y segmento retenida por el gestor de colas para el descriptor de contexto de cola. Las condiciones siguientes se aplican a las llamadas MQPUT *sucesivas* para el descriptor de contexto de cola cuando se especifica MQPMO_LOGICAL_ORDER:

- Si no hay ningún grupo o mensaje lógico actual, todos estos distintivos (y combinaciones de ellos) son válidos.

- Una vez que se ha especificado MQMF_MSG_IN_GROUP, debe permanecer activado hasta que se especifique MQMF_LAST_MSG_IN_GROUP. La llamada falla con el código de razón MQRC_INCOMPLETE_GROUP si no se cumple esta condición.
- Una vez que se ha especificado MQMF_SEGMENT, debe permanecer activado hasta que se especifique MQMF_LAST_SEGMENT. La llamada falla con el código de razón MQRC_INCOMPLETE_MSG si no se cumple esta condición.
- Una vez que se ha especificado MQMF_SEGMENT sin MQMF_MSG_IN_GROUP, MQMF_MSG_IN_GROUP debe permanecer *desactivado* hasta que se haya especificado MQMF_LAST_SEGMENT. La llamada falla con el código de razón MQRC_INCOMPLETE_MSG si no se cumple esta condición.

Orden físico en una cola muestra las combinaciones válidas de los distintivos y los valores utilizados para diversos campos.

Estos distintivos son distintivos de entrada en las llamadas MQPUT y MQPUT1 y distintivos de salida en la llamada MQGET. En la última llamada, el gestor de colas también repite los valores de los distintivos en los campos *GroupStatus* y *SegmentStatus* en MQGMO.

No puede utilizar mensajes agrupados o segmentados con la publicación/suscripción.

Distintivos predeterminados: se puede especificar lo siguiente para indicar que el mensaje tiene atributos predeterminados:

MQMF_NONE
Sin distintivos de mensaje (atributos de mensaje predeterminados).

Esto inhibe la segmentación e indica que el mensaje no está en un grupo y no es un segmento de un mensaje lógico. MQMF_NONE está definido para ayudar a la documentación del programa. No se pretende que este distintivo se utilice con ningún otro, pero como su valor es cero, no se puede detectar dicho uso.

El campo *MsgFlags* se particiona en subcampos; para obtener detalles, consulte “Opciones de informe y distintivos de mensaje” en la página 884.

El valor inicial de este campo es MQMF_NONE. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

MsgId (MQBYTE24)

Es una serie de bytes que se utiliza para distinguir un mensaje de otro. Generalmente, ningún mensaje debe tener el mismo identificador de mensaje, aunque el gestor de colas no lo permita. El identificador de mensaje es una propiedad permanente del mensaje y persiste en los reinicios del gestor de colas. Puesto que el identificador de mensaje es una serie de bytes y no una serie de caracteres, el identificador de mensaje *no* se convierte entre juegos de caracteres cuando el mensaje fluye de un gestor de colas a otro.

Para las llamadas MQPUT y MQPUT1 , si la aplicación especifica MQMI_NONE o MQPMO_NEW_MSG_ID, el gestor de colas genera un identificador de mensaje exclusivo[2]cuando se transfiera el mensaje y lo coloque en el descriptor de mensaje enviado con el mensaje. El gestor de colas también devuelve este identificador de mensaje en el descriptor de mensaje que pertenece a la aplicación emisora. La aplicación puede utilizar este valor para registrar información sobre mensajes concretos y para responder a consultas de otras partes de la aplicación.

[2] Un *MsgId* generado por el gestor de colas consta de un identificador de producto de 4 bytes (AMQ ¬ o CSQ ¬ in ASCII o EBCDIC, donde ¬ representa un carácter en blanco), seguido de una implementación específica del producto de una serie exclusiva. En WebSphere MQ contiene los primeros 12 caracteres del nombre del gestor de colas y un valor derivado del reloj del sistema. Por lo tanto, todos los gestores de colas que pueden intercomunicarse deben tener nombres que difieran en los primeros 12 caracteres, para asegurarse de que los identificadores de mensaje sean exclusivos. La capacidad de generar una serie exclusiva también depende de que el reloj del sistema no se cambie hacia atrás. Para eliminar la posibilidad de que un identificador de mensaje generado por el gestor de colas duplique uno generado por la aplicación, la aplicación debe evitar generar identificadores con caracteres iniciales en el rango de A a I en ASCII o EBCDIC (de X'41 'a X'49' y de X'C1'a X'C9'). Sin embargo, no se impide que la aplicación genere identificadores con caracteres iniciales en estos rangos.

Si el mensaje se está colocando en un tema, el gestor de colas genera identificadores de mensaje exclusivos según sea necesario para cada mensaje publicado. Si la aplicación especifica MQPMO_NEW_MSG_ID, el gestor de colas genera un identificador de mensaje exclusivo para devolver en la salida. Si la aplicación especifica MQMI_NONE, el valor del campo *MsgId* en MQMD no se modifica al devolver la llamada.

Consulte la descripción de MQPMO_RETAIN en "Opciones de MQPMO (MQLONG)" en la página 482 para obtener más detalles sobre las publicaciones retenidas.

Si el mensaje se está colocando en una lista de distribución, el gestor de colas genera identificadores de mensaje exclusivos según sea necesario, pero el valor del campo *MsgId* en MQMD no se modifica al devolver la llamada, incluso si se ha especificado MQMI_NONE o MQPMO_NEW_MSG_ID. Si la aplicación necesita conocer los identificadores de mensaje generados por el gestor de colas, la aplicación debe proporcionar registros MQPMR que contengan el campo *MsgId* .

La aplicación emisora también puede especificar un valor para el identificador de mensaje que no sea MQMI_NONE; esto detiene el gestor de colas que genera un identificador de mensaje exclusivo. Una aplicación que reenvía un mensaje puede utilizarlo para propagar el identificador de mensaje del mensaje original.

El gestor de colas no utiliza este campo excepto para:

- Generar un valor exclusivo si se solicita, tal como se ha descrito anteriormente
- Entregue el valor a la aplicación que emite la solicitud de obtención para el mensaje
- Copie el valor en el campo *CorrelId* de cualquier mensaje de informe que genere sobre este mensaje (en función de las opciones de *Report*)

Cuando el gestor de colas o un agente de canal de mensajes genera un mensaje de informe, establece el campo *MsgId* de la forma especificada por el campo *Report* del mensaje original, MQRO_NEW_MSG_ID o MQRO_PASS_MSG_ID. Las aplicaciones que generan mensajes de informe también deben hacerlo.

Para la llamada MQGET, *MsgId* es uno de los cinco campos que se pueden utilizar para recuperar un mensaje determinado de la cola. Normalmente, la llamada MQGET devuelve el siguiente mensaje en la cola, pero se puede obtener un mensaje determinado especificando uno o más de los cinco criterios de selección, en cualquier combinación; estos campos son:

- *MsgId*
- *CorrelId*
- *GroupId*
- *MsgSeqNumber*
- *Offset*

La aplicación establece uno o más de estos campos en los valores necesarios y, a continuación, establece las opciones de coincidencia MQMO_ * correspondientes en el campo *MatchOptions* en MQGMO para utilizar estos campos como criterios de selección. Sólo los mensajes que tienen los valores especificados en esos campos son candidatos para la recuperación. El valor predeterminado para el campo *MatchOptions* (si no lo modifica la aplicación) es que coincida con el identificador de mensaje y el identificador de correlación.

En z/OS, los criterios de selección que puede utilizar están restringidos por el tipo de índice utilizado para la cola. Consulte el atributo de cola *IndexType* para obtener más detalles.

Normalmente, el mensaje devuelto es el *primer* mensaje de la cola que cumple los criterios de selección. Pero si se especifica MQGMO_BROWSE_NEXT, el mensaje devuelto es el *siguiente* mensaje que satisface los criterios de selección; la exploración de este mensaje se inicia con el mensaje *siguiente* la posición actual del cursor.

Nota: La cola se explora secuencialmente en busca de un mensaje que satisfaga los criterios de selección, por lo que los tiempos de recuperación son más lentos que si no se especifica ningún criterio de selección, especialmente si se tienen que explorar muchos mensajes antes de que se encuentre uno adecuado. Las excepciones a esto son:

- una llamada MQGET de *CorrelId* en plataformas distribuidas de 64 bits donde el índice *CorrelId* elimina la necesidad de realizar una exploración secuencial verdadera.
- una llamada MQGET de *IndexType* en z/OS.

En ambos casos, se mejora el rendimiento de recuperación.

Consulte Tabla 506 en la página 364 para obtener más información sobre cómo se utilizan los criterios de selección en diversas situaciones.

Especificar MQMI_NONE como identificador de mensaje tiene el mismo efecto que *no* especificar MQMO_MATCH_MSG_ID, es decir, *cualquier* coincidencia de identificador de mensaje.

Este campo se ignora si se especifica la opción MQGMO_MSG_UNDER_CURSOR en el parámetro *GetMsgOpts* de la llamada MQGET.

Al volver de una llamada MQGET, el campo *MsgId* se establece en el identificador de mensaje del mensaje devuelto (si lo hay).

Se puede utilizar el siguiente valor especial:

MQMI_NONE
No se ha especificado ningún identificador de mensaje.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, la constante MQMI_NONE_ARRAY también está definida; tiene el mismo valor que MQMI_NONE, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada/salida para las llamadas MQGET, MQPUT y MQPUT1 . La longitud de este campo la proporciona MQ_MSG_ID_LENGTH. El valor inicial de este campo es MQMI_NONE.

MsgSeqNumber (MQLONG)
Es el número de secuencia de un mensaje lógico dentro de un grupo.

Los números de secuencia empiezan en 1 y aumentan en 1 para cada nuevo mensaje lógico del grupo, hasta un máximo de 999 999 999. Un mensaje físico que no está en un grupo tiene un número de secuencia de 1.

La aplicación no tiene que establecer este campo en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica MQPMO_LOGICAL_ORDER.
- En la llamada MQGET, *no* se ha especificado MQMO_MATCH_MSG_SEQ_NUMBER.

Estas son las formas recomendadas de utilizar estas llamadas para mensajes que no son mensajes de informe. Sin embargo, si la aplicación requiere más control, o la llamada es MQPUT1, la aplicación debe asegurarse de que *MsgSeqNumber* esté establecido en un valor adecuado.

En la entrada a las llamadas MQPUT y MQPUT1 , el gestor de colas utiliza el valor descrito en Orden físico en una cola. En la salida de las llamadas MQPUT y MQPUT1 , el gestor de colas establece este campo en el valor que se ha enviado con el mensaje.

En la entrada de la llamada MQGET, el gestor de colas utiliza el valor que se muestra en Tabla 506 en la página 364. En la salida de la llamada MQGET, el gestor de colas establece este campo en el valor del mensaje recuperado.

El valor inicial de este campo es uno. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

MsgType (MQLONG)

Indica el tipo del mensaje. Los tipos de mensaje se agrupan de la forma siguiente:

MQMT_SYSTEM_FIRST
Valor más bajo para tipos de mensajes definidos por el sistema.

MQMT_SYSTEM_LAST
Valor más alto para tipos de mensajes definidos por el sistema.

Los valores siguientes están definidos actualmente dentro del rango del sistema:

MQMT_DATAGRAM
El mensaje es uno que no requiere una respuesta.

MQMT_REQUEST
El mensaje es uno que requiere una respuesta.

Especifique el nombre de la cola a la que enviar la respuesta en el campo *ReplyToQ* . El campo *Report* indica cómo establecer *MsgId* y *CorrelId* de la respuesta.

MQMT_REPLY
El mensaje es la respuesta a un mensaje de solicitud anterior (MQMT_REQUEST). El mensaje debe enviarse a la cola indicada por el campo *ReplyToQ* del mensaje de solicitud. Utilice el campo *Report* de la solicitud para controlar cómo establecer *MsgId* y *CorrelId* de la respuesta.

Nota: El gestor de colas no impone la relación de solicitud-respuesta; esto es una responsabilidad de la aplicación.

MQMT_REPORT
El mensaje está informando sobre alguna aparición esperada o inesperada, normalmente relacionada con algún otro mensaje (por ejemplo, se ha recibido un mensaje de solicitud que contenía datos que no eran válidos). Envíe el mensaje a la cola indicada por el campo *ReplyToQ* del descriptor de mensaje del mensaje original. Establezca los campos *Feedback* para indicar la naturaleza del informe. Utilice el campo *Report* del mensaje original para controlar cómo establecer *MsgId* y *CorrelId* del mensaje de informe.

Los mensajes de informe generados por el gestor de colas o el agente de canal de mensajes siempre se envían a la cola *ReplyToQ* , con los campos *Feedback* y *CorrelId* establecidos como se ha descrito anteriormente.

También se pueden utilizar valores definidos por la aplicación. Deben estar dentro del rango siguiente:

MQMT_APPL_PRIMERO
Valor más bajo para tipos de mensajes definidos por la aplicación.

MQMT_APPL_LAST
Valor más alto para tipos de mensajes definidos por la aplicación.

Para las llamadas MQPUT y MQPUT1 , el valor *MsgType* debe estar dentro del rango definido por el sistema o del rango definido por la aplicación; si no lo está, la llamada falla con el código de razón MQRC_MSG_TYPE_ERROR.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQMT_DATAGRAM.

Offset (MQLONG)

Es el desplazamiento en bytes de los datos del mensaje físico desde el inicio del mensaje lógico del que forman parte los datos. Estos datos se denominan *segmento*. El desplazamiento está en el rango de 0 a 999.999.999. Un mensaje físico que no es un segmento de un mensaje lógico tiene un desplazamiento de cero.

La aplicación no necesita establecer este campo en la llamada MQPUT o MQGET si:

- En la llamada MQPUT, se especifica MQPMO_LOGICAL_ORDER.
- En la llamada MQGET, *no* se especifica MQMO_MATCH_OFFSET.

Estas son las formas recomendadas de utilizar estas llamadas para mensajes que no son mensajes de informe. Sin embargo, si la aplicación no cumple estas condiciones, o la llamada es MQPUT1, la aplicación debe asegurarse de que *Offset* esté establecido en un valor adecuado.

En la entrada a las llamadas MQPUT y MQPUT1 , el gestor de colas utiliza el valor descrito en Orden físico en una cola. En la salida de las llamadas MQPUT y MQPUT1 , el gestor de colas establece este campo en el valor que se ha enviado con el mensaje.

Para un mensaje de informe que informa sobre un segmento de un mensaje lógico, el campo *OriginalLength* (siempre que no sea MQOL_UNDEFINED) se utiliza para actualizar el desplazamiento en la información de segmento retenida por el gestor de colas.

En la entrada de la llamada MQGET, el gestor de colas utiliza el valor que se muestra en Tabla 506 en la página 364. En la salida de la llamada MQGET, el gestor de colas establece este campo en el valor del mensaje recuperado.

El valor inicial de este campo es cero. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

OriginalLength (MQLONG)

Este campo sólo es relevante para los mensajes de informe que son segmentos. Especifica la longitud del segmento de mensaje con el que se relaciona el mensaje de informe; no especifica la longitud del mensaje lógico del que forma parte el segmento o la longitud de los datos del mensaje de informe.

Nota: Al generar un mensaje de informe para un mensaje que es un segmento, el gestor de colas y el agente de canal de mensajes copian en el MQMD del mensaje de informe los campos *GroupId*, *MsgSeqNumber*, *Offset*y *MsgFlags*, del mensaje original. Como resultado, el mensaje de informe también es un segmento. Las aplicaciones que generan mensajes de informe deben hacer lo mismo y establecer el campo *OriginalLength* correctamente.

Se define el siguiente valor especial:

MQOL_UNDEFINED
No se ha definido la longitud original del mensaje.

OriginalLength es un campo de entrada en las llamadas MQPUT y MQPUT1 , pero el valor que proporciona la aplicación sólo se acepta en determinadas circunstancias:

- Si el mensaje que se está colocando es un segmento y también es un mensaje de informe, el gestor de colas acepta el valor especificado. El valor debe ser:
  - Mayor que cero si el segmento no es el último segmento
  - No menor que cero si el segmento es el último segmento
  - No menor que la longitud de los datos presentes en el mensaje

  Si no se cumplen estas condiciones, la llamada falla con el código de razón MQRC_ORIGINAL_LENGTH_ERROR.
- Si el mensaje que se está colocando es un segmento pero no un mensaje de informe, el gestor de colas ignora el campo y utiliza en su lugar la longitud de los datos del mensaje de aplicación.
- En todos los demás casos, el gestor de colas ignora el campo y utiliza el valor MQOL_UNDEFINED en su lugar.

Es un campo de salida en la llamada MQGET.

El valor inicial de este campo es MQOL_UNDEFINED. Este campo se ignora si *Version* es menor que MQMD_VERSION_2.

Persistence (MQLONG)

Esto indica si el mensaje sobrevive a las anomalías del sistema y a los reinicios del gestor de colas. Para las llamadas MQPUT y MQPUT1 , el valor debe ser uno de los siguientes:

MQPER_PERSISTENT
El mensaje sobrevive a las anomalías del sistema y a los reinicios del gestor de colas. Una vez que se ha colocado el mensaje y se ha confirmado la unidad de trabajo en la que se ha colocado (si el mensaje se coloca como parte de una unidad de trabajo), el mensaje se conserva en el almacenamiento auxiliar. Permanece allí hasta que el mensaje se elimina de la cola y la unidad de trabajo en la que se obtuvo se ha confirmado (si el mensaje se recupera como parte de una unidad de trabajo).

Cuando se envía un mensaje persistente a una cola remota, un mecanismo de almacenamiento y reenvío mantiene el mensaje en cada gestor de colas a lo largo de la ruta al destino, hasta que se sepa que el mensaje ha llegado al siguiente gestor de colas.

Los mensajes persistentes no se pueden colocar en:

- Colas dinámicas temporales
- Colas compartidas que se correlacionan con un objeto CFSTRUCT en CFLEVEL (2) o inferior, o donde el objeto CFSTRUCT se define como RECOVER (NO).

Los mensajes persistentes se pueden colocar en colas dinámicas permanentes y colas predefinidas.

MQPER_NOT_PERSISTENT
El mensaje no suele sobrevivir a anomalías del sistema o a reinicios del gestor de colas. Esto se aplica incluso si se encuentra una copia intacta del mensaje en el almacenamiento auxiliar cuando se reinicia el gestor de colas.

En el caso de las colas NPMCLASS (HIGH), los mensajes no persistentes sobreviven a una conclusión y reinicio normales del gestor de colas.

En el caso de las colas compartidas, los mensajes no persistentes sobreviven a los reinicios del gestor de colas en el grupo de compartición de colas, pero no sobreviven a las anomalías del recurso de acoplamiento utilizado para almacenar mensajes en las colas compartidas.

MQPER_PERSISTENCE_AS_Q_DEF

- Si la cola es una cola de clúster, la persistencia del mensaje se toma del atributo *DefPersistence* definido en el gestor de colas de *destino* que es propietario de la instancia concreta de la cola en la que se coloca el mensaje. Normalmente, todas las instancias de una cola de clúster tienen el mismo valor para el atributo *DefPersistence* , aunque esto no es obligatorio.

  El valor de *DefPersistence* se copia en el campo *Persistence* cuando el mensaje se coloca en la cola de destino. Si posteriormente se cambia *DefPersistence* , los mensajes que ya se han colocado en la cola no se verán afectados.
- Si la cola no es una cola de clúster, la persistencia del mensaje se toma del atributo *DefPersistence* definido en el gestor de colas *local* , incluso si el gestor de colas de destino es remoto.

  Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la persistencia predeterminada se toma del valor de este atributo en la *primera* definición de la vía de acceso. Este puede ser:

  – una cola alias
  – Una cola local
  – Una definición local de una cola remota
  – Un alias de gestor de colas
  – Una cola de transmisión (por ejemplo, la cola *DefXmitQName*)

  El valor de *DefPersistence* se copia en el campo *Persistence* cuando se coloca el mensaje. Si posteriormente se cambia *DefPersistence* , los mensajes que ya se han colocado no se verán afectados.

Los mensajes persistentes y no persistentes pueden existir en la misma cola.

Al responder a un mensaje, las aplicaciones deben utilizar la persistencia del mensaje de solicitud para el mensaje de respuesta.

Para una llamada MQGET, el valor devuelto es MQPER_PERSISTENT o MQPER_NOT_PERSISTENT.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQPER_PERSISTENCE_AS_Q_DEF.

Priority (MQLONG)

Para las llamadas MQPUT y MQPUT1 , el valor debe ser mayor o igual que cero; cero es la prioridad más baja. También se puede utilizar el siguiente valor especial:

MQPRI_PRIORITY_AS_Q_DEF

- Si la cola es una cola de clúster, la prioridad del mensaje se toma del atributo *DefPriority* tal como se define en el gestor de colas de *destino* que es propietario de la instancia concreta de la cola en la que se coloca el mensaje. Normalmente, todas las instancias de una cola de clúster tienen el mismo valor para el atributo *DefPriority* , aunque esto no es obligatorio.

  El valor de *DefPriority* se copia en el campo *Priority* cuando el mensaje se coloca en la cola de destino. Si posteriormente se cambia *DefPriority* , los mensajes que ya se han colocado en la cola no se verán afectados.
- Si la cola no es una cola de clúster, la prioridad del mensaje se toma del atributo *DefPriority* tal como se define en el gestor de colas *local* , incluso si el gestor de colas de destino es remoto.

  Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la prioridad predeterminada se toma del valor de este atributo en la *primera* definición de la vía de acceso. Este puede ser:

  – una cola alias
  – Una cola local
  – Una definición local de una cola remota
  – Un alias de gestor de colas
  – Una cola de transmisión (por ejemplo, la cola *DefXmitQName*)

  El valor de *DefPriority* se copia en el campo *Priority* cuando se coloca el mensaje. Si posteriormente se cambia *DefPriority* , los mensajes que ya se han colocado no se verán afectados.

El valor devuelto por la llamada MQGET es siempre mayor o igual que cero; el valor MQPRI_PRIORITY_AS_Q_DEF nunca se devuelve.

Si un mensaje se coloca con una prioridad mayor que el máximo soportado por el gestor de colas local (este máximo lo proporciona el atributo de gestor de colas *MaxPriority*), el mensaje lo acepta el gestor de colas, pero se coloca en la cola con la prioridad máxima del gestor de colas; la llamada MQPUT o MQPUT1 se completa con MQCC_WARNING y el código de razón MQRC_PRIORITY_EXCEEDS_MAXIMUM. Sin embargo, el campo *Priority* conserva el valor especificado por la aplicación que ha colocado el mensaje.

En z/OS, si un mensaje con un número MsgSeqde 1 se coloca en una cola que tiene una secuencia de entrega de mensajes de MQMDS_PRIORITY y un tipo de índice de MQIT_GROUP_ID, la cola podría tratar el mensaje con una prioridad diferente. Si el mensaje se ha colocado en la cola con una prioridad de 0 o 1, se procesa como si tuviera una prioridad de 2. Esto se debe a que el orden de los mensajes colocados en este tipo de cola se ha optimizado para habilitar las pruebas de compleción de grupo eficientes. Para obtener más información sobre la secuencia de entrega de mensajes MQMDS_PRIORITY y el tipo de índice MQIT_GROUP_ID, consulte Atributo de secuenciaMsgDelivery.

Al responder a un mensaje, las aplicaciones deben utilizar la prioridad del mensaje de solicitud para el mensaje de respuesta. En otras situaciones, especificar MQPRI_PRIORITY_AS_Q_DEF permite que el ajuste de prioridad se lleve a cabo sin cambiar la aplicación.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQPRI_PRIORITY_AS_Q_DEF.

PutApplNombre (MQCHAR28)
Es el nombre de la aplicación que ha colocado el mensaje y forma parte del *contexto de origen* del mensaje. El contenido difiere entre plataformas y puede diferir entre releases.

Para obtener más información sobre el contexto de mensaje, consulte “Visión general de MQMD” en la página 394 y Contexto de mensaje .

El formato de *PutApplName* depende del valor de *PutApplType* y puede cambiar de un release a otro. Los cambios son raros, pero ocurren si el entorno cambia.

Cuando el gestor de colas establece este campo (es decir, para todas las opciones excepto MQPMO_SET_ALL_CONTEXT), establece el campo en un valor determinado por el entorno:

- En z/OS, el gestor de colas utiliza:
  - Para el proceso por lotes z/OS , el nombre de trabajo de 8 caracteres de la tarjeta JES JOB
  - Para TSO, el identificador de usuario TSO de 7 caracteres
  - Para CICS, el ID de aplicación de 8 caracteres, seguido del ID de aplicación de 4 caracteres
  - Para IMS, el identificador del sistema IMS de 8 caracteres, seguido del nombre de PSB de 8 caracteres
  - Para XCF, el nombre de grupo XCF de 8 caracteres, seguido del nombre de miembro XCF de 16 caracteres
  - Para un mensaje generado por un gestor de colas, los primeros 28 caracteres del nombre del gestor de colas
  - Para colas distribuidas sin CICS, el nombre de trabajo de 8 caracteres del iniciador de canal seguido del nombre de 8 caracteres del módulo que se coloca en la cola de mensajes no entregados seguido de un identificador de tarea de 8 caracteres.

  El nombre o los nombres se rellenan cada uno a la derecha con espacios en blanco, al igual que cualquier espacio en el resto del campo. Cuando hay más de un nombre, no hay ningún separador entre ellos.
- En sistemas Windows , el gestor de colas utiliza:
  - Para una aplicación CICS , el nombre de transacción CICS
  - Para una aplicación no CICS, los 28 caracteres más a la derecha del nombre completo del ejecutable
- En IBM i, el gestor de colas utiliza el nombre de trabajo completo.
- En sistemas UNIX , el gestor de colas utiliza:
  - Para una aplicación CICS , el nombre de transacción CICS
  - Para una aplicación no CICS, MQ solicita al sistema operativo el nombre del proceso. Se devuelve como nombre de archivo de programa, sin vía de acceso completa. A continuación, MQ coloca este nombre de proceso en el MQMD de MQMD.PutApplName como se indica a continuación:

    AIX
    Si el nombre es menor o igual que 28 bytes, se inserta el nombre, rellenado a la derecha con espacios.

    Si el nombre es mayor que 28 bytes, se insertan los 28 bytes más a la izquierda del nombre.

    Linux y Solaris
    Si el nombre es menor o igual que 15 bytes, se inserta el nombre, rellenado a la derecha con espacios.

    Si el nombre es mayor que 15 bytes, se insertan los 15 bytes más a la izquierda del nombre, rellenados a la derecha con espacios.

    HP-UX
    Si el nombre es menor o igual que 14 bytes, se inserta el nombre, rellenado a la derecha con espacios.

    Si el nombre es mayor que 14 bytes, se insertan los 14 bytes más a la izquierda del nombre, rellenados a la derecha con espacios.

    Por ejemplo, si ejecuta `/opt/mqm/samp/bin/amqsput QNAME QMNAME`, el nombre de PutApples `'amqsput'`. Hay 21 caracteres de espacio de relleno en este campo MQCHAR28 . Tenga en cuenta que la vía de acceso completa que incluye `/opt/mqm/samp/bin` no se incluye en el nombre de PutAppl.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_ALL_CONTEXT en el parámetro *PutMsgOpts* . Se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida.

PutApplType (MQLONG)

Es el tipo de aplicación que coloca el mensaje y forma parte del **contexto de origen** del mensaje. Para obtener más información sobre el contexto de mensaje, consulte "Visión general de MQMD" en la página 394 y Contexto de mensaje.

PutApplType puede tener uno de los siguientes tipos estándar. También puede definir sus propios tipos, pero sólo con valores en el rango de MQAT_USER_FIRST a MQAT_USER_LAST.

MQAT_AIX
Aplicación AIX (el mismo valor que MQAT_UNIX).

INTERMEDIARIO
el intermediario de ubicación.

MQAT_CICS
Transacción CICS .

MQAT_CICS_BRIDGE
Puente CICS .

MQAT_CICS_VSE
Transacción CICS/VSE .

MQAT_DOS
WebSphere MQ Aplicación cliente MQI en PC DOS.

MQAT_DQM
Agente de gestor de colas distribuido.

MQAT_GUARDIÁN
Aplicación Tándem Guardian (mismo valor que MQAT_NSK).

MQAT_IMS
Aplicación IMS .

MQAT_IMS_BRIDGE
Puente IMS .

MQAT_JAVA
Java.

MQAT_MVS
MVS o aplicación TSO (el mismo valor que MQAT_ZOS).

MQAT_NOTES_AGENT
Lotus Notes .

MQAT_NSK
Aplicación HP Integrity NonStop Server .

MQAT_OS390
OS/390 (mismo valor que MQAT_ZOS).

MQAT_OS400
Aplicación IBM i .

MQAT_QMGR
Gestor de colas.

MQAT_UNIX
Aplicación UNIX .

MQAT_VOS
Aplicación Stratus VOS.

MQAT_WINDOWS
Aplicación Windows de 16 bits.

MQAT_WINDOWS_NT
Aplicación Windows de 32 bits.

MQAT_WLM
Aplicación del gestor de carga de trabajo de z/OS .

MQAT_XCF
XCF.

MQAT_ZOS
Aplicación z/OS .

MQAT_DEFAULT
Tipo de aplicación predeterminado.

Es el tipo de aplicación predeterminado para la plataforma en la que se ejecuta la aplicación.

Nota: El valor de esta constante es específico del entorno. Debido a esto, compile siempre la aplicación utilizando los archivos de cabecera, include o COPY que sean adecuados para la plataforma en la que se ejecutará la aplicación.

MQAT_DESCONOCIDO
Utilice este valor para indicar que el tipo de aplicación es desconocido, aunque haya otra información de contexto.

MQAT_USER_FIRST
Valor más bajo para el tipo de aplicación definido por el usuario.

MQAT_USER_LAST
Valor más alto para el tipo de aplicación definido por el usuario.

También se puede producir el siguiente valor especial:

MQAT_NO_CONTEXT
Este valor lo establece el gestor de colas cuando se coloca un mensaje sin contexto (es decir, se especifica la opción de contexto MQPMO_NO_CONTEXT).

Cuando se recupera un mensaje, *PutApplType* se puede probar para este valor para decidir si el mensaje tiene contexto (se recomienda que *PutApplType* nunca se establezca en MQAT_NO_CONTEXT, por parte de una aplicación que utilice MQPMO_SET_ALL_CONTEXT, si alguno de los otros campos de contexto no está en blanco).

Cuando el gestor de colas genera esta información como resultado de una colocación de aplicación, el campo se establece en un valor determinado por el entorno. En IBM i, se establece en MQAT_OS400; el gestor de colas nunca utiliza MQAT_CICS en IBM i.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_ALL_CONTEXT en el parámetro *PutMsgOpts* . Si no se especifica MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida.

Es un campo de salida para la llamada MQGET. El valor inicial de este campo es MQAT_NO_CONTEXT.

PutDate (MQCHAR8)

Es la fecha en la que se ha colocado el mensaje y forma parte del **contexto de origen** del mensaje. Para obtener más información sobre el contexto de mensaje, consulte "Visión general de MQMD" en la página 394 y Contexto de mensaje.

El formato utilizado para la fecha en la que el gestor de colas genera este campo es:

- AAAAMMDD

donde los caracteres representan:

AAAA
año (cuatro dígitos numéricos)

MM
mes del año (01 a 12)

DD
día del mes (01 a 31)

La hora media de Greenwich (GMT) se utiliza para los campos *PutDate* y *PutTime* , sujeto a que el reloj del sistema se establezca correctamente en GMT.

Si el mensaje se ha colocado como parte de una unidad de trabajo, la fecha es la fecha en la que se ha colocado el mensaje y no la fecha en la que se ha confirmado la unidad de trabajo.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_ALL_CONTEXT en el parámetro *PutMsgOpts* . El gestor de colas no comprueba el contenido del campo, excepto que se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona MQ_PUT_DATE_LENGTH. El valor inicial de este campo es la serie nula en C y 8 caracteres en blanco en otros lenguajes de programación.

PutTime (MQCHAR8)

Es la hora a la que se ha colocado el mensaje y forma parte del **contexto de origen** del mensaje. Para obtener más información sobre el contexto de mensaje, consulte "Visión general de MQMD" en la página 394 y Contexto de mensaje.

El formato utilizado para la hora en que el gestor de colas genera este campo es:

- HHMMSSTH

donde los caracteres representan (en orden):

HH
horas (de 00 a 23)

MM
minutos (de 00 a 59)

SS
segundos (de 00 a 59; consulte la nota)

T
décimas de segundo (de 0 a 9)

H
centésimas de segundo (0 a 9)

Nota: Si el reloj del sistema está sincronizado con un estándar de tiempo muy preciso, es posible que en raras ocasiones se devuelvan 60 o 61 durante los segundos en *PutTime*. Esto sucede cuando se insertan segundos bisiestos en el estándar de tiempo global.

La hora media de Greenwich (GMT) se utiliza para los campos *PutDate* y *PutTime* , sujeto a que el reloj del sistema se establezca correctamente en GMT.

Si el mensaje se ha colocado como parte de una unidad de trabajo, el tiempo es el momento en que se colocó el mensaje, y no el momento en que se comprometió la unidad de trabajo.

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_ALL_CONTEXT en el parámetro *PutMsgOpts* . El gestor de colas no comprueba el contenido del campo, excepto que se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona MQ_PUT_TIME_LENGTH. El valor inicial de este campo es la serie nula en C y 8 caracteres en blanco en otros lenguajes de programación.

ReplyToQ (MQCHAR48)

Es el nombre de la cola de mensajes a la que la aplicación que ha emitido la solicitud get para el mensaje envía los mensajes MQMT_REPLY y MQMT_REPORT. El nombre es el nombre local de una cola definida en el gestor de colas identificado por *ReplyToQMgr*. Esta cola no debe ser una cola modelo, aunque el gestor de colas emisor no lo verifique cuando se transfiere el mensaje.

Para las llamadas MQPUT y MQPUT1 , este campo no debe estar en blanco si el campo *MsgType* tiene el valor MQMT_REQUEST, o si el campo *Report* solicita algún mensaje de informe. Sin embargo, el valor especificado (o sustituido) se pasa a la aplicación que emite la solicitud get para el mensaje, sea cual sea el tipo de mensaje.

Si el campo *ReplyToQMgr* está en blanco, el gestor de colas local busca el nombre *ReplyToQ* en sus propias definiciones de cola. Si existe una definición local de una cola remota con este nombre, el valor *ReplyToQ* del mensaje transmitido se sustituye por el valor del atributo *RemoteQName* de la definición de la cola remota, y este valor se devuelve en el descriptor de mensaje cuando la aplicación receptora emite una llamada MQGET para el mensaje. Si no existe una definición local de una cola remota, *ReplyToQ* no se modifica.

Si se especifica el nombre, puede contener espacios en blanco finales; el primer carácter nulo y los caracteres que le siguen se tratan como espacios en blanco. De lo contrario, no se realiza ninguna comprobación de que el nombre cumple las reglas de denominación para las colas; esto también es cierto para el nombre transmitido, si el *ReplyToQ* se sustituye en el mensaje transmitido. La única comprobación realizada es que se ha especificado un nombre, si las circunstancias lo requieren.

Si no es necesaria una cola de respuestas, establezca el campo *ReplyToQ* en blancos, o (en el lenguaje de programación C) en la serie nula, o en uno o más blancos seguidos de un carácter nulo; no deje el campo sin inicializar.

Para la llamada MQGET, el gestor de colas siempre devuelve el nombre rellenado con espacios en blanco a la longitud del campo.

Si un mensaje que requiere un mensaje de informe no se puede entregar, y el mensaje de informe tampoco se puede entregar a la cola especificada, tanto el mensaje original como el mensaje de informe van a la cola de mensajes no entregados (undelivered-message) (consulte el atributo *DeadLetterQName* descrito en “Atributos para el gestor de colas” en la página 780).

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ReplyToGestor de colas (MQCHAR48)

Es el nombre del gestor de colas al que enviar el mensaje de respuesta o el mensaje de informe. *ReplyToQ* es el nombre local de una cola definida en este gestor de colas.

Si el campo *ReplyToQMgr* está en blanco, el gestor de colas local busca el nombre *ReplyToQ* en sus definiciones de cola. Si existe una definición local de una cola remota con este nombre, el valor *ReplyToQMgr* del mensaje transmitido se sustituye por el valor del atributo *RemoteQMgrName* de la definición de la cola remota, y este valor se devuelve en el descriptor de mensaje cuando la aplicación receptora emite una llamada MQGET para el mensaje. Si no existe una definición local de una cola remota, el *ReplyToQMgr* que se transmite con el mensaje es el nombre del gestor de colas local.

Si se especifica el nombre, puede contener espacios en blanco finales; el primer carácter nulo y los caracteres que le siguen se tratan como espacios en blanco. De lo contrario, no se realiza ninguna comprobación de que el nombre cumple las reglas de denominación para los gestores de colas, o de que el gestor de colas emisor conoce este nombre; esto también es cierto para el nombre transmitido, si el *ReplyToQMgr* se sustituye en el mensaje transmitido.

Si no es necesaria una cola de respuestas, establezca el campo *ReplyToQMgr* en blancos, o (en el lenguaje de programación C) en la serie nula, o en uno o más blancos seguidos de un carácter nulo; no deje el campo sin inicializar.

Para la llamada MQGET, el gestor de colas siempre devuelve el nombre rellenado con espacios en blanco a la longitud del campo.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

Report (MQLONG)

Un mensaje de informe es un mensaje sobre otro mensaje, utilizado para informar a una aplicación sobre sucesos esperados o inesperados relacionados con el mensaje original. El campo *Report* permite a la aplicación que envía el mensaje original especificar qué mensajes de informe son necesarios, si los datos de mensaje de aplicación se van a incluir en ellos y también (para informes y respuestas) cómo se van a establecer los identificadores de mensaje y correlación en el informe o mensaje de respuesta. Se puede solicitar cualquiera o todos (o ninguno) de los siguientes tipos de mensajes de informe:

- Excepción
- Caducidad
- Confirmar al llegar (COA)
- Confirmar en entrega (COD)
- notificación de acción positiva (PAN)
- notificación de acción negativa (NAN)

Si se necesita más de un tipo de mensaje de informe u otras opciones de informe, los valores pueden ser:

- Sumado (no añadir la misma constante más de una vez), o
- Combinado utilizando la operación OR bit a bit (si el lenguaje de programación da soporte a operaciones de bits).

La aplicación que recibe el mensaje de informe puede determinar la razón por la que se ha generado el informe examinando el campo *Feedback* en el MQMD; consulte el campo *Feedback* para obtener más detalles.

El uso de opciones de informe al colocar un mensaje en un tema puede hacer que se generen y envíen a la aplicación cero, uno o varios mensajes de informe. Esto se debe a que el mensaje de publicación puede enviarse a cero, una o muchas aplicaciones de suscripción.

Opciones de excepción: especifique una de las opciones listadas para solicitar un mensaje de informe de excepción.

MQRO_EXCEPTION
Un agente de canal de mensajes genera este tipo de informe cuando se envía un mensaje a otro gestor de colas y el mensaje no se puede entregar a la cola de destino especificada. Por ejemplo, la cola de destino o una cola de transmisión intermedia puede estar llena, o el mensaje puede ser demasiado grande para la cola.

La generación del mensaje de informe de excepción depende de la persistencia del mensaje original y de la velocidad del canal de mensajes (normal o rápido) a través del cual viaja el mensaje original:

- Para todos los mensajes persistentes y para los mensajes no persistentes que viajan a través de canales de mensajes normales, el informe de excepción se genera *sólo* si la acción especificada por la aplicación emisora para la condición de error se puede completar correctamente. La aplicación emisora puede especificar una de las acciones siguientes para controlar la disposición del mensaje original cuando se produce la condición de error:
  - MQRO_DEAD_LETTER_Q (coloca el mensaje original en la cola de mensajes no entregados).
  - MQRO_DISCARD_MSG (descarta el mensaje original).

Si la acción especificada por la aplicación emisora no se puede completar correctamente, el mensaje original se deja en la cola de transmisión y no se genera ningún mensaje de informe de excepción.

- Para los mensajes no persistentes que viajan a través de canales de mensajes rápidos, el mensaje original se elimina de la cola de transmisión y el informe de excepción genera *incluso si* la acción especificada para la condición de error no se puede completar correctamente. Por ejemplo, si se especifica MQRO_DEAD_LETTER_Q, pero el mensaje original no se puede colocar en la cola de mensajes no entregados porque dicha cola está llena, se genera el mensaje de informe de excepción y se descarta el mensaje original.

  Para obtener más información sobre los canales de mensajes normales y rápidos, consulte Velocidad de mensajes no persistentes (NPMSPEED) .

No se genera un informe de excepción si la aplicación que ha colocado el mensaje original puede recibir una notificación síncrona del problema mediante el código de razón devuelto por la llamada MQPUT o MQPUT1 .

Las aplicaciones también pueden enviar informes de excepción, para indicar que un mensaje no se puede procesar (por ejemplo, porque es una transacción de débito que haría que la cuenta excediera su límite de crédito).

Los datos de mensaje del mensaje original no se incluyen con el mensaje de informe.

No especifique más de uno de MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA y MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_EXCEPTION_WITH_DATA
Es lo mismo que MQRO_EXCEPTION, excepto que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de cabecera MQ , se incluyen en el mensaje de informe, además de los 100 bytes de datos de aplicación.

No especifique más de uno de MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA y MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_EXCEPTION_WITH_FULL_DATA
Informes de excepción con datos completos necesarios.

Es lo mismo que MQRO_EXCEPTION, excepto que todos los datos de mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA y MQRO_EXCEPTION_WITH_FULL_DATA.

Opciones de caducidad: especifique una de las opciones listadas para solicitar un mensaje de informe de caducidad.

MQRO_EXPIRATION
Este tipo de informe lo genera el gestor de colas si el mensaje se descarta antes de la entrega a una aplicación porque ha pasado su hora de caducidad (consulte el campo *Expiry*). Si esta opción no está establecida, no se genera ningún mensaje de informe si se descarta un mensaje por esta razón (incluso si especifica una de las opciones MQRO_EXCEPTION_ *).

Los datos de mensaje del mensaje original no se incluyen con el mensaje de informe.

No especifique más de uno de MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA y MQRO_EXPIRATION_WITH_FULL_DATA.

MQRO_EXPIRATION_WITH_DATA
Es el mismo que MQRO_EXPIRATION, excepto que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de cabecera MQ , se incluyen en el mensaje de informe, además de los 100 bytes de datos de aplicación.

No especifique más de uno de MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA y MQRO_EXPIRATION_WITH_FULL_DATA.

MQRO_EXPIRATION_WITH_FULL_DATA
Es lo mismo que MQRO_EXPIRATION, excepto que todos los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA y MQRO_EXPIRATION_WITH_FULL_DATA.

Opciones de confirmación al llegar: especifique una de las opciones listadas para solicitar un mensaje de informe de confirmación al llegar.

MQRO_COA
Este tipo de informe lo genera el gestor de colas propietario de la cola de destino cuando el mensaje se coloca en la cola de destino. Los datos de mensaje del mensaje original no se incluyen con el mensaje de informe.

Si el mensaje se coloca como parte de una unidad de trabajo y la cola de destino es una cola local, el mensaje de informe de COA generado por el gestor de colas sólo se puede recuperar si se confirma la unidad de trabajo.

No se genera un informe COA si el campo *Format* del descriptor de mensaje es MQFMT_XMIT_Q_HEADER o MQFMT_DEAD_LETTER_HEADER. Esto impide que se genere un informe de COA si el mensaje se coloca en una cola de transmisión o no se puede entregar y se coloca en una cola de mensajes no entregados.

En el caso de una cola puente IMS , el informe de COA se genera cuando el mensaje llega a la cola IMS (acuse de recibo recibido de IMS) y no cuando el mensaje se coloca en la cola puente MQ . Esto significa que si IMS no está activo, no se genera ningún informe de COA hasta que se inicia IMS y se pone en cola un mensaje en la cola IMS .

El usuario que ejecuta un programa que coloca un mensaje con MQMD.Report= MQRO_COA debe tener autorización + passid en la cola de respuestas. Si el usuario no tiene autorización + passid, el mensaje de informe COA no llega a la cola de respuestas. Se ha intentado colocar el mensaje de informe en la cola de mensajes no entregados.

No especifique más de uno de MQRO_COA, MQRO_COA_WITH_DATA y MQRO_COA_WITH_FULL_DATA.

MQRO_COA_WITH_DATA
Es el mismo que MQRO_COA, excepto que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de cabecera MQ , se incluyen en el mensaje de informe, además de los 100 bytes de datos de aplicación.

No especifique más de uno de MQRO_COA, MQRO_COA_WITH_DATA y MQRO_COA_WITH_FULL_DATA.

MQRO_COA_WITH_FULL_DATA
Es lo mismo que MQRO_COA, excepto que todos los datos de mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

No especifique más de uno de MQRO_COA, MQRO_COA_WITH_DATA y MQRO_COA_WITH_FULL_DATA.

Opciones de confirmación de entrega: especifique una de las opciones listadas para solicitar un mensaje de informe de confirmación de entrega.

MQRO_COD
Este tipo de informe lo genera el gestor de colas cuando una aplicación recupera el mensaje de la cola de destino de una forma que suprime el mensaje de la cola. Los datos de mensaje del mensaje original no se incluyen con el mensaje de informe.

Si el mensaje se recupera como parte de una unidad de trabajo, el mensaje de informe se genera dentro de la misma unidad de trabajo, de modo que el informe no está disponible hasta que se confirma la unidad de trabajo. Si la unidad de trabajo se restituye, el informe no se envía.

No siempre se genera un informe COD si se recupera un mensaje con la opción MQGMO_MARK_SKIP_BACKOUT. Si la unidad de trabajo primaria se restituye pero se confirma la unidad de trabajo secundaria, el mensaje se elimina de la cola, pero no se genera un informe COD.

No se genera un informe COD si el campo *Format* del descriptor de mensaje es MQFMT_DEAD_LETTER_HEADER. Esto impide que se genere un informe COD si el mensaje no se puede entregar y se coloca en una cola de mensajes no entregados.

MQRO_COD no es válido si la cola de destino es una cola XCF.

No especifique más de uno de MQRO_COD, MQRO_COD_WITH_DATA y MQRO_COD_WITH_FULL_DATA.

MQRO_COD_WITH_DATA
Es el mismo que MQRO_COD, excepto que los primeros 100 bytes de los datos del mensaje de aplicación del mensaje original se incluyen en el mensaje de informe. Si el mensaje original contiene una o más estructuras de cabecera MQ , se incluyen en el mensaje de informe, además de los 100 bytes de datos de aplicación.

Si se especifica MQGMO_ACCEPT_TRUNCATED_MSG en la llamada MQGET para el mensaje original y el mensaje recuperado se trunca, la cantidad de datos de mensaje de aplicación colocados en el mensaje de informe depende del entorno:

- En z/OS, es el mínimo de:
  - La longitud del mensaje original
  - La longitud del almacenamiento intermedio utilizado para recuperar el mensaje
  - 100 bytes.
- En otros entornos, es el mínimo de:
  - La longitud del mensaje original
  - 100 bytes.

MQRO_COD_WITH_DATA no es válido si la cola de destino es una cola XCF.

No especifique más de uno de MQRO_COD, MQRO_COD_WITH_DATA y MQRO_COD_WITH_FULL_DATA.

MQRO_COD_WITH_FULL_DATA
Es lo mismo que MQRO_COD, excepto que todos los datos de mensaje de aplicación del mensaje original se incluyen en el mensaje de informe.

MQRO_COD_WITH_FULL_DATA no es válido si la cola de destino es una cola XCF.

No especifique más de uno de MQRO_COD, MQRO_COD_WITH_DATA y MQRO_COD_WITH_FULL_DATA.

Opciones de notificación de acción: especifique una o ambas de las opciones listadas para solicitar que la aplicación receptora envíe un mensaje de informe de acción positiva o de acción negativa.

MQRO_PAN
Este tipo de informe lo genera la aplicación que recupera el mensaje y actúa sobre él. Indica que la acción solicitada en el mensaje se ha realizado correctamente. La aplicación que genera el informe determina si se van a incluir datos con el informe.

Aparte de transmitir esta solicitud a la aplicación que recupera el mensaje, el gestor de colas no realiza ninguna acción basada en esta opción. La aplicación de recuperación debe generar el informe si procede.

MQRO_NAN
Este tipo de informe lo genera la aplicación que recupera el mensaje y actúa sobre él. Indica que la acción solicitada en el mensaje *no* se ha realizado correctamente. La aplicación que genera el informe determina si se van a incluir datos con el informe. Por ejemplo, es posible que desee incluir algunos datos que indiquen por qué no se ha podido realizar la solicitud.

Aparte de transmitir esta solicitud a la aplicación que recupera el mensaje, el gestor de colas no realiza ninguna acción basada en esta opción. La aplicación de recuperación debe generar el informe si procede.

La solicitud debe determinar qué condiciones corresponden a una acción positiva y cuáles a una acción negativa. Sin embargo, si la solicitud sólo se ha realizado parcialmente, genere un informe NAN en lugar de un informe PAN si se solicita. Cada condición posible debe corresponder a una acción positiva, o a una acción negativa, pero no a ambas.

Opciones de identificador de mensaje: especifique una de las opciones listadas para controlar cómo se va a establecer el *MsgId* del mensaje de informe (o del mensaje de respuesta).

MQRO_NEW_MSG_ID
Esta es la acción predeterminada e indica que si se genera un informe o respuesta como resultado de este mensaje, se genera un nuevo *MsgId* para el informe o mensaje de respuesta.

MQRO_PASS_MSG_ID
Si se genera un informe o respuesta como resultado de este mensaje, el *MsgId* de este mensaje se copia en el *MsgId* del informe o mensaje de respuesta.

El *MsgId* de un mensaje de publicación será diferente para cada suscriptor que reciba una copia de la publicación y, por lo tanto, el *MsgId* copiado en el informe o mensaje de respuesta será diferente para cada uno.

Si no se especifica esta opción, se presupone MQRO_NEW_MSG_ID.

Opciones de identificador de correlación: especifique una de las opciones listadas para controlar cómo se va a establecer el *CorrelId* del mensaje de informe (o del mensaje de respuesta).

MQRO_COPY_MSG_ID_TO_CORREL_ID
Esta es la acción predeterminada e indica que si se genera un informe o respuesta como resultado de este mensaje, el *MsgId* de este mensaje se copia en el *CorrelId* del informe o mensaje de respuesta.

El *MsgId* de un mensaje de publicación será diferente para cada suscriptor que reciba una copia de la publicación y, por lo tanto, el *MsgId* copiado en el *CorrelId* del informe o mensaje de respuesta será diferente para cada uno.

MQRO_PASS_CORREL_ID
Si se genera un informe o respuesta como resultado de este mensaje, el *CorrelId* de este mensaje se copia en el *CorrelId* del informe o mensaje de respuesta.

El *CorrelId* de un mensaje de publicación será específico de un suscriptor a menos que utilice la opción MQSO_SET_CORREL_ID y establezca el campo de ID SubCorrelen MQSD en MQCI_NONE. Por lo tanto, es posible que el *CorrelId* copiado en el *CorrelId* del informe o mensaje de respuesta sea diferente para cada uno.

Si no se especifica esta opción, se presupone MQRO_COPY_MSG_ID_TO_CORREL_ID.

Los servidores que respondían a solicitudes o generaban mensajes de informe deben comprobar si las opciones MQRO_PASS_MSG_ID o MQRO_PASS_CORREL_ID se habían establecido en el mensaje original. Si lo eran, los servidores deben realizar la acción descrita para estas opciones. Si no se establece ninguna, los servidores deben realizar la acción predeterminada correspondiente.

Opciones de disposición: especifique una de las opciones listadas para controlar la disposición del mensaje original cuando no se puede entregar a la cola de destino. La aplicación puede establecer las opciones de disposición independientemente de solicitar informes de excepción.

MQRO_DEAD_LETTER_Q
Es la acción predeterminada y coloca el mensaje en la cola de mensajes no entregados si el mensaje no se puede entregar a la cola de destino. Esto sucede en las situaciones siguientes:

- Cuando la aplicación que ha colocado el mensaje original no puede ser notificada de forma síncrona del problema mediante el código de razón devuelto por la llamada MQPUT o MQPUT1 . Se genera un mensaje de informe de excepción, si el remitente lo ha solicitado.
- Cuando la aplicación que ha colocado el mensaje original estaba colocando en un tema

MQRO_DISCARD_MSG
Esto descarta el mensaje si no se puede entregar a la cola de destino. Esto sucede en las situaciones siguientes:

- Cuando la aplicación que ha colocado el mensaje original no puede ser notificada de forma síncrona del problema mediante el código de razón devuelto por la llamada MQPUT o MQPUT1 . Se genera un mensaje de informe de excepción, si el remitente lo ha solicitado.
- Cuando la aplicación que ha colocado el mensaje original estaba colocando en un tema

Si desea devolver el mensaje original al remitente, sin que el mensaje original se coloque en la cola de mensajes no entregados, el remitente debe especificar MQRO_DISCARD_MSG con MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_PASS_DISCARD_AND_EXPIRY
Si esta opción se establece en un mensaje y se genera un informe o una respuesta debido a él, el descriptor de mensaje del informe hereda:

- MQRO_DISCARD_MSG si se ha establecido.
- El tiempo de caducidad restante del mensaje (si no es un informe de caducidad). Si se trata de un informe de caducidad, el tiempo de caducidad se establece en 60 segundos.

Opción de actividad

MQRO_ACTIVITY
El uso de este valor permite rastrear la ruta de **cualquier** mensaje a través de una red de gestores de colas. La opción de informe se puede especificar en cualquier mensaje de usuario actual, lo que permite al instante empezar a calcular la ruta del mensaje a través de la red.

Si la aplicación que genera el mensaje no puede activar los informes de actividad, los informes se pueden activar utilizando una salida cruzada de API proporcionada por los administradores del gestor de colas.

Nota:

1. Cuanto menos sean los gestores de colas de la red capaces de generar informes de actividad, menos detallada será la ruta.
2. Los informes de actividad pueden ser difíciles de colocar en el orden correcto para determinar la ruta tomada.
3. Es posible que los informes de actividad no puedan encontrar una ruta a su destino solicitado.
4. Los mensajes con este conjunto de opciones de informe deben ser aceptados por cualquier gestor de colas, incluso si no entienden la opción. Esto permite que la opción de informe se establezca en cualquier mensaje de usuario, incluso si lo procesa un gestor de colas que no sea de la versión 6.0 o posterior.
5. Si un proceso, ya sea un gestor de colas o un proceso de usuario, realiza una actividad en un mensaje con esta opción establecida, puede elegir generar y colocar un informe de actividad.

Opción predeterminada: especifique lo siguiente si no se necesitan opciones de informe:

MQRO_NONE
Utilice este valor para indicar que no se ha especificado ninguna otra opción. MQRO_NONE está definido para ayudar a la documentación del programa. No se pretende que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

Información general:

1. La aplicación que envía el mensaje original debe solicitar específicamente todos los tipos de informe necesarios. Por ejemplo, si se solicita un informe de COA pero no se solicita un informe de excepción, se genera un informe de COA cuando el mensaje se coloca en la cola de destino, pero no se genera ningún informe de excepción si la cola de destino está llena cuando el mensaje llega allí. Si no se establece ninguna opción *Report* , el gestor de colas o el agente de canal de mensajes (MCA) no genera ningún mensaje de informe.

   Se pueden especificar algunas opciones de informe aunque el gestor de colas local no las reconozca; esto es útil cuando el gestor de colas de *destino* debe procesar la opción. Consulte "Opciones de informe y distintivos de mensaje" en la página 884 para obtener más detalles.

   Si se solicita un mensaje de informe, el nombre de la cola a la que enviar el informe debe especificarse en el campo *ReplyToQ* . Cuando se recibe un mensaje de informe, la naturaleza del informe se puede determinar examinando el campo *Feedback* en el descriptor de mensaje.
2. Si el gestor de colas o MCA que genera un mensaje de informe no puede colocar el mensaje de informe en la cola de respuestas (por ejemplo, porque la cola de respuestas o la cola de transmisión está llena), el mensaje de informe se coloca en su lugar en la cola de mensajes no entregados. Si *también* falla, o no hay ninguna cola de mensajes no entregados, la acción realizada depende del tipo de mensaje de informe:
   - Si el mensaje de informe es un informe de excepción, el mensaje que ha generado el informe de excepción se deja en su cola de transmisión; esto garantiza que el mensaje no se pierda.
   - Para todos los demás tipos de informe, el mensaje de informe se descarta y el proceso continúa normalmente. Esto se hace porque el mensaje original ya se ha entregado de forma segura (para los mensajes de informe COA o COD), o ya no es de ningún interés (para un mensaje de informe de caducidad).

   Una vez que un mensaje de informe se ha colocado correctamente en una cola (ya sea la cola de destino o una cola de transmisión intermedia), el mensaje ya no está sujeto a un proceso especial; se trata igual que cualquier otro mensaje.
3. Cuando se genera el informe, se abre la cola *ReplyToQ* y el mensaje de informe se coloca utilizando la autorización del *UserIdentifier* en el MQMD del mensaje que provoca el informe, excepto en los casos siguientes:
   - Los informes de excepción generados por un MCA receptor se colocan con la autorización que el MCA utilizó cuando intentó colocar el mensaje que causaba el informe.
   - Los informes de COA generados por el gestor de colas se colocan con la autorización que se haya utilizado cuando el mensaje que ha provocado el informe se ha colocado en el gestor de colas que genera el informe. Por ejemplo, si el mensaje ha sido transferido por un MCA receptor utilizando el identificador de usuario del MCA, el gestor de colas coloca el informe COA utilizando el identificador de usuario del MCA.

   Las aplicaciones que generan informes deben utilizar la misma autorización que utilizan para generar una respuesta; normalmente es la autorización del identificador de usuario en el mensaje original.

   Si el informe tiene que viajar a un destino remoto, los remitentes y receptores pueden decidir si lo aceptan, del mismo modo que lo hacen para otros mensajes.
4. Si se solicita un mensaje de informe con datos:
   - El mensaje de informe siempre se genera con la cantidad de datos solicitados por el remitente del mensaje original. Si el mensaje de informe es demasiado grande para la cola de respuestas, se produce el proceso descrito anteriormente; el mensaje de informe nunca se trunca para que quepa en la cola de respuestas.
   - Si el *Format* del mensaje original es MQFMT_XMIT_Q_HEADER, los datos incluidos en el informe no incluyen MQXQH. Los datos del informe empiezan con el primer byte de los datos más allá de MQXQH en el mensaje original. Esto ocurre tanto si la cola es una cola de transmisión como si no.
5. Si se recibe un mensaje de informe de COA, COD o caducidad en la cola de respuestas, se garantiza que el mensaje original ha llegado, se ha entregado o ha caducado, según corresponda. Sin embargo,

si se solicita uno o varios de estos mensajes de informe y *no* se recibe, no se puede suponer lo contrario, porque podría haberse producido una de las situaciones siguientes:

a. El mensaje de informe se retiene porque un enlace está inactivo.

b. El mensaje de informe se retiene porque existe una condición de bloqueo en una cola de transmisión intermedia o en la cola de respuestas (por ejemplo, la cola está llena o inhibida para colocaciones).

c. El mensaje de informe está en una cola de mensajes no entregados.

d. Cuando el gestor de colas intentaba generar el mensaje de informe, no podía colocarlo en la cola adecuada, ni en la cola de mensajes no entregados, por lo que no se podía generar el mensaje de informe.

e. Se ha producido una anomalía del gestor de colas entre la acción que se notifica (llegada, entrega o caducidad) y la generación del mensaje de informe correspondiente. (Esto no sucede para los mensajes de informe COD si la aplicación recupera el mensaje original dentro de una unidad de trabajo, ya que el mensaje de informe COD se genera dentro de la misma unidad de trabajo.)

Los mensajes de informe de excepción se pueden contener de la misma forma por las razones 1, 2 y 3 anteriores. Sin embargo, cuando un MCA no puede generar un mensaje de informe de excepción (el mensaje de informe no se puede colocar en la cola de respuestas o en la cola de mensajes no entregados), el mensaje original permanece en la cola de transmisión del emisor y el canal se cierra. Esto se produce independientemente de si el mensaje de informe se iba a generar en el extremo emisor o receptor del canal.

6. Si el mensaje original se bloquea temporalmente (lo que genera un mensaje de informe de excepción y el mensaje original se coloca en una cola de mensajes no entregados), pero el bloqueo se borra y una aplicación lee el mensaje original de la cola de mensajes no entregados y lo coloca de nuevo en su destino, puede ocurrir lo siguiente:

   - Aunque se ha generado un mensaje de informe de excepción, el mensaje original finalmente llega correctamente a su destino.
   - Se genera más de un mensaje de informe de excepción con respecto a un único mensaje original, porque el mensaje original puede encontrar otro bloqueo más adelante.

Informar de mensajes al transferir a un tema:

1. Se pueden generar informes al colocar un mensaje en un tema. Este mensaje se enviará a todos los suscriptores del tema, que podría ser cero, uno o muchos. Esto debe tenerse en cuenta al elegir utilizar las opciones de informe, ya que se podrían generar muchos mensajes de informe como resultado.
2. Al colocar un mensaje en un tema, puede haber muchas colas de destino a las que se debe dar una copia del mensaje. Si algunas de estas colas de destino tienen un problema, como la cola llena, la finalización satisfactoria de MQPUT depende del valor de NPMSGDLV o PMSGDLV (en función de la persistencia del mensaje). Si el valor es tal que la entrega de mensajes a la cola de destino debe ser satisfactoria (por ejemplo, es un mensaje persistente para un suscriptor duradero y PMSGDLV se establece en ALL o ALLDUR), el éxito se define como uno de los siguientes criterios que se cumplen:

   - Se ha colocado correctamente en la cola de suscriptores
   - Uso de MQRO_DEAD_LETTER_Q y una colocación satisfactoria en la cola de mensajes no entregados si la cola de suscriptores no puede tomar el mensaje
   - Utilice MQRO_DISCARD_MSG si la cola de suscriptores no puede tomar el mensaje.

Mensajes de informe para segmentos de mensajes:

1. Se pueden solicitar mensajes de informe para los mensajes que tienen la segmentación permitida (consulte la descripción del distintivo MQMF_SEGMENTATION_ALLOWED). Si el gestor de colas considera necesario segmentar el mensaje, se puede generar un mensaje de informe para cada uno de los segmentos que posteriormente encuentren la condición relevante. Las aplicaciones deben estar preparadas para recibir varios mensajes de informe para cada tipo de mensaje de informe solicitado. Utilice el campo *GroupId* en el mensaje de informe para correlacionar los varios informes con el

identificador de grupo del mensaje original, y el campo *Feedback* identifique el tipo de cada mensaje de informe.

2. Si se utiliza MQGMO_LOGICAL_ORDER para recuperar mensajes de informe para segmentos, tenga en cuenta que las llamadas MQGET sucesivas pueden devolver informes de *tipos diferentes* . Por ejemplo, si se solicitan los informes COA y COD para un mensaje segmentado por el gestor de colas, las llamadas MQGET para los mensajes de informe pueden devolver los mensajes de informe COA y COD intercalados de forma imprevisible. Evite esto utilizando la opción MQGMO_COMPLETE_MSG (opcionalmente con MQGMO_ACCEPT_TRUNCATED_MSG). MQGMO_COMPLETE_MSG hace que el gestor de colas vuelva a ensamblar los mensajes de informe que tienen el mismo tipo de informe. Por ejemplo, la primera llamada MQGET podría volver a ensamblar todos los mensajes COA relacionados con el mensaje original, y la segunda llamada MQGET podría volver a ensamblar todos los mensajes COD. Lo que se reensambla primero depende del tipo de mensaje de informe que aparezca primero en la cola.
3. Las aplicaciones que ellas mismas colocan segmentos pueden especificar diferentes opciones de informe para cada segmento. Sin embargo, tenga en cuenta los siguientes puntos:
   - Si los segmentos se recuperan utilizando la opción MQGMO_COMPLETE_MSG, el gestor de colas solo respeta las opciones de informe del *primer* segmento.
   - Si los segmentos se recuperan uno por uno, y la mayoría de ellos tienen una de las opciones MQRO_COD_ *, pero al menos un segmento no lo hace, no puede utilizar la opción MQGMO_COMPLETE_MSG para recuperar los mensajes de informe con una sola llamada MQGET, o utilizar la opción MQGMO_ALL_SEGMENTS_AVAILABLE para detectar cuándo han llegado todos los mensajes de informe.
4. En una red MQ , los gestores de colas pueden tener distintas prestaciones. Si un mensaje de informe para un segmento es generado por un gestor de colas o MCA que no soporta la segmentación, el gestor de colas o MCA no incluye de forma predeterminada la información de segmento necesaria en el mensaje de informe, y esto podría dificultar la identificación del mensaje original que ha hecho que se generara el informe. Evite esta dificultad solicitando datos con el mensaje de informe, es decir, especificando las opciones MQRO_ * _WITH_DATA o MQRO_ * _WITH_FULL_DATA adecuadas. Sin embargo, tenga en cuenta que si se especifica MQRO_ * _WITH_DATA, es posible que se devuelvan *menos de* 100 bytes de datos de mensaje de aplicación a la aplicación que recupera el mensaje de informe, si el mensaje de informe lo genera un gestor de colas o un MCA que no soporta la segmentación.

Contenido del descriptor de mensaje para un mensaje de informe: cuando el gestor de colas o el agente de canal de mensajes (MCA) genera un mensaje de informe, establece los campos del descriptor de mensaje en los valores siguientes y, a continuación, coloca el mensaje de la forma normal.

Campo de MQMD	Valor utilizado
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_2
Report	MQRO_NONE
MsgType	MQMT_REPORT
Expiry	MQEI_UNLIMITED
Feedback	Según corresponda para la naturaleza del informe (MQFB_COA, MQFB_COD, MQFB_EXPIRATION o un valor MQRC_ *)
Encoding	Copiado del descriptor de mensaje original
CodedCharSetId	Copiado del descriptor de mensaje original
Format	Copiado del descriptor de mensaje original
Priority	Copiado del descriptor de mensaje original
Persistence	Copiado del descriptor de mensaje original

Campo de MQMD	Valor utilizado
MsgId	Según lo especificado por las opciones de informe en el descriptor de mensaje original
CorrelId	Según lo especificado por las opciones de informe en el descriptor de mensaje original
BackoutCount	0
ReplyToQ	Espacios en blanco
ReplyToQMgr	Nombre de gestor de colas
UserIdentifier	Tal como lo establece la opción MQPMO_PASS_IDENTITY_CONTEXT
AccountingToken	Tal como lo establece la opción MQPMO_PASS_IDENTITY_CONTEXT
ApplIdentityData	Tal como lo establece la opción MQPMO_PASS_IDENTITY_CONTEXT
PutApplType	MQAT_QMGR, o según corresponda para el agente de canal de mensajes
PutApplName	Primeros 28 bytes del nombre del gestor de colas o del nombre del agente de canal de mensajes. Para los mensajes de informe generados por el puente IMS , este campo contiene el nombre de grupo XCF y el nombre de miembro XCF del sistema IMS con el que se relaciona el mensaje.
PutDate	Fecha en la que se envía el mensaje de informe
PutTime	Hora a la que se envía el mensaje de informe
ApplOriginData	Espacios en blanco
GroupId	Copiado del descriptor de mensaje original
MsgSeqNumber	Copiado del descriptor de mensaje original
Offset	Copiado del descriptor de mensaje original
MsgFlags	Copiado del descriptor de mensaje original
OriginalLength	Copiado del descriptor de mensaje original si no es MQOL_UNDEFINED, y establecido en la longitud de los datos de mensaje originales de lo contrario

Se recomienda una aplicación que genere un informe para establecer valores similares, excepto para lo siguiente:

- El campo *ReplyToQMgr* se puede establecer en blancos (el gestor de colas lo cambia por el nombre del gestor de colas local cuando se coloca el mensaje).
- Establezca los campos de contexto utilizando la opción que se habría utilizado para una respuesta, normalmente MQPMO_PASS_IDENTITY_CONTEXT.

Análisis del campo de informe: el campo *Report* contiene subcampos; debido a esto, las aplicaciones que necesitan comprobar si el remitente del mensaje ha solicitado un informe determinado deben utilizar una de las técnicas descritas en "Análisis del campo de informe" en la página 886.

Es un campo de salida para la llamada MQGET y un campo de entrada para las llamadas MQPUT y MQPUT1 . El valor inicial de este campo es MQRO_NONE.

StrucId (MQCHAR4)

Este es el identificador de estructura y debe ser:

MQMD_STRUC_ID
Identificador de la estructura del descriptor de mensaje.

Para el lenguaje de programación C, también se define la constante MQMD_STRUC_ID_ARRAY; tiene el mismo valor que MQMD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQMD_STRUC_ID.

UserIdentifier (MQCHAR12)

Forma parte del **contexto de identidad** del mensaje. Para obtener más información sobre el contexto de mensaje, consulte "Visión general de MQMD" en la página 394 y Contexto de mensaje .

UserIdentifier especifica el identificador de usuario de la aplicación que ha originado el mensaje. El gestor de colas trata esta información como datos de tipo carácter, pero no define su formato.

Después de recibir un mensaje, utilice *UserIdentifier* en el campo *AlternateUserId* del parámetro *ObjDesc* de una llamada MQOPEN o MQPUT1 posterior para realizar la comprobación de autorización para el usuario *UserIdentifier* en lugar de la aplicación que realiza la apertura.

Cuando el gestor de colas genera esta información para una llamada MQPUT o MQPUT1 :

- En z/OS, el gestor de colas utiliza *AlternateUserId* del parámetro *ObjDesc* de la llamada MQOPEN o MQPUT1 si se ha especificado la opción MQOO_ALTERNATE_USER_AUTHORITY o MQPMO_ALTERNATE_USER_AUTHORITY. Si no se ha especificado la opción relevante, el gestor de colas utiliza un identificador de usuario determinado desde el entorno.
- En otros entornos, el gestor de colas siempre utiliza un identificador de usuario determinado a partir del entorno.

Cuando el identificador de usuario se determina desde el entorno:

- En z/OS, el gestor de colas utiliza:
  - Para MVS (lote), el identificador de usuario de la tarjeta JES JOB o tarea iniciada
  - Para TSO, el identificador de usuario propagado al trabajo durante el envío del trabajo
  - Para CICS, el identificador de usuario asociado a la tarea
  - Para IMS, el identificador de usuario depende del tipo de aplicación:
    - Durante:
      - Regiones BMP no de mensaje
      - Regiones IFP no de mensajes
      - Las regiones BMP y IFP de mensajes que *no* han emitido una llamada de GU satisfactoria

      el gestor de colas utiliza el identificador de usuario de la tarjeta JES JOB de la región o el identificador de usuario TSO. Si están en blanco o son nulos, utiliza el nombre del bloque de especificación de programa (PSB).
    - Durante:
      - Mensaje BMP y regiones IFP de mensaje que *han* emitido una llamada de GU satisfactoria
      - Regiones MPP

      el gestor de colas utiliza uno de los siguientes:
      - El identificador de usuario conectado asociado con el mensaje
      - El nombre del terminal lógico (LTERM)
      - El identificador de usuario de la tarjeta JES JOB de la región
      - El identificador de usuario TSO
      - El nombre de PSB
- En IBM i, el gestor de colas utiliza el nombre del perfil de usuario asociado con el trabajo de aplicación.
- En sistemas UNIX , el gestor de colas utiliza:
  - El nombre de inicio de sesión de la aplicación
  - Identificador de usuario efectivo del proceso si no hay ningún inicio de sesión disponible

  - El identificador de usuario asociado a la transacción, si la aplicación es una transacción CICS
- En sistemas Windows , el gestor de colas utiliza los primeros 12 caracteres del nombre de usuario conectado.

Normalmente, este campo es un campo de salida generado por el gestor de colas, pero para una llamada MQPUT o MQPUT1 puede hacer que este campo sea un campo de entrada/salida y especificar el campo UserIdentification en lugar de dejar que el gestor de colas genere esta información. Especifique MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT en el parámetro Opts PutMsgy especifique un ID de usuario en el campo UserIdentifier si no desea que el gestor de colas genere el campo UserIdentifier para una llamada MQPUT o MQPUT1 .

Para las llamadas MQPUT y MQPUT1 , es un campo de entrada/salida si se especifica MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT en el parámetro *PutMsgOpts* . Se descarta cualquier información que siga a un carácter nulo dentro del campo. El gestor de colas convierte el carácter nulo y los caracteres siguientes en espacios en blanco. Si no se especifica MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT, este campo se ignora en la entrada y es un campo de sólo salida.

Tras la finalización satisfactoria de una llamada MQPUT o MQPUT1 , este campo contiene el *UserIdentifier* que se ha transmitido con el mensaje si se ha colocado en una cola. Este será el valor de *UserIdentifier* que se mantiene con el mensaje si se conserva (consulte la descripción de MQPMO_RETAIN para obtener más detalles sobre las publicaciones retenidas) pero no se utiliza como *UserIdentifier* cuando el mensaje se envía como publicación a los suscriptores porque proporcionan un valor para alterar temporalmente *UserIdentifier* en todas las publicaciones que se les envían. Si el mensaje no tiene contexto, el campo está totalmente en blanco.

Es un campo de salida para la llamada MQGET. La longitud de este campo la proporciona MQ_USER_ID_LENGTH. El valor inicial de este campo es la serie nula en C y 12 caracteres en blanco en otros lenguajes de programación.

Versión (MQLONG)

Este es el número de versión de la estructura y debe ser uno de los siguientes:

MQMD_VERSION_1
Estructura del descriptor de mensaje Version-1 .

Esta versión está soportada en todos los entornos.

MQMD_VERSION_2
Estructura del descriptor de mensaje Version-2 .

Esta versión está soportada en todos los entornos de WebSphere MQ V6.0 y posteriores, además de los clientes MQI de WebSphere MQ conectados a estos sistemas.

Nota: Cuando se utiliza un MQMD version-2 , el gestor de colas realiza comprobaciones adicionales en cualquier estructura de cabecera MQ que pueda estar presente al principio de los datos del mensaje de aplicación; para obtener más detalles, consulte las notas de uso de la llamada MQPUT.

Los campos que existen sólo en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQMD_CURRENT_VERSION
Versión actual de la estructura del descriptor de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es MQMD_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQMD

Tabla 515. Valores iniciales de los campos en MQMD para MQMD

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQMD_STRUC_ID	'MD'

Tabla 515. Valores iniciales de los campos en MQMD para MQMD (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
Version	MQMD_VERSION_1	1
Report	MQRO_NONE	0
MsgType	MQMT_DATAGRAM	8
Expiry	MQEI_UNLIMITED	-1
Feedback	MQFB_NONE	0
Encoding	MQENC_NATIVE	Depende del entorno
CodedCharSetId	MQCCSI_Q_MGR	0
Format	MQFMT_NONE	Espacios en blanco
Priority	MQPRI_PRIORITY_AS_Q_DEF	-1
Persistence	MQPER_PERSISTENCE_AS_Q_DEF	2
MsgId	MQMI_NONE	Nulos
CorrelId	MQCI_NONE	Nulos
BackoutCount	Ninguna	0
ReplyToQ	Ninguna	Serie nula o espacios en blanco
ReplyToQMgr	Ninguna	Serie nula o espacios en blanco
UserIdentifier	Ninguna	Serie nula o espacios en blanco
AccountingToken	MQACT_NONE	Nulos
ApplIdentityData	Ninguna	Serie nula o espacios en blanco
PutApplType	MQAT_NO_CONTEXT	0
PutApplName	Ninguna	Serie nula o espacios en blanco
PutDate	Ninguna	Serie nula o espacios en blanco
PutTime	Ninguna	Serie nula o espacios en blanco
ApplOriginData	Ninguna	Serie nula o espacios en blanco
GroupId	MQGI_NONE	Nulos
MsgSeqNumber	Ninguna	1
Offset	Ninguna	0
MsgFlags	MQMF_NONE	0
OriginalLength	MQOL_UNDEFINED	-1

Tabla 515. Valores iniciales de los campos en MQMD para MQMD (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**

Notas:

1. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
2. En el lenguaje de programación C, la variable de macroMQMD_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQMD MyMD = {MQMD_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

MQMDE-Extensión de descriptor de mensaje

La tabla siguiente resume los campos de la estructura.

Tabla 516. Campos en MQMDE

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
StrucLength	Longitud de la estructura MQMDE	StrucLength
Encoding	Codificación numérica de los datos que siguen a MQMDE	Codificación
CodedCharSetId	Identificador de juego de caracteres de los datos que siguen a MQMDE	CodedCharSetId
Format	Nombre de formato de los datos que siguen a MQMDE	Formato
Flags	Distintivos generales	Distintivos
GroupId	Identificador de grupo	GroupId
MsgSeqNumber	Número de secuencia del mensaje lógico en el grupo	MsgSeqNumber
Offset	Desplazamiento de datos en el mensaje físico desde el inicio del mensaje lógico	desplazamiento
MsgFlags	Distintivos de mensajes	MsgFlags
OriginalLength	Longitud del mensaje original	OriginalLength

Visión general de MQMDE

Disponibilidad: todos los sistemas WebSphere MQ , además de los clientes WebSphere MQ conectados a estos sistemas.

Finalidad: la estructura MQMDE describe los datos que a veces se producen antes de los datos del mensaje de aplicación. La estructura contiene los campos MQMD que existen en el MQMD version-2 , pero no en el MQMD version-1 .

Nombre de formato: MQFMT_MD_EXTENSION.

Juego de caracteres y codificación: Los datos de MQMDE deben estar en el juego de caracteres y la codificación del gestor de colas local; los proporciona el atributo de gestor de colas *CodedCharSetId* y MQENC_NATIVE para el lenguaje de programación C.

Establezca el juego de caracteres y la codificación de MQMDE en los campos *CodedCharSetId* y *Encoding* en:

- El MQMD (si la estructura MQMDE está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQMDE (todos los demás casos).

Si el MQMDE no está en el juego de caracteres y la codificación del gestor de colas, el MQMDE se acepta pero no se respeta, es decir, el MQMDE se trata como datos de mensaje.

Nota: En Windows, las aplicaciones compiladas con Micro Focus COBOL utilizan un valor de MQENC_NATIVE que es diferente de la codificación del gestor de colas. Aunque los campos numéricos de la estructura MQMD en las llamadas MQPUT, MQPUT1y MQGET deben estar en la codificación Micro Focus COBOL, los campos numéricos de la estructura MQMDE deben estar en la codificación del gestor de colas. Esto último lo proporciona MQENC_NATIVE para el lenguaje de programación C, y tiene el valor 546.

Uso: las aplicaciones que utilizan un MQMD version-2 no encontrarán una estructura MQMDE. Sin embargo, las aplicaciones especializadas y las aplicaciones que siguen utilizando un MQMD de version-1 , pueden encontrar un MQMDE en algunas situaciones. La estructura MQMDE puede producirse en las circunstancias siguientes:

- Especificado en las llamadas MQPUT y MQPUT1
- Devuelto por la llamada MQGET
- En mensajes en colas de transmisión

MQMDE especificado en las llamadas MQPUT y MQPUT1: en las llamadas MQPUT y MQPUT1 , si la aplicación proporciona un MQMD version-1 , la aplicación puede, opcionalmente, añadir un prefijo a los datos del mensaje con un MQMDE, estableciendo el campo *Format* de MQMD en MQFMT_MD_EXTENSION para indicar que existe un MQMDE. Si la aplicación no proporciona un MQMDE, el gestor de colas asume los valores predeterminados para los campos de MQMDE. Los valores predeterminados que utiliza el gestor de colas son los mismos que los valores iniciales para la estructura; consulte Tabla 518 en la página 449.

Si la aplicación proporciona un version-2 MQMD *y* prefija los datos de mensaje de aplicación con un MQMDE, las estructuras se procesan tal como se muestra en la Tabla 517 en la página 446.

Tabla 517. Acción del gestor de colas cuando se especifica MQMDE en MQPUT o MQPUT1 para MQMDE

MQMD Version	Valores de los campos version-2	Valores de los campos correspondientes en MQMDE	Acción realizada por el gestor de colas
1	-	Válido	MQMDE se respeta
2	Valor predeterminado	Válido	MQMDE se respeta
2	No predeterminado	Válido	MQMDE se trata como datos de mensaje
1 o 2	Cualquiera	No válido	La llamada falla con un código de razón adecuado
1 o 2	Cualquiera	MQMDE está en el juego de caracteres o codificación incorrectos, o es una versión no soportada	MQMDE se trata como datos de mensaje

Nota: En z/OS, si la aplicación especifica un MQMD version-1 con un MQMDE, el gestor de colas valida el MQMDE sólo si la cola tiene un *IndexType* de MQIT_GROUP_ID.

Hay un caso especial. Si la aplicación utiliza un MQMD version-2 para colocar un mensaje que es un segmento (es decir, se ha establecido el distintivo MQMF_SEGMENT o MQMF_LAST_SEGMENT), y el nombre de formato del MQMD es MQFMT_DEAD_LETTER_HEADER, el gestor de colas genera una estructura MQMDE y la inserta *entre* la estructura MQDLH y los datos que le siguen. En el MQMD que el gestor de colas conserva con el mensaje, los campos version-2 se establecen en sus valores predeterminados.

Varios de los campos que existen en MQMD version-2 pero no en MQMD version-1 son campos de entrada/salida en MQPUT y MQPUT1. Sin embargo, el gestor de colas *no* devuelve ningún valor en los campos equivalentes de MQMDE en la salida de las llamadas MQPUT y MQPUT1 ; si la aplicación requiere estos valores de salida, debe utilizar un MQMD version-2 .

MQMDE devuelto por llamada MQGET: en la llamada MQGET, si la aplicación proporciona un MQMD version-1 , el gestor de colas prefija el mensaje devuelto con un MQMDE, pero sólo si uno o varios de los campos del MQMDE tienen un valor no predeterminado. El gestor de colas establece el campo *Format* en MQMD en el valor MQFMT_MD_EXTENSION para indicar que hay un MQMDE presente.

Si la aplicación proporciona un MQMDE al inicio del parámetro *Buffer* , el MQMDE se ignora. Al volver de la llamada MQGET, se sustituye por el MQMDE para el mensaje (si es necesario), o se sobrescribe por los datos del mensaje de aplicación (si el MQMDE no es necesario).

Si la llamada MQGET devuelve un MQMDE, los datos del MQMDE suelen estar en el juego de caracteres y la codificación del gestor de colas. Sin embargo, MQMDE puede estar en algún otro juego de caracteres y codificación si:

- MQMDE se ha tratado como datos en la llamada MQPUT o MQPUT1 (consulte Tabla 517 en la página 446 para conocer las circunstancias que pueden causar esto).
- El mensaje se ha recibido de un gestor de colas remoto conectado mediante una conexión TCP y el agente de canal de mensajes (MCA) receptor no se ha configurado correctamente.

Nota: En Windows, las aplicaciones compiladas con Micro Focus COBOL utilizan un valor de MQENC_NATIVE que es diferente de la codificación del gestor de colas (consulte más arriba).

MQMDE en mensajes en colas de transmisión: los mensajes en colas de transmisión llevan como prefijo la estructura MQXQH, que contiene un MQMD version-1 . También puede estar presente un MQMDE, situado entre la estructura MQXQH y los datos del mensaje de aplicación, pero normalmente solo está presente si uno o varios de los campos de MQMDE tienen un valor no predeterminado.

También se pueden producir otras estructuras de cabecera MQ entre la estructura MQXQH y los datos del mensaje de aplicación. Por ejemplo, cuando la cabecera de mensaje no entregado MQDLH está presente y el mensaje no es un segmento, el orden es:

- MQXQH (que contiene un MQMD version-1)
- MQMDE
- MQDLH
- datos de mensaje de aplicación

Campos para MQMDE

La estructura MQMDE contiene los campos siguientes; los campos se describen en **orden alfabético**:

CodedCharSetId (MQLONG)

Especifica el identificador de juego de caracteres de los datos que siguen a la estructura MQMDE; no se aplica a los datos de tipo carácter de la propia estructura MQMDE.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. El gestor de colas no comprueba que este campo sea válido. Se puede utilizar el siguiente valor especial:

MQCCSI_INHERIT
Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

MQCCSI_INHERIT no se puede utilizar si el valor del campo *PutApplType* en MQMD es MQAT_BROKER.

Este valor está soportado en los entornos siguientes: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más clientes WebSphere MQ conectados a estos sistemas.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Encoding (MQLONG)

Especifica la codificación numérica de los datos que siguen a la estructura MQMDE; no se aplica a los datos numéricos de la propia estructura MQMDE.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. El gestor de colas no comprueba que el campo sea válido. Consulte el campo *Encoding* descrito en "MQMD - Descriptor de mensaje" en la página 392 para obtener más información sobre las codificaciones de datos.

El valor inicial de este campo es MQENC_NATIVE.

Flags (MQLONG)

Se puede especificar el distintivo siguiente:

MQMDEF_NONE
Sin distintivos.

El valor inicial de este campo es MQMDEF_NONE.

Format (MQCHAR8)

Especifica el nombre de formato de los datos que siguen a la estructura MQMDE.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. El gestor de colas no comprueba que este campo sea válido. Consulte el campo *Format* descrito en "MQMD - Descriptor de mensaje" en la página 392 para obtener más información sobre los nombres de formato.

El valor inicial de este campo es MQFMT_NONE.

GroupId (MQBYTE24)

Consulte el campo *GroupId* descrito en "MQMD - Descriptor de mensaje" en la página 392. El valor inicial de este campo es MQGI_NONE.

MsgFlags (MQLONG)

Consulte el campo *MsgFlags* descrito en "MQMD - Descriptor de mensaje" en la página 392. El valor inicial de este campo es MQMF_NONE.

MsgSeqNumber (MQLONG)

Consulte el campo *MsgSeqNumber* descrito en "MQMD - Descriptor de mensaje" en la página 392. El valor inicial de este campo es 1.

Offset (MQLONG)

Consulte el campo *Offset* descrito en "MQMD - Descriptor de mensaje" en la página 392. El valor inicial de este campo es 0.

OriginalLength (MQLONG)

Consulte el campo *OriginalLength* descrito en “MQMD - Descriptor de mensaje” en la página 392. El valor inicial de este campo es MQOL_UNDEFINED.

StrucId (MQCHAR4)

El valor debe ser:

MQMDE_STRUC_ID
Identificador de la estructura de extensión del descriptor de mensaje.

Para el lenguaje de programación C, también se define la constante MQMDE_STRUC_ID_ARRAY; tiene el mismo valor que MQMDE_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQMDE_STRUC_ID.

StrucLength (MQLONG)

Es la longitud de la estructura MQMDE; se define el valor siguiente:

MQMDE_LENGTH_2
Longitud de la estructura de extensión del descriptor de mensaje version-2 .

El valor inicial de este campo es MQMDE_LENGTH_2.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQMDE_VERSION_2
Estructura de extensión del descriptor de mensaje Version-2 .

La constante siguiente especifica el número de versión de la versión actual:

MQMDE_CURRENT_VERSION
Versión actual de la estructura de extensión del descriptor de mensaje.

El valor inicial de este campo es MQMDE_VERSION_2.

Valores iniciales y declaraciones de lenguaje para MQMDE

Tabla 518. Valores iniciales de los campos en MQMDE para MQMDE

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQMDE_STRUC_ID	'MDE¬'
Version	MQMDE_VERSION_2	2
StrucLength	MQMDE_LENGTH_2	72
Encoding	MQENC_NATIVE	Depende del entorno
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Espacios en blanco
Flags	MQMDEF_NONE	0
GroupId	MQGI_NONE	Nulos
MsgSeqNumber	Ninguna	1
Offset	Ninguna	0
MsgFlags	MQMF_NONE	0
OriginalLength	MQOL_UNDEFINED	-1

Tabla 518. Valores iniciales de los campos en MQMDE para MQMDE (continuación)

Nombre de campo	Nombre de constante	Valor de constante

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQMDE_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQMDE MyMDE = {MQMDE_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

MQMHBO-Descriptor de mensaje para opciones de almacenamiento intermedio

La tabla siguiente resume los campos de la estructura. Estructura MQMHBO de -Descriptor de mensaje para opciones de almacenamiento intermedio

Tabla 519. Campos en MQMHBO		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Options	Opciones que controlan la acción de MQMHBUF	Opciones

Visión general de MQMHBO

Disponibilidad: todos los sistemas WebSphere MQ y clientes MQI de WebSphere MQ .

Finalidad: la estructura MQMHBO permite a las aplicaciones especificar opciones que controlan cómo se generan los almacenamientos intermedios a partir de los manejadores de mensajes. La estructura es un parámetro de entrada en la llamada MQMHBUF.

Juego de caracteres y codificación: Los datos de MQMHBO deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (MQENC_NATIVE).

Campos para MQMHBO

Descriptor de contexto de mensaje para campos de estructura de opciones de almacenamiento intermedio

La estructura MQMHBO contiene los campos siguientes; los campos se describen en **orden alfabético**:

Opciones (MQLONG)
Descriptor de contexto de mensaje para estructura de opciones de almacenamiento intermedio-Campo Opciones

Estas opciones controlan la acción de MQMHBUF.

Debe especificar la opción siguiente:

MQMHBO_PROPERTIES_IN_MQRFH2
Al convertir propiedades de un descriptor de mensaje en un almacenamiento intermedio, conviértalas al formato MQRFH2 .

Opcionalmente, también puede especificar el valor siguiente. Si los valores necesarios pueden ser:

- Sumado (no añadir la misma constante más de una vez), o
- Combinado utilizando la operación OR bit a bit (si el lenguaje de programación da soporte a operaciones de bits).

MQMHBO_DELETE_PROPERTIES
Las propiedades que se añaden al almacenamiento intermedio se suprimen del descriptor de contexto de mensaje. Si la llamada falla, no se suprimen las propiedades.

Siempre es un campo de entrada. El valor inicial de este campo es MQMHBO_PROPERTIES_IN_MQRFH2.

StrucId (MQCHAR4)
Estructura de descriptor de contexto de mensaje para opciones de almacenamiento intermedio-campo StrucId

Es el identificador de estructura. El valor debe ser:

MQMHBO_STRUC_ID
Identificador del descriptor de mensaje para la estructura de opciones de almacenamiento intermedio.

Para el lenguaje de programación C, también se define la constante MQMHBO_STRUC_ID_ARRAY; tiene el mismo valor que MQMHBO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQMHBO_STRUC_ID.

Versión (MQLONG)
Descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio-Campo Versión

Es el número de versión de la estructura. El valor debe ser:

MQMHBO_VERSION_1
Número de versión del descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio.

La constante siguiente especifica el número de versión de la versión actual:

MQMHBO_CURRENT_VERSION
Versión actual del descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio.

Siempre es un campo de entrada. El valor inicial de este campo es MQMHBO_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQMHBO
Descriptor de contexto de mensaje para estructura de almacenamiento intermedio-Valores iniciales

Tabla 520. Valores iniciales de los campos en MQMHBO

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQMHBO_STRUC_ID	'MHBO'
Version	MQMHBO_VERSION_1	1
Options	MQMHBO_PROPERTIES_IN_MQRFH2	

Notas:

1. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
2. En el lenguaje de programación C, la variable de macroMQMHBO_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQMHBO MyMHBO = {MQMHBO_DEFAULT};
```

Declaración C
Descriptor de contexto de mensaje para estructura de opciones de almacenamiento intermedio-Declaración de lenguaje C

```
typedef struct tagMQMHBO MQMHBO;
struct tagMQMHBO {
  MQCHAR4  StrucId;        /* Structure identifier */
  MQLONG   Version;        /* Structure version number */
  MQLONG   Options;        /* Options that control the action of
                              MQMHBUF */
};
```

declaración COBOL
Descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio-Declaración de lenguaje COBOL

```
**   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.
```

Declaración PL/I
Descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio-Declaración de lenguaje PL/I

```
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 */
```

Declaración High Level Assembler
Descriptor de contexto de mensaje para estructura de opciones de almacenamiento intermedio-Declaración de lenguaje Assembler

```
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)
```

MQOD-Descriptor de objeto

La tabla siguiente resume los campos de la estructura.

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
ObjectType	Tipo de objeto	ObjectType
ObjectName	Nombre de objeto	ObjectName
ObjectQMgrName	Nombre del gestor de colas de objetos	ObjectQMgrName
DynamicQName	Nombre de la cola dinámica	DynamicQName
AlternateUserId	Identificador de usuario alternativo	AlternateUserId
Nota: Los campos restantes se ignoran si *Version* es menor que MQOD_VERSION_2.		
RecsPresent	Número de registros de objeto presentes	RecsPresent
KnownDestCount	Número de colas locales abiertas satisfactoriamente.	KnownDestCount
UnknownDestCount	Número de colas remotas abiertas satisfactoriamente.	UnknownDestCount
InvalidDestCount	Número de colas que no se han podido abrir.	InvalidDestCount
ObjectRecOffset	Desplazamiento del primer registro de objeto desde el principio de MQOD	Desplazamiento deObjectRec
ResponseRecOffset	Desplazamiento del primer registro de respuesta desde el principio de MQOD	ResponseRecOffset
ObjectRecPtr	Dirección del primer registro de objeto	ObjectRecPtr

Campo	Descripción	Tema
ResponseRecPtr	Dirección del primer registro de respuesta	ResponseRecPtr
Nota: Los campos restantes se ignoran si *Version* es menor que MQOD_VERSION_3.		
AlternateSecurityId	Identificador de seguridad alternativo	AlternateSecurityId
ResolvedQName	Nombre de cola resuelto	ResolvedQName
ResolvedQMgrName	Nombre del gestor de colas resuelto	ResolvedQMgrName
Nota: Los campos restantes se ignoran si *Version* es menor que MQOD_VERSION_4.		
ObjectString	Nombre de objeto largo	ObjectString
SelectionString	Serie de selección	SelectionString
ResObjectString	Nombre de objeto largo resuelto	ResObjectString
ResolvedType	Tipo de objeto resuelto	ResolvedType

Visión general de MQOD

Disponibilidad: todos los sistemas WebSphere MQ , más WebSphere MQ clientes MQI conectados a dichos sistemas.

Finalidad: la estructura MQOD se utiliza para especificar un objeto por nombre. Son válidos los siguientes tipos de objeto:

- Cola o lista de distribución
- Lista de nombres
- Definición de proceso
- Gestor de colas
- Tema

La estructura es un parámetro de entrada/salida en las llamadas MQOPEN y MQPUT1 .

Versión: La versión actual de MQOD es MQOD_VERSION_4. Las aplicaciones que desee portar entre varios entornos deben asegurarse de que la versión necesaria de MQOD esté soportada en todos los entornos afectados. Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones siguientes.

Los archivos de cabecera, COPY e INCLUDE proporcionados para los lenguajes de programación soportados contienen la versión más reciente de MQOD soportada por el entorno, pero con el valor inicial del campo *Version* establecido en MQOD_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1 , la aplicación debe establecer el campo *Version* en el número de versión de la versión necesaria.

Para abrir una lista de distribución, *Version* debe ser MQOD_VERSION_2 o superior.

Conjunto de caracteres y codificación: los datos de MQOD deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos para MQOD

La estructura MQOD contiene los campos siguientes; los campos se describen en **orden alfabético**:

AlternateSecurityId (MQBYTE40)

Se trata de un identificador de seguridad que se pasa con *AlternateUserId* al servicio de autorización para permitir que se realicen las comprobaciones de autorización adecuadas. *AlternateSecurityId* sólo se utiliza si:

- Se ha especificado MQOO_ALTERNATE_USER_AUTHORITY en la llamada MQOPEN, o
- Se ha especificado MQPMO_ALTERNATE_USER_AUTHORITY en la llamada MQPUT1 ,

y el campo *AlternateUserId* no está completamente en blanco hasta el primer carácter nulo o el final del campo.

En Windows, *AlternateSecurityId* se puede utilizar para proporcionar el identificador de seguridad (SID) de Windows que identifica de forma exclusiva el *AlternateUserId*. El SID de un usuario se puede obtener del sistema Windows utilizando la llamada de API LookupAccountName() Windows .

En z/OS, este campo se ignora.

El campo *AlternateSecurityId* tiene la estructura siguiente:

- El primer byte es un entero binario que contiene la longitud de los datos significativos que siguen; el valor excluye el propio byte de longitud. Si no hay ningún identificador de seguridad, la longitud es cero.
- El segundo byte indica el tipo de identificador de seguridad que está presente; son posibles los valores siguientes:

  MQSIDT_NT_SECURITY_ID
  Identificador de seguridad de Windows .

  MQSIDT_NONE
  Sin identificador de seguridad.
- El tercer byte y los bytes subsiguientes hasta la longitud definida por el primer byte contienen el propio identificador de seguridad.
- Los bytes restantes en el campo se establecen en cero binario.

Puede utilizar el siguiente valor especial:

MQSID_NONE
No se ha especificado ningún identificador de seguridad.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQSID_NONE_ARRAY; tiene el mismo valor que MQSID_NONE, pero es una matriz de caracteres en lugar de una serie.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_SECURITY_ID_LENGTH. El valor inicial de este campo es MQSID_NONE. Este campo se ignora si *Version* es menor que MQOD_VERSION_3.

AlternateUserId (MQCHAR12)

Si especifica MQOO_ALTERNATE_USER_AUTHORITY para la llamada MQOPEN, o MQPMO_ALTERNATE_USER_AUTHORITY para la llamada MQPUT1 , este campo contiene un identificador de usuario alternativo que se utiliza para comprobar la autorización para la apertura, en lugar del identificador de usuario bajo el que se ejecuta actualmente la aplicación. Sin embargo, algunas comprobaciones se siguen realizando con el identificador de usuario actual (por ejemplo, comprobaciones de contexto).

Si se especifica MQOO_ALTERNATE_USER_AUTHORITY o MQPMO_ALTERNATE_USER_AUTHORITY y este campo está totalmente en blanco hasta el primer carácter nulo o el final del campo, la apertura sólo puede realizarse correctamente si no se necesita autorización de usuario para abrir este objeto con las opciones especificadas.

Si no se especifica MQOO_ALTERNATE_USER_AUTHORITY ni MQPMO_ALTERNATE_USER_AUTHORITY, este campo se ignora.

Existen las siguientes diferencias en los entornos indicados:

- En z/OS, solo se utilizan los primeros 8 caracteres de *AlternateUserId* para comprobar la autorización de la apertura. Sin embargo, el identificador de usuario actual debe estar autorizado para especificar este identificador de usuario alternativo concreto; para esta comprobación se utilizan los 12 caracteres del identificador de usuario alternativo. El identificador de usuario sólo debe contener caracteres permitidos por el gestor de seguridad externa.

  Si se especifica *AlternateUserId* para una cola, el gestor de colas puede utilizar posteriormente el valor cuando se colocan los mensajes. Si las opciones MQPMO_ * _CONTEXT especificadas en la llamada MQPUT o MQPUT1 hacen que el gestor de colas genere la información de contexto de identidad, el gestor de colas coloca *AlternateUserId* en el campo *UserIdentifier* en el MQMD del mensaje, en lugar del identificador de usuario actual.
- En otros entornos, *AlternateUserId* sólo se utiliza para comprobaciones de control de acceso en el objeto que se está abriendo. Si el objeto es una cola, *AlternateUserId* no afecta al contenido del campo *UserIdentifier* en el MQMD de los mensajes enviados utilizando ese manejador de cola.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_USER_ID_LENGTH. El valor inicial de este campo es la serie nula en C y 12 caracteres en blanco en otros lenguajes de programación.

DynamicQName (MQCHAR48)

Es el nombre de una cola dinámica que va a crear la llamada MQOPEN. Esto sólo es relevante cuando *ObjectName* especifica el nombre de una cola modelo; en todos los demás casos, se ignora *DynamicQName* .

Los caracteres que son válidos en el nombre son los mismos que los de *ObjectName*, excepto que también es válido un asterisco. Un nombre que esté en blanco (o uno en el que sólo haya espacios en blanco antes del primer carácter nulo) no es válido si *ObjectName* es el nombre de una cola modelo.

Si el último carácter no en blanco del nombre es un asterisco (*), el gestor de colas sustituye el asterisco por una serie de caracteres que garantiza que el nombre generado para la cola es exclusivo en el gestor de colas local. Para permitir un número suficiente de caracteres para esto, el asterisco sólo es válido en las posiciones de la 1 a la 33. No debe haber caracteres que no sean espacios en blanco o un carácter nulo detrás del asterisco.

Es válido que el asterisco aparezca en la primera posición de carácter, en cuyo caso el nombre consta únicamente de los caracteres generados por el gestor de colas.

En z/OS, no utilice un nombre con el asterisco en la primera posición de carácter, ya que no puede haber comprobaciones de seguridad realizadas en una cola con un nombre completo que se genere automáticamente.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo lo determina el entorno:

- En z/OS, el valor es `'CSQ.*'`.
- En otras plataformas, el valor es `'AMQ.*'`.

El valor es una serie terminada en nulo en C y una serie rellenada en blanco en otros lenguajes de programación.

Recuento de InvalidDest(MQLONG)

Es el número de colas de la lista de distribución que no se han podido abrir correctamente. Si está presente, este campo también se establece al abrir una sola cola que no está en una lista de distribución.

Nota: Si está presente, este campo se establece *solo* si el parámetro *CompCode* de la llamada MQOPEN o MQPUT1 es MQCC_OK o MQCC_WARNING; *no* se establece si el parámetro *CompCode* es MQCC_FAILED.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_1.

Recuento de KnownDest(MQLONG)

Es el número de colas de la lista de distribución que se resuelven en colas locales y que se han abierto correctamente. El recuento no incluye las colas que se resuelven en colas remotas (aunque inicialmente se utilice una cola de transmisión local para almacenar el mensaje). Si está presente, este campo también se establece al abrir una sola cola que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_1.

ObjectName (MQCHAR48)

Es el nombre local del objeto tal como se define en el gestor de colas identificado por *ObjectQMgrName*. El nombre puede contener los siguientes caracteres:

- Caracteres alfabéticos en mayúsculas (de la A a la Z)
- Caracteres alfabéticos en minúsculas (de la a a la z)
- Dígitos numéricos (de 0 a 9)
- Punto (.), barra inclinada (/), subrayado (_), porcentaje (%)

El nombre no debe contener espacios en blanco iniciales o intercalados, pero puede contener espacios en blanco finales. Utilice un carácter nulo para indicar el final de los datos significativos en el nombre; el nulo y los caracteres que le siguen se tratan como espacios en blanco. Las restricciones siguientes se aplican en los entornos indicados:

- En sistemas que utilizan EBCDIC Katakana, no se pueden utilizar caracteres en minúsculas.
- En z/OS:
  - Evite los nombres que empiezan o terminan con un guión bajo; no pueden ser procesados por las operaciones y los paneles de control.
  - El carácter de porcentaje tiene un significado especial para RACF. Si se utiliza RACF como gestor de seguridad externa, los nombres no deben contener el porcentaje. Si lo hacen, estos nombres no se incluyen en ninguna comprobación de seguridad cuando se utilizan perfiles genéricos RACF .
- En IBM i, los nombres que contienen caracteres en minúsculas, barras inclinadas o porcentaje deben estar entre comillas cuando se especifican en los mandatos. Estas comillas no se deben especificar para nombres que aparecen como campos en estructuras o como parámetros en llamadas.

El nombre completo del tema se puede crear a partir de dos campos diferentes: *ObjectName* y *ObjectString*. Para obtener detalles sobre cómo se utilizan estos dos campos, consulte "Utilización de series de tema" en la página 558.

Los siguientes puntos se aplican a los tipos de objeto indicados:

- Si *ObjectName* es el nombre de una cola modelo, el gestor de colas crea una cola dinámica con los atributos de la cola modelo y devuelve en el campo *ObjectName* el nombre de la cola creada. Una cola modelo sólo se puede especificar en la llamada MQOPEN; una cola modelo no es válida en la llamada MQPUT1 .
- Si *ObjectName* es el nombre de una cola alias con TARGTYPE (TOPIC), primero se realiza una comprobación de seguridad en la cola alias con nombre; esto es normal cuando se utilizan colas alias. Cuando la comprobación de seguridad se completa correctamente, la llamada MQOPEN continuará y se comportará como una llamada MQOPEN en un MQOT_TOPIC; esto incluye realizar una comprobación de seguridad en el objeto de tema administrativo.
- Si *ObjectName* y *ObjectQMgrName* identifican una cola compartida propiedad del grupo de compartición de colas al que pertenece el gestor de colas local, no debe haber también una definición de cola con el mismo nombre en el gestor de colas local. Si existe una definición de este tipo (una cola local, una cola alias, una cola remota o una cola modelo), la llamada falla con el código de razón MQRC_OBJECT_NOT_UNIQUE.
- Si el objeto que se está abriendo es una lista de distribución (es decir, *RecsPresent* está presente y es mayor que cero), *ObjectName* debe estar en blanco o la serie nula. Si no se cumple esta condición, la llamada falla con el código de razón MQRC_OBJECT_NAME_ERROR.

- Si *ObjectType* es MQOT_Q_MGR, se aplican reglas especiales; en este caso, el nombre debe estar completamente en blanco hasta el primer carácter nulo o el final del campo.

Se trata de un campo de entrada/salida para la llamada MQOPEN cuando *ObjectName* es el nombre de una cola modelo y un campo de sólo entrada en todos los demás casos. La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ObjectQMgrNombre (MQCHAR48)

Es el nombre del gestor de colas en el que se define el objeto *ObjectName* . Los caracteres que son válidos en el nombre son los mismos que los de *ObjectName* (consulte "ObjectName (MQCHAR48)" en la página 458). Un nombre que está completamente en blanco hasta el primer carácter nulo o el final del campo indica el gestor de colas al que está conectada la aplicación (el gestor de colas local).

Los siguientes puntos se aplican a los tipos de objeto indicados:

- Si *ObjectType* es MQOT_TOPIC, MQOT_NAMELIST, MQOT_PROCESS o MQOT_Q_MGR, *ObjectQMgrName* debe estar en blanco o el nombre del gestor de colas local.
- Si *ObjectName* es el nombre de una cola modelo, el gestor de colas crea una cola dinámica con los atributos de la cola modelo y devuelve en el campo *ObjectQMgrName* el nombre del gestor de colas en el que se crea la cola; este es el nombre del gestor de colas local. Una cola modelo sólo se puede especificar en la llamada MQOPEN; una cola modelo no es válida en la llamada MQPUT1 .
- Si *ObjectName* es el nombre de una cola de clúster y *ObjectQMgrName* está en blanco, el destino de los mensajes enviados utilizando el descriptor de contexto de cola devuelto por la llamada MQOPEN es elegido por el gestor de colas (o la salida de carga de trabajo de clúster, si hay uno instalado) de la forma siguiente:
  - Si se especifica MQOO_BIND_ON_OPEN, el gestor de colas selecciona una instancia determinada de la cola de clúster al procesar la llamada MQOPEN, y todos los mensajes colocados utilizando este descriptor de contexto de cola se envían a dicha instancia.
  - Si se especifica MQOO_BIND_NOT_FIXED, el gestor de colas puede elegir una instancia diferente de la cola de destino (que reside en un gestor de colas diferente del clúster) para cada llamada MQPUT sucesiva que utilice este descriptor de contexto de cola.

  Si la aplicación necesita enviar un mensaje a una instancia *específica* de una cola de clúster (es decir, una instancia de cola que reside en un gestor de colas determinado del clúster), la aplicación debe especificar el nombre de dicho gestor de colas en el campo *ObjectQMgrName* . Esto fuerza al gestor de colas local a enviar el mensaje al gestor de colas de destino especificado.
- Si *ObjectName* es el nombre de una cola compartida quees propiedad del grupo de compartición de colas al que pertenece el gestor de colas local, *ObjectQMgrName* puede ser el nombre del grupo de compartición de colas, el nombre del gestor de colas local o en blanco; el mensaje se coloca en la misma cola cualquiera de estos valores especificados.

  Los grupos de compartición de colas solo están soportados en z/OS.
- Si *ObjectName* es el nombre de una cola compartida propiedad de un grupo de compartición de colas remoto (es decir, un grupo de compartición de colas al que *no* pertenece el gestor de colas local), *ObjectQMgrName* debe ser el nombre del grupo de compartición de colas. Puede utilizar el nombre de un gestor de colas que pertenece a ese grupo, pero esto puede retrasar el mensaje si ese gestor de colas concreto no está disponible cuando el mensaje llega al grupo de compartición de colas.
- Si el objeto que se está abriendo es una lista de distribución (es decir, *RecsPresent* es mayor que cero), *ObjectQMgrName* debe estar en blanco o la serie nula. Si esta condición no se cumple, la llamada falla con el código de razón MQRC_OBJECT_Q_MGR_NAME_ERROR.

Se trata de un campo de entrada/salida para la llamada MQOPEN cuando *ObjectName* es el nombre de una cola modelo y un campo de sólo entrada en todos los demás casos. La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

Desplazamiento de ObjectRec(MQLONG)

Es el desplazamiento en bytes del primer registro de objeto MQOR desde el inicio de la estructura MQOD. El desplazamiento puede ser positivo o negativo. *ObjectRecOffset* sólo se utiliza cuando se está abriendo una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Cuando se está abriendo una lista de distribución, se debe proporcionar una matriz de uno o más registros de objeto MQOR para especificar los nombres de las colas de destino en la lista de distribución. Esto se puede hacer de una de dos maneras:

- Utilizando el campo de desplazamiento *ObjectRecOffset*.

  En este caso, la aplicación debe declarar su propia estructura que contenga un MQOD seguido de la matriz de registros MQOR (con tantos elementos de matriz como sean necesarios) y establecer *ObjectRecOffset* en el desplazamiento del primer elemento de la matriz desde el inicio del MQOD. Asegúrese de que este desplazamiento sea correcto y tenga un valor que se pueda acomodar dentro de un MQLONG (el lenguaje de programación más restrictivo es COBOL, para el que el rango válido es de -999 999 999 a +999 999 999 999).

  Utilice *ObjectRecOffset* para lenguajes de programación que no soportan el tipo de datos de puntero, o que implementan el tipo de datos de puntero de una forma que no es portable a entornos diferentes (por ejemplo, el lenguaje de programación COBOL).

- Utilizando el campo de puntero *ObjectRecPtr*.

  En este caso, la aplicación puede declarar la matriz de estructuras MQOR por separado de la estructura MQOD y establecer *ObjectRecPtr* en la dirección de la matriz.

  Utilice *ObjectRecPtr* para lenguajes de programación que den soporte al tipo de datos de puntero de una forma que sea portable a distintos entornos (por ejemplo, el lenguaje de programación C).

Sea cual sea la técnica que elija, utilice uno de *ObjectRecOffset* y *ObjectRecPtr* ; la llamada falla con el código de razón MQRC_OBJECT_RECORDS_ERROR si ambos son cero, o ambos son distintos de cero.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_2.

ObjectRecPtr (MQPTR)

Es la dirección del primer registro de objeto MQOR. *ObjectRecPtr* sólo se utiliza cuando se está abriendo una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Puede utilizar *ObjectRecPtr* o *ObjectRecOffset* para especificar los registros de objeto, pero no ambos; consulte la descripción del campo *ObjectRecOffset* anterior para obtener más detalles. Si no utiliza *ObjectRecPtr*, establézcalo en el puntero nulo o en bytes nulos.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula. Este campo se ignora si *Version* es menor que MQOD_VERSION_2.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada, siendo el valor inicial la serie de bytes totalmente nula.

ObjectString (MQCHARV)
El campo ObjectString especifica el nombre de objeto largo.

Especifica el nombre de objeto largo que se va a utilizar. Sólo se hace referencia a este campo para determinados valores de *ObjectType*y se ignora para todos los demás valores. Consulte la descripción de *ObjectType* para obtener detalles de qué valores indican que se utiliza este campo.

Si *ObjectString* se especifica incorrectamente, de acuerdo con la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón MQRC_OBJECT_STRING_ERROR.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQCHARV.

El nombre completo del tema se puede crear a partir de dos campos diferentes: *ObjectName* y *ObjectString*. Para obtener detalles sobre cómo se utilizan estos dos campos, consulte "Utilización de series de tema" en la página 558.

ObjectType (MQLONG)

El tipo de objeto que se está nombrando en el descriptor de objeto. Los valores posibles son:

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente. El nombre del objeto se encuentra en el campo *ObjectName* .

MQOT_Q
Cola. El nombre del objeto se encuentra en el campo *ObjectName* .

MQOT_NAMELIST
Lista de nombres. El nombre del objeto se encuentra en el campo *ObjectName*

MQOT_PROCESS
. El nombre del objeto se encuentra en el campo *ObjectName*

MQOT_Q_MGR
Gestor de colas. El nombre del objeto se encuentra en el campo *ObjectName*

MQOT_TOPIC
. El nombre completo del tema se puede crear a partir de dos campos diferentes: *ObjectName* y *ObjectString*.

Para obtener detalles sobre cómo se utilizan estos dos campos, consulte "Utilización de series de tema" en la página 558.

Siempre es un campo de entrada. El valor inicial de este campo es MQOT_Q.

RecsPresent (MQLONG)

Es el número de registros de objeto MQOR que ha proporcionado la aplicación. Si este número es mayor que cero, indica que se está abriendo una lista de distribución, siendo *RecsPresent* el número de colas de destino de la lista. Una lista de distribución sólo puede contener un destino.

El valor de *RecsPresent* no debe ser menor que cero, y si es mayor que cero *ObjectType* debe ser MQOT_Q; la llamada falla con el código de razón MQRC_RECS_PRESENT_ERROR si no se cumplen estas condiciones.

En z/OS, este campo debe ser cero.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_2.

Serie ResObject(MQCHARV)
El campo ResObjectString es el nombre de objeto largo después de que el gestor de colas resuelva el nombre proporcionado en el campo *ObjectName* .

Este campo sólo se devuelve para temas y alias de cola que hacen referencia a un objeto de tema.

Si el nombre de objeto largo se proporciona en *ObjectString* y no se proporciona nada en *ObjectName*, el valor devuelto en este campo es el mismo que el proporcionado en *ObjectString*.

Si este campo se omite (es decir, ResObjectString.VSBufSize es cero), no se devolverá *ResObjectString* , pero la longitud se devolverá en ResObjectString.VSLength.

Si la longitud del almacenamiento intermedio (proporcionada en ResObjectStrng.VSBufSize) es más corta que la *ResObjectString*completa, la serie se truncará y devolverá tantos caracteres situados más a la derecha como quepa en el almacenamiento intermedio proporcionado.

Si *ResObjectString* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón MQRC_RES_OBJECT_STRING_ERROR.

ResolvedQMgrNombre (MQCHAR48)

Es el nombre del gestor de colas de destino después de que el gestor de colas local resuelva el nombre. El nombre devuelto es el nombre del gestor de colas que es propietario de la cola identificada por *ResolvedQName*. *ResolvedQMgrName* puede ser el nombre del gestor de colas local.

Si *ResolvedQName* es una cola compartida que es propiedad del grupo de compartición de colas al que pertenece el gestor de colas local, *ResolvedQMgrName* es el nombre del grupo de compartición de colas. Si la cola es propiedad de algún otro grupo de compartición de colas, *ResolvedQName* puede ser el nombre del grupo de compartición de colas o el nombre de un gestor de colas que es miembro del grupo de compartición de colas (la naturaleza del valor devuelto viene determinada por las definiciones de cola que existen en el gestor de colas local).

Sólo se devuelve un valor no en blanco si el objeto es una única cola abierta para examinar, entrar o salir (o cualquier combinación). Si el objeto abierto es cualquiera de los siguientes, *ResolvedQMgrName* se establece en blancos:

- No es una cola
- Una cola, pero no abierta para examinar, entrar o salir
- Una cola de clúster con MQOO_BIND_NOT_FIXED especificado (o con MQOO_BIND_AS_Q_DEF en vigor cuando el atributo de cola *DefBind* tiene el valor MQBND_BIND_NOT_FIXED)
- Una lista de distribución

Se trata de un campo de salida. La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación. Este campo se ignora si *Version* es menor que MQOD_VERSION_3.

ResolvedQName (MQCHAR48)

Es el nombre de la cola de destino después de que el gestor de colas local resuelva el nombre. El nombre devuelto es el nombre de una cola que existe en el gestor de colas identificado por *ResolvedQMgrName*.

Sólo se devuelve un valor no en blanco si el objeto es una única cola abierta para examinar, entrar o salir (o cualquier combinación). Si el objeto abierto es cualquiera de los siguientes, *ResolvedQName* se establece en blancos:

- No es una cola
- Una cola, pero no abierta para examinar, entrar o salir
- Una lista de distribución
- Una cola alias que hace referencia a un objeto de tema (consulte ResObjectString en su lugar).
- Una cola alias que se resuelve en un objeto de tema.

Se trata de un campo de salida. La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación. Este campo se ignora si *Version* es menor que MQOD_VERSION_3.

ResolvedType (MQLONG)
El tipo del objeto (base) resuelto que se está abriendo.

Los valores posibles son:

MQOT_Q
El objeto resuelto es una cola. Este valor se aplica cuando una cola se abre directamente o cuando se abre una cola alias que apunta a una cola.

MQOT_TOPIC
El objeto resuelto es un tema. Este valor se aplica cuando se abre un tema directamente o cuando se abre una cola alias que apunta a un objeto de tema.

MQOT_NONE
El tipo resuelto no es una cola ni un tema.

Desplazamiento de ResponseRec(MQLONG)

Es el desplazamiento en bytes del primer registro de respuesta MQRR desde el inicio de la estructura MQOD. El desplazamiento puede ser positivo o negativo. *ResponseRecOffset* sólo se utiliza cuando se está abriendo una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Cuando se está abriendo una lista de distribución, puede proporcionar una matriz de uno o más registros de respuesta MQRR para identificar las colas que no se han podido abrir (campo*CompCode* en MQRR) y la razón de cada anomalía (campo*Reason* en MQRR). Los datos se devuelven en la matriz de registros de respuesta en el mismo orden en que aparecen los nombres de cola en la matriz de registros de objeto. El gestor de colas establece los registros de respuesta sólo cuando el resultado de la llamada es mixto (es decir, algunas colas se han abierto correctamente mientras que otras han fallado, o todas han fallado pero por motivos diferentes); el código de razón MQRC_MULTIPLE_REASON de la llamada indica este caso. Si el mismo código de razón se aplica a todas las colas, esa razón se devuelve en el parámetro *Reason* de la llamada MQOPEN o MQPUT1 y los registros de respuesta no se establecen. Los registros de respuesta son opcionales, pero si se proporcionan, debe haber *RecsPresent* de ellos.

Los registros de respuesta se pueden proporcionar de la misma forma que los registros de objeto, ya sea especificando un desplazamiento en *ResponseRecOffset*, o especificando una dirección en *ResponseRecPtr*; consulte la descripción de *ObjectRecOffset* anterior para obtener detalles sobre cómo hacerlo. Sin embargo, no se puede utilizar más de uno de *ResponseRecOffset* y *ResponseRecPtr* ; la llamada falla con el código de razón MQRC_RESPONSE_RECORDS_ERROR si ambos son distintos de cero.

Para la llamada MQPUT1 , estos registros de respuesta se utilizan para devolver información sobre los errores que se producen cuando se envía el mensaje a las colas de la lista de distribución, así como los errores que se producen cuando se abren las colas. El código de terminación y el código de razón de la operación de transferencia para una cola sustituyen a los de la operación abierta para dicha cola sólo si el código de terminación de esta última era MQCC_OK o MQCC_WARNING.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_2.

ResponseRecPtr (MQPTR)

Es la dirección del primer registro de respuesta MQRR. *ResponseRecPtr* sólo se utiliza cuando se está abriendo una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Utilice *ResponseRecPtr* o *ResponseRecOffset* para especificar los registros de respuesta, pero no ambos; consulte la descripción del campo *ResponseRecOffset* anterior para obtener más detalles. Si no utiliza *ResponseRecPtr*, establézcalo en el puntero nulo o en bytes nulos.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula. Este campo se ignora si *Version* es menor que MQOD_VERSION_2.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada, siendo el valor inicial la serie de bytes totalmente nula.

SelectionString (MQCHARV)
Es la serie utilizada para proporcionar los criterios de selección utilizados al recuperar mensajes de una cola.

SelectionString no debe proporcionarse en los casos siguientes:

- Si *ObjectType* no es MQOT_Q
- Si la cola que se está abriendo no se está abriendo utilizando una de las opciones MQOO_BROWSE o MQOO_INPUT_ *

Si se proporciona *SelectionString* en estos casos, la llamada falla con el código de razón MQRC_SELECTOR_INVALID_FOR_TYPE.

Si *SelectionString* se especifica incorrectamente, según la descripción de cómo utilizar la estructura “MQCHARV-Serie de longitud variable” en la página 275 , o si supera la longitud máxima, la llamada falla

con el código de razón MQRC_SELECTION_STRING_ERROR. La longitud máxima de *SelectionString* es MQ_SELECTOR_LENGTH.

El uso de *SelectionString* se describe en Selectores.

StrucId (MQCHAR4)

Este es el identificador de estructura; el valor debe ser:

MQOD_ID_STRUCD
Identificador de la estructura de descriptor de objeto.

Para el lenguaje de programación C, también se define la constante MQOD_STRUC_ID_ARRAY; tiene el mismo valor que MQOD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQOD_STRUC_ID.

Recuento de UnknownDest(MQLONG)

Es el número de colas de la lista de distribución que se resuelven en colas remotas y que se han abierto correctamente. Si está presente, este campo también se establece al abrir una sola cola que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQOD_VERSION_1.

Versión (MQLONG)

Es el número de versión de la estructura; el valor debe ser uno de los siguientes:

MQOD_VERSION_1
Estructura del descriptor de objeto Version-1 .

MQOD_VERSION_2
Estructura del descriptor de objeto Version-2 .

MQOD_VERSION_3
Estructura del descriptor de objeto Version-3 .

MQOD_VERSION_4
Estructura del descriptor de objeto Version-4 .

Todas las versiones están soportadas en todos los entornos de WebSphere MQ V7.0 .

Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQOD_CURRENT_VERSION
Versión actual de la estructura de descriptor de objeto.

Siempre es un campo de entrada. El valor inicial de este campo es MQOD_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQOD

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQOD_ID_STRUCD	'OD¬¬'
Version	MQOD_VERSION_1	1
ObjectType	MQOT_Q	1
ObjectName	Ninguna	Serie nula o espacios en blanco
ObjectQMgrName	Ninguna	Serie nula o espacios en blanco
DynamicQName	Ninguna	'CSQ.*' en z/OS; 'AMQ.*' de lo contrario

Nombre de campo	Nombre de constante	Valor de constante
AlternateUserId	Ninguna	Serie nula o espacios en blanco
RecsPresent	Ninguna	0
KnownDestCount	Ninguna	0
UnknownDestCount	Ninguna	0
InvalidDestCount	Ninguna	0
ObjectRecOffset	Ninguna	0
ResponseRecOffset	Ninguna	0
ObjectRecPtr	Ninguna	Puntero nulo o bytes nulos
ResponseRecPtr	Ninguna	Puntero nulo o bytes nulos
AlternateSecurityId	MQSID_NONE	Nulos
ResolvedQName	Ninguna	Serie nula o espacios en blanco
ResolvedQMgrName	Ninguna	Serie nula o espacios en blanco
ObjectString	MQCHARV_PREDETERMINADO	Tal como se ha definido para MQCHARV
SelectionString	MQCHARV_PREDETERMINADO	Tal como se ha definido para MQCHARV
ResObjectString	MQCHARV_PREDETERMINADO	Tal como se ha definido para MQCHARV
ResolvedType	MQOT_NONE	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQOD_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQOD MyOD = {MQOD_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
** 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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

MQOR-Registro de objeto

La tabla siguiente resume los campos de la estructura.

Tabla 521. Campos en MQOR		
Campo	**Descripción**	**Tema**
ObjectName	Nombre de objeto	ObjectName
ObjectQMgrName	Nombre del gestor de colas de objetos	ObjectQMgrName

Visión general de MQOR

Disponibilidad: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más WebSphere MQ clientes MQI conectados a estos sistemas.

Finalidad: Utilice la estructura MQOR para especificar el nombre de cola y el nombre de gestor de colas de una única cola de destino. MQOR es una estructura de entrada para las llamadas MQOPEN y MQPUT1 .

Conjunto de caracteres y codificación: Los datos de MQOR deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Uso: al proporcionar una matriz de estas estructuras en la llamada MQOPEN, puede abrir una lista de colas; esta lista se denomina *lista de distribución*. Cada mensaje colocado utilizando el descriptor de contexto de cola devuelto por esa llamada MQOPEN se coloca en cada una de las colas de la lista, siempre que la cola se haya abierto correctamente.

Campos para MQOR

La estructura MQOR contiene los campos siguientes; los campos se describen en **orden alfabético**:

ObjectName (MQCHAR48)

Es el mismo que el campo *ObjectName* en la estructura MQOD (consulte MQOD para obtener detalles), excepto que:

- Debe ser el nombre de una cola.
- No debe ser el nombre de una cola modelo.

Siempre es un campo de entrada. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ObjectQMgrNombre (MQCHAR48)

Es el mismo que el campo *ObjectQMgrName* en la estructura MQOD (consulte MQOD para obtener más detalles).

Siempre es un campo de entrada. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

Valores iniciales y declaraciones de lenguaje para MQOR

Tabla 522. Valores iniciales de los campos en MQOR para MQOR		
Nombre de campo	**Nombre de constante**	**Valor de constante**
ObjectName	Ninguna	Serie nula o espacios en blanco
ObjectQMgrName	Ninguna	Serie nula o espacios en blanco

Tabla 522. Valores iniciales de los campos en MQOR para MQOR (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**

Notas:

1. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
2. En el lenguaje de programación C, la variable de macroMQOR_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQOR MyOR = {MQOR_DEFAULT};
```

Declaración C

```
typedef struct tagMQOR MQOR;
struct tagMQOR {
  MQCHAR48  ObjectName;      /* Object name */
  MQCHAR48  ObjectQMgrName;  /* Object queue manager name */
};
```

declaración COBOL

```
**   MQOR structure
  10 MQOR.
**    Object name
   15 MQOR-OBJECTNAME     PIC X(48).
**    Object queue manager name
   15 MQOR-OBJECTQMGRNAME PIC X(48).
```

Declaración PL/I

```
dcl
 1 MQOR based,
  3 ObjectName     char(48), /* Object name */
  3 ObjectQMgrName char(48); /* Object queue manager name */
```

Declaración de Visual Basic

```
Type MQOR
  ObjectName     As String*48 'Object name'
  ObjectQMgrName As String*48 'Object queue manager name'
End Type
```

MQPD-Descriptor de propiedad

La tabla siguiente resume los campos de la estructura.

Tabla 523. Campos en MQPD		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Options	Opciones	Opciones
Support	Soporte necesario para la propiedad de mensaje	Soporte

Tabla 523. Campos en MQPD (continuación)		
Campo	**Descripción**	**Tema**
Context	Contexto de mensaje al que pertenece la propiedad	Contexto
CopyOptions	Opciones de copia a las que pertenece la propiedad	CopyOptions

Visión general de MQPD

Disponibilidad: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS y clientes MQI de WebSphere MQ .

Finalidad: **MQPD** se utiliza para definir los atributos de una propiedad. La estructura es un parámetro de entrada/salida en la llamada MQSETMP y un parámetro de salida en la llamada MQINQMP.

Juego de caracteres y codificación: Los datos de **MQPD** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (**MQENC_NATIVE**).

Campos para MQPD

La estructura MQPD contiene los campos siguientes; los campos se describen en **orden alfabético**:

Contexto (MQLONG)

Describe a qué contexto de mensaje pertenece la propiedad.

Cuando un gestor de colas recibe un mensaje que contiene una propiedad definida por WebSphere MQque el gestor de colas reconoce como incorrecta, el gestor de colas corrige el valor del campo *Context* .

Se puede especificar la opción siguiente:

CONTEXTO_USUARIO_MQPDD
La propiedad está asociada al contexto de usuario.

No se requiere ninguna autorización especial para poder establecer una propiedad asociada al contexto de usuario utilizando la llamada MQSETMP.

En un gestor de colas WebSphere MQ Versión 7.0 , una propiedad asociada con el contexto de usuario se guarda tal como se describe para MQOO_SAVE_ALL_CONTEXT. Una llamada MQPUT con MQPMO_PASS_ALL_CONTEXT especificado, hace que la propiedad se copie del contexto guardado en el nuevo mensaje.

Si la opción descrita anteriormente no es necesaria, se puede utilizar la opción siguiente:

MQPD_NO_CONTEXT
La propiedad no está asociada a un contexto de mensaje.

Un valor no reconocido se rechaza con un código *Reason*de MQRC_PD_ERROR

Es un campo de entrada/salida para la llamada MQSETMP y un campo de salida de la llamada MQINQMP. El valor inicial de este campo es MQPD_NO_CONTEXT.

CopyOptions (MQLONG)

Describe en qué tipo de mensajes se debe copiar la propiedad. Es un campo de sólo salida para las propiedades definidas de WebSphere MQ reconocidas; WebSphere MQ establece el valor adecuado.

Cuando un gestor de colas recibe un mensaje que contiene una propiedad definida de WebSphere MQ que el gestor de colas reconoce como incorrecta, el gestor de colas corrige el valor del campo *CopyOptions* .

Puede especificar una o más de estas opciones y, si necesita más de una, los valores pueden ser:

- Añadido (no añadir la misma constante más de una vez), o
- Combinado utilizando la operación OR bit a bit (si el lenguaje de programación da soporte a operaciones de bits).

REENVÍO de MQCOPY_FORWARD
Esta propiedad se copia en un mensaje que se está reenviando.

MQCOPY_PUBLISH
Esta propiedad se copia en el mensaje recibido por un suscriptor cuando se publica un mensaje.

MQCOPY_REPLY
Esta propiedad se copia en un mensaje de respuesta.

INFORME de MQCOPY_REPORT
Esta propiedad se copia en un mensaje de informe.

MQCOPY_ALL
Esta propiedad se copia en todos los tipos de mensajes posteriores.

Opción predeterminada: Se puede especificar la siguiente opción para proporcionar el conjunto predeterminado de opciones de copia:

MQCOPY_DEFAULT
Esta propiedad se copia en un mensaje que se está reenviando, en un mensaje de informe o en un mensaje recibido por un suscriptor cuando se está publicando un mensaje.

Esto equivale a especificar la combinación de las opciones MQCOPY_FORWARD, más MQCOPY_REPORT, más MQCOPY_PUBLISH.

Si no se requiere ninguna de las opciones descritas anteriormente, utilice la opción siguiente:

MQCOPY_NONE
Utilice este valor para indicar que no se han especificado otras opciones de copia; mediante programación no existe ninguna relación entre esta propiedad y los mensajes posteriores. Esto siempre se devuelve para las propiedades del descriptor de mensaje.

Es un campo de entrada/salida para la llamada MQSETMP y un campo de salida de la llamada MQINQMP. El valor inicial de este campo es MQCOPY_DEFAULT.

Opciones (MQLONG)

El valor debe ser:

MQPD_NONE
No se ha especificado ninguna opción

Siempre es un campo de entrada. El valor inicial de este campo es MQPD_NONE.

StrucId (MQCHAR4)

Este es el identificador de estructura; el valor debe ser:

MQPD_STRUC_ID
Identificador de la estructura del descriptor de propiedad.

Para el lenguaje de programación C, la constante **MQPD_STRUC_ID_ARRAY** también está definida; tiene el mismo valor que **MQPD_STRUC_ID**, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es **MQPD_STRUC_ID**.

Soporte (MQLONG)

Este campo describe qué nivel de soporte para la propiedad de mensaje es necesario para el gestor de colas, para que el mensaje que contiene esta propiedad se coloque en una cola. Esto sólo se aplica a las propiedades definidas por WebSphere MQ; el soporte para todas las demás propiedades es opcional.

El campo se establece automáticamente en el valor correcto cuando el gestor de colas conoce la propiedad definida por WebSphere MQ. Si no se reconoce la propiedad, se asigna MQPD_SUPPORT_OPTIONAL. Cuando un gestor de colas recibe un mensaje que contiene una propiedad definida por WebSphere MQque el gestor de colas reconoce como incorrecta, el gestor de colas corrige el valor del campo *Support* .

Cuando se establece una propiedad definida por WebSphere MQutilizando la llamada MQSETMP en un manejador de mensajes donde se ha establecido la opción MQCMHO_NO_VALIDATION, *Support* se convierte en un campo de entrada. Esto permite a una aplicación colocar una propiedad definida por WebSphere MQ, con el valor correcto, donde la propiedad no está soportada por el gestor de colas conectado, pero donde el mensaje está pensado para procesarse en otro gestor de colas.

El valor MQPD_SUPPORT_OPTIONAL siempre se asigna a propiedades que no son propiedades definidas por WebSphere MQ.

Si un gestor de colas WebSphere MQ Versión 7.0 , que da soporte a propiedades de mensaje, recibe una propiedad que contiene un valor *Support* no reconocido, la propiedad se trata como si:

- Se ha especificado MQPD_SUPPORT_REQUIRED si alguno de los valores no reconocidos está contenido en MQPD_REJECT_UNSUP_MASK.
- Se ha especificado MQPD_SUPPORT_REQUIRED_IF_LOCAL si alguno de los valores no reconocidos están contenidos en MQPD_ACCEPT_UNSUP_IF_XMIT_MASK
- De lo contrario, se ha especificado MQPD_SUPPORT_OPTIONAL.

La llamada MQINQMP devuelve uno de los valores siguientes, o se puede especificar uno de los valores, cuando se utiliza la llamada MQSETMP en un manejador de mensajes donde se establece la opción MQCMHO_NO_VALIDATION:

MQPD_SUPPORT_OPTIONAL
Un gestor de colas acepta la propiedad aunque no esté soportada. La propiedad se puede descartar para que el mensaje fluya a un gestor de colas que no da soporte a las propiedades de mensaje. Este valor también se asigna a propiedades que no están definidas por WebSphere MQ.

MQPD_SUPPORT_REQUIRED
Se necesita soporte para la propiedad. El mensaje es rechazado por un gestor de colas que no da soporte a la propiedad definida por WebSphere MQ. La llamada MQPUT o MQPUT1 falla con el código de terminación MQCC_FAILED y el código de razón MQRC_UNSUPPORTED_PROPERTY.

MQPD_SUPPORT_REQUIRED_IF_LOCAL
El mensaje es rechazado por un gestor de colas que no da soporte a la propiedad definida por WebSphere MQsi el mensaje está destinado a una cola local. La llamada MQPUT o MQPUT1 falla con el código de terminación MQCC_FAILED y el código de razón MQRC_UNSUPPORTED_PROPERTY.

La llamada MQPUT o MQPUT1 se ejecuta correctamente si el mensaje está destinado a un gestor de colas remoto.

Es un campo de salida en la llamada MQINQMP y un campo de entrada en la llamada MQSETMP si el descriptor de mensaje se ha creado con la opción MQCMHO_NO_VALIDATION establecida. El valor inicial de este campo es MQPD_SUPPORT_OPTIONAL.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQPD_VERSION_1
Estructura del descriptor de propiedad Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQPD_CURRENT_VERSION
Versión actual de la estructura del descriptor de propiedad.

Siempre es un campo de entrada. El valor inicial de este campo es **MQPD_VERSION_1**.

Valores iniciales y declaraciones de lenguaje para MQPD

Tabla 524. Valores iniciales de los campos en MQPD

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQPD_STRUC_ID	'PD'

Tabla 524. Valores iniciales de los campos en MQPD (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
Version	MQPD_VERSION_1	1
Options	MQPD_NONE	0
Support	MQPD_SUPPORT_OPTIONAL	0
Context	MQPD_NO_CONTEXT	0
CopyOptions	MQCOPY_DEFAULT	0

Notas:

1. En el lenguaje de programación C, la variable de macro MQPD_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQPD MyPD = {MQPD_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

MQPMO-Opciones de transferencia de mensajes

La tabla siguiente resume los campos de la estructura.

Tabla 525. estructura MQPMO		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Options	Opciones que controlan la acción de MQPUT y MQPUT1	Opciones
Timeout	Reserved	Timeout
Context	Descriptor de contexto de objeto de la cola de entrada	Contexto
KnownDestCount	Número de mensajes enviados satisfactoriamente a las colas locales.	KnownDestCount
UnknownDestCount	Número de mensajes enviados satisfactoriamente a las colas remotas.	UnknownDestCount
InvalidDestCount	Número de mensajes que no se han podido enviar	InvalidDestCount
ResolvedQName	Nombre resuelto de la cola de destino	ResolvedQName
ResolvedQMgrName	Nombre resuelto del gestor de colas de destino	ResolvedQMgrName
Nota: Los campos restantes se ignoran si *Version* es menor que MQPMO_VERSION_2.		
RecsPresent	Número de registros de transferencia de mensaje o de registros de respuesta presente	RecsPresent
PutMsgRecFields	Distintivos que indican qué campos de MQPMR están presentes	PutMsgRecFields
PutMsgRecOffset	Desplazamiento del primer registro de transferencia de mensaje desde el principio de MQPMO	PutMsgRecOffset
ResponseRecOffset	Desplazamiento del primer registro de respuesta desde el principio de MQPMO	ResponseRecOffset
PutMsgRecPtr	Dirección del primer registro de transferencia de mensaje	PutMsgRecPtr
ResponseRecPtr	Dirección del primer registro de respuesta	ResponseRecPtr

Tabla 525. estructura MQPMO (continuación)		
Campo	**Descripción**	**Tema**
Nota: Los campos restantes se ignoran si *Version* es menor que MQPMO_VERSION_3.		
OriginalMsgHandle	Descriptor de contexto de mensaje original	OriginalMsgHandle
NewMsgHandle	Manejador de mensajes nuevo	NewMsgHandle
Action	Tipo de colocación que se está realizando y la relación entre el mensaje original especificado por el campo *OriginalMsgHandle* y el nuevo mensaje especificado por el campo *NewMsgHandle*	Acción
PubLevel	Nivel de suscripción al que se dirige la publicación	PubLevel

Visión general de MQPMO

Disponibilidad: todos los sistemas WebSphere MQ , además de los clientes WebSphere MQ conectados a estos sistemas.

Finalidad: la estructura MQPMO permite a la aplicación especificar opciones que controlan cómo se colocan los mensajes en las colas o se publican en los temas. La estructura es un parámetro de entrada/salida en las llamadas MQPUT y MQPUT1 .

Versión: La versión actual de MQPMO es MQPMO_VERSION_3. Determinados campos sólo están disponibles en determinadas versiones de MQPMO. Si necesita portar aplicaciones entre varios entornos, debe asegurarse de que la versión de MQPMO sea coherente en todos los entornos. Los campos que existen sólo en determinadas versiones de la estructura se identifican como tales en "MQPMO-Opciones de transferencia de mensajes" en la página 475 y en las descripciones de campo.

Los archivos de cabecera, COPY e INCLUDE proporcionados para los lenguajes de programación soportados contienen la versión más reciente de MQPMO soportada por el entorno, pero con el valor inicial del campo *Version* establecido en MQPMO_VERSION_1. Para utilizar campos que no están presentes en la estructura version-1 , la aplicación debe establecer el campo *Version* en el número de versión de la versión necesaria.

Conjunto de caracteres y codificación: los datos de MQPMO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos para MQPMO

La estructura MQPMO contiene los campos siguientes; los campos se describen en **orden alfabético**:

Acción (MQLONG)
Especifica el tipo de colocación que se está realizando y la relación entre el mensaje original especificado por el campo de manejador OriginalMsgy el nuevo mensaje especificado por el campo de manejador NewMsg. El gestor de colas elige las propiedades del mensaje de acuerdo con el valor de la acción especificada.

Puede optar por proporcionar el contenido del descriptor de mensaje utilizando el parámetro MsgDesc en las llamadas MQPUT o MQPUT1 . De forma alternativa, es posible no proporcionar el parámetro MsgDesc o especificar que es sólo de salida incluyendo MQPMO_MD_FOR_OUTPUT_ONLY en el campo Opciones de la estructura MQPMO.

Si no se proporciona el parámetro MsgDesc , o si se especifica que sea sólo de salida, el descriptor de mensaje para el nuevo mensaje se llena a partir de los campos de manejador de mensajes de MQPMO, de acuerdo con las reglas descritas en este tema.

El valor de contexto y las actividades de paso descritas en Control de la información de contexto entran en vigor después de que se haya compuesto el descriptor de mensaje.

Si se especifica un valor de acción incorrecto, la llamada falla con el código de razón MQRC_ACTION_ERROR.

Se puede especificar cualquiera de las acciones siguientes:

MQACTP_NEW
Se está colocando un nuevo mensaje y el programa no está especificando ninguna relación con un mensaje anterior. El descriptor de mensaje se compone de lo siguiente:

- Si se proporciona un MsgDesc en la llamada MQPUT o MQPUT1 , y MQPMO_MD_FOR_OUTPUT_ONLY no está en el MQPMO de MQPMO.Options, se utiliza como descriptor de mensaje no modificado.
- Si no se proporciona un MsgDesc , o MQPMO_MD_FOR_OUTPUT_ONLY está en el MQPMO de MQPMO.Options el gestor de colas genera el descriptor de mensaje utilizando una combinación de propiedades de OriginalMsgHandle y NewMsgHandle. Los campos de descriptor de mensaje establecidos explícitamente en el nuevo descriptor de mensaje tienen prioridad sobre los del descriptor de mensaje original.

Los datos de mensaje se toman del parámetro de almacenamiento intermedio MQPUT o MQPUT1 .

MQACTP_FORWARD
Se está reenviando un mensaje recuperado anteriormente. El descriptor de mensaje original especifica el mensaje que se ha recuperado anteriormente.

El nuevo descriptor de mensaje especifica las modificaciones en las propiedades (incluidas las del descriptor de mensaje) del descriptor de mensaje original.

El descriptor de mensaje se compone de lo siguiente:

- Si se proporciona un MsgDesc en la llamada MQPUT o MQPUT1 , y MQPMO_MD_FOR_OUTPUT_ONLY no está en el MQPMO de MQPMO.Options, se utiliza como descriptor de mensaje no modificado.
- Si no se proporciona un MsgDesc , o MQPMO_MD_FOR_OUTPUT_ONLY está en el MQPMO de MQPMO.Options el gestor de colas genera el descriptor de mensaje utilizando una combinación de propiedades de OriginalMsgHandle y NewMsgHandle. Los campos de descriptor de mensaje establecidos explícitamente en el nuevo descriptor de mensaje tienen prioridad sobre los del descriptor de mensaje original.
- Si se especifica MQPMO_NEW_MSG_ID o MQPMO_NEW_CORREL_ID en el MQPMO de MQPMO.Options, a continuación, se respetan.

Las propiedades del mensaje se componen de la forma siguiente:

- Todas las propiedades del descriptor de mensaje original que tienen MQCOPY_FORWARD en la MQPD de MQPD.CopyOptions
- Todas las propiedades del nuevo manejador de mensajes. Para cada propiedad del nuevo descriptor de contexto de mensaje que tenga el mismo nombre que una propiedad del descriptor de contexto de mensaje original, el valor se toma del nuevo descriptor de contexto de mensaje. La única excepción a esta regla es el caso especial cuando la propiedad del nuevo descriptor de contexto de mensaje tiene el mismo nombre que una propiedad del descriptor de contexto de mensaje original, pero el valor de la propiedad es nulo. En este caso, la propiedad se elimina del mensaje.

Los datos de mensaje que se van a reenviar se toman del parámetro de almacenamiento intermedio MQPUT o MQPUT1 .

MQACTP_REPLY
Se está realizando una respuesta a un mensaje recuperado anteriormente. El descriptor de mensaje original especifica el mensaje que se ha recuperado anteriormente.

El nuevo descriptor de mensaje especifica las modificaciones en las propiedades (incluidas las del descriptor de mensaje) del descriptor de mensaje original.

El descriptor de mensaje se compone de lo siguiente:

- Si se proporciona un MsgDesc en la llamada MQPUT o MQPUT1 , y MQPMO_MD_FOR_OUTPUT_ONLY no está en el MQPMO de MQPMO.Options, se utiliza como descriptor de mensaje no modificado.
- Si no se proporciona un MsgDesc , o MQPMO_MD_FOR_OUTPUT_ONLY está en el MQPMO de MQPMO.Options y, a continuación, los campos del descriptor de mensaje inicial se seleccionan de la forma siguiente:

Tabla 526. Transformación de manejador de mensajes de respuesta

Campo de MQMD	**Valor utilizado**
Informe	Si MQRO_PASS_DESCARD_AND_EXPIRE y MQRO_DISCARD_MSG están establecidos: MQRO_DISCARD_MSG de lo contrario MQRO_NONE
MsgType	MQMT_REPLY
Caducidad	Si MQRO_PASS_DESCARD_AND_EXPIRE está establecido: Copiado del mensaje de entrada de lo contrario MQEI_UNLIMITED
Comentarios	MQFB_NONE
MsgId	Si se establece MQPMO_NEW_MSG_ID: Se genera un nuevo identificador de mensaje de lo contrario, si se establece MQRO_PASS_MSG_ID: Copiado del mensaje de entrada de lo contrario MQMI_NONE
CorrelId	Si se establece MQPMO_NEW_CORREL_ID: Se genera un nuevo identificador de correlación else si se establece MQRO_COPY_MSG_ID_TO_CORREL_ID: Copiado del campo MsgId del mensaje de entrada else si se establece MQRO_PASS_CORREL_ID: Copiado del campo CorrelId del mensaje de entrada de lo contrario MQCI_NONE
BackoutCount	0
ReplyToQ	Espacios en blanco
GestorColasRespuesta	Espacios en blanco
GroupId	MQGI_NONE
MsgSeqNumber	1
Desplazamiento	0
MsgFlags	MQMF_NONE
OriginalLength	MQOL_UNDEFINED

- A continuación, el descriptor de mensaje se modifica mediante el nuevo descriptor de mensaje; los campos de descriptor de mensaje establecidos explícitamente como propiedades en el nuevo descriptor de mensaje tienen prioridad sobre los campos de descriptor de mensaje tal como se ha descrito anteriormente.

Las propiedades del mensaje se componen de la forma siguiente:

- Todas las propiedades del descriptor de mensaje original que tienen MQCOPY_REPLY en la MQPD de MQPD.CopyOptions
- Todas las propiedades del nuevo manejador de mensajes. Para cada propiedad del nuevo descriptor de contexto de mensaje que tenga el mismo nombre que una propiedad del descriptor de contexto de mensaje original, el valor se toma del nuevo descriptor de contexto de mensaje. La única excepción a esta regla es el caso especial cuando la propiedad del nuevo descriptor de contexto de mensaje tiene el mismo nombre que una propiedad del descriptor de contexto de mensaje original, pero el valor de la propiedad es nulo. En este caso, la propiedad se elimina del mensaje.

Los datos de mensaje que se van a reenviar se toman del parámetro de almacenamiento intermedio MQPUT/MQPUT1 .

MQACTP_REPORT
Se está generando un informe como resultado de un mensaje recuperado anteriormente. El descriptor de contexto de mensaje original especifica el mensaje que hace que se genere el informe.

El nuevo descriptor de mensaje especifica las modificaciones en las propiedades (incluidas las del descriptor de mensaje) del descriptor de mensaje original.

El descriptor de mensaje se compone de lo siguiente:

- Si se proporciona un MsgDesc en la llamada MQPUT o MQPUT1 , y MQPMO_MD_FOR_OUTPUT_ONLY no está en el MQPMO de MQPMO.Options, se utiliza como descriptor de mensaje no modificado.
- Si no se proporciona un MsgDesc , o MQPMO_MD_FOR_OUTPUT_ONLY está en el MQPMO de MQPMO.Options y, a continuación, los campos del descriptor de mensaje inicial se seleccionan de la forma siguiente:

Tabla 527. Transformación de manejador de mensajes de informe

Campo de MQMD	**Valor utilizado**
Informe	Si MQRO_PASS_DESCARD_AND_EXPIRE y MQRO_DISCARD_MSG están establecidos: MQRO_DISCARD_MSG de lo contrario MQRO_NONE
MsgType	MQMT_REPORT
Caducidad	Si MQRO_PASS_DESCARD_AND_EXPIRE está establecido: Copiado del mensaje de entrada de lo contrario MQEI_UNLIMITED
MsgId	Si se establece MQPMO_NEW_MSG_ID: Se genera un nuevo identificador de mensaje de lo contrario, si se establece MQRO_PASS_MSG_ID: Copiado del mensaje de entrada de lo contrario MQMI_NONE

Tabla 527. Transformación de manejador de mensajes de informe (continuación)	
Campo de MQMD	**Valor utilizado**
CorrelId	Si se establece MQPMO_NEW_CORREL_ID: Se genera un nuevo identificador de correlación else si se establece MQRO_COPY_MSG_ID_TO_CORREL_ID: Copiado del campo MsgId del mensaje de entrada else si se establece MQRO_PASS_CORREL_ID: Copiado del campo CorrelId del mensaje de entrada de lo contrario MQCI_NONE
BackoutCount	0
ReplyToQ	Espacios en blanco
GestorColasRespuesta	Espacios en blanco
OriginalLength	Establézcalo en *BufferLength*

- A continuación, el descriptor de mensaje se modifica mediante el nuevo descriptor de mensaje; los campos de descriptor de mensaje establecidos explícitamente como propiedades en el nuevo descriptor de mensaje tienen prioridad sobre los campos de descriptor de mensaje tal como se ha descrito anteriormente.

Las propiedades del mensaje se componen de la forma siguiente:

- Todas las propiedades del descriptor de mensaje original que tienen MQCOPY_REPORT en la MQPD de MQPD.CopyOptions
- Todas las propiedades del nuevo manejador de mensajes. Para cada propiedad del nuevo descriptor de contexto de mensaje que tenga el mismo nombre que una propiedad del descriptor de contexto de mensaje original, el valor se toma del nuevo descriptor de contexto de mensaje. La única excepción a esta regla es el caso especial cuando la propiedad del nuevo descriptor de contexto de mensaje tiene el mismo nombre que una propiedad del descriptor de contexto de mensaje original, pero el valor de la propiedad es nulo. En este caso, la propiedad se elimina del mensaje.

El campo Comentarios en el MQMD resultante representa el informe que se va a generar. Un valor de retorno de MQFB_NONE hace que la llamada MQPUT o MQPUT1 falle con el código de razón MQRC_FEEDBACK_ERROR.

Para elegir los datos de usuario del mensaje de informe, WebSphere MQ consulta los campos Informe y Comentarios en el MQMD resultante, y los parámetros Buffer y BufferLength de la llamada MQPUT o MQPUT1 .

- Si los comentarios son MQFB_COA, MQFB_COD o MQFB_EXPIRATION, se inspecciona el valor del informe.
- Si se cumple alguno de los casos siguientes, se utilizan los datos de mensaje completos del almacenamiento intermedio para una longitud de BufferLength .
  - Los comentarios son MQFB_EXPIRATION y el informe contiene MQRO_EXPIRATION_WITH_FULL_DATA
  - Los comentarios son MQFB_COD y el informe contiene MQRO_COD_WITH_FULL_DATA
  - Los comentarios son MQFB_COA y el informe contiene MQRO_COA_WITH_FULL_DATA
- Si se cumple alguno de los casos siguientes, se utilizan los primeros 100 bytes del mensaje (o BufferLength si es inferior a 100) del almacenamiento intermedio
  - Los comentarios son MQFB_EXPIRATION y el informe contiene MQRO_EXPIRATION_WITH_DATA

  - Los comentarios son MQFB_COD y el informe contiene MQRO_COD_WITH_DATA
  - Los comentarios son MQFB_COA y el informe contiene MQRO_COA_WITH_DATA
- Si los comentarios son MQFB_EXPIRATION, MQFB_COD o MQFB_COA, y el informe no contiene las opciones * _WITH_FULL_DATA o * _WITH_DATA relevantes para ese valor de Feeback, no se incluirán datos de usuario con el mensaje.
- Si Feedback toma un valor diferente de los listados anteriormente, Buffer y BufferLength se utilizan de forma normal.

La derivación de los datos de usuario se muestra en la tabla siguiente:

Tabla 528. Origen de datos de usuario

	MQFB_COA	**MQFB_COD**	**MQFB_EXPIRATION**
MQRO_EXPIRATION_WITH_FULL_DATA	Ninguna	Ninguna	Almacenamiento intermedio (Bufferlength)
MQRO_COD_WITH_FULL_DATA	Ninguna	Almacenamiento intermedio (Bufferlength)	Ninguna
MQRO_COA_WITH_FULL_DATA	Almacenamiento intermedio (Bufferlength)	Ninguna	Ninguna
MQRO_EXPIRATION_WITH_DATA	Ninguna	Ninguna	Almacenamiento intermedio (primeros 100 bytes)
MQRO_COD_WITH_DATA	Ninguna	Almacenamiento intermedio (primeros 100 bytes)	Ninguna
MQRO_COA_WITH_DATA	Almacenamiento intermedio (primeros 100 bytes)	Ninguna	Ninguna

Contexto (MQHOBJ)

Si se especifica MQPMO_PASS_IDENTITY_CONTEXT o MQPMO_PASS_ALL_CONTEXT, este campo debe contener el descriptor de contexto de la cola de entrada del que se toma la información de contexto que se va a asociar con el mensaje que se va a colocar.

Si no se especifica MQPMO_PASS_IDENTITY_CONTEXT ni MQPMO_PASS_ALL_CONTEXT, este campo se ignora.

Este es un campo de entrada. El valor inicial de este campo es `0`.

Recuento de InvalidDest(MQLONG)

Es el número de mensajes que no se han podido enviar a las colas de la lista de distribución. El recuento incluye las colas que no se han podido abrir, así como las colas que se han abierto correctamente pero para las que la operación de colocación ha fallado. Este campo también se establece cuando se coloca un mensaje en una sola cola que no está en una lista de distribución.

Nota: Este campo se establece si el parámetro *CompCode* de la llamada MQPUT o MQPUT1 es MQCC_OK o MQCC_WARNING; se puede establecer si el parámetro *CompCode* es MQCC_FAILED, pero no se basa en esto en el código de aplicación.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *Version* es menor que MQPMO_VERSION_1.

Este campo no está definido en z/OS porque las listas de distribución no están soportadas.

Recuento de KnownDest(MQLONG)

Es el número de mensajes que la llamada MQPUT o MQPUT1 actual ha enviado correctamente a las colas de la lista de distribución que son colas locales. El recuento no incluye los mensajes enviados a colas que se resuelven en colas remotas (aunque inicialmente se utilice una cola de transmisión local para almacenar el mensaje). Este campo también se establece cuando se coloca un mensaje en una sola cola que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *Version* es menor que MQPMO_VERSION_1.

Este campo no está definido en z/OS porque las listas de distribución no están soportadas.

Manejador NewMsg(MQHMSG)

Se trata de un descriptor de contexto opcional para el mensaje que se coloca sujeto al valor del campo Acción. Define las propiedades del mensaje y altera temporalmente los valores de *OriginalMsgHandle*, si se especifica.

Al volver de la llamada **MQPUT** o **MQPUT1** , el contenido del descriptor de contexto refleja el mensaje que se ha colocado realmente.

Este es un campo de entrada. El valor inicial de este campo es **MQHM_NONE**. Este campo se ignora si la versión es menor que **MQPMO_VERSION_3**.

Opciones de MQPMO (MQLONG)
El campo Opciones controla el funcionamiento de las llamadas **MQPUT** y **MQPUT1** .

Opción de ámbito. Puede especificar alguna o ninguna de las opciones MQPMO. Si se necesita más de una opción, los valores que especifique para las opciones se pueden utilizar de las maneras siguientes:

- Se pueden añadir los valores. No añada la misma constante más de una vez.
- Los valores se pueden combinar utilizando la operación OR a nivel de bit, si el lenguaje de programación da soporte a operaciones a nivel de bit.

Las combinaciones que no son válidas se anotan; cualquier otra combinación es válida.

La siguiente opción controla el ámbito de las publicaciones enviadas:

MQPMO_SCOPE_QMGR

La publicación solo se envía a los suscriptores que se han suscrito a este gestor de colas. La publicación no se reenvía a ningún gestor de colas de publicación/suscripción remoto que haya realizado una suscripción a este gestor de colas, lo que altera temporalmente cualquier comportamiento que se haya establecido utilizando el atributo de tema PUBSCOPE.

Nota: Si no se establece, el ámbito de publicación viene determinado por el atributo de tema PUBSCOPE.

Opciones de publicación. Las opciones siguientes controlan la forma en que se publican los mensajes en un tema:

MQPMO_SUPPRESS_REPLYTO

Cualquier información especificada en los campos *ReplyToQ* y *ReplyToQMgr* del MQMD de esta publicación no se pasa a los suscriptores. Si esta opción se utiliza con una opción de informe que requiere un *ReplyToQ*, la llamada falla con MQRC_MISSING_REPLY_TO_Q.

MQPMO_RETAIN

El gestor de colas debe retener la publicación que se está enviando. Esta retención permite a un suscriptor solicitar una copia de esta publicación después de la hora en que se publicó, utilizando la llamada MQSUBRQ. También permite que se envíe una publicación a las aplicaciones que realizan su suscripción después del momento en que se realizó esta publicación (a menos que elijan no enviarla utilizando la opción MQSO_NEW_PUBLICATIONS_ONLY). Si a una aplicación se le envía una

publicación que se ha retenido, se indica mediante la propiedad de mensaje MQIsRetained de dicha publicación.

Sólo se puede retener una publicación en cada nodo del árbol de temas. Por lo tanto, si ya hay una publicación retenida para este tema, publicada por cualquier otra aplicación, se sustituye por esta publicación. Por lo tanto, es mejor evitar que más de un editor retenga mensajes sobre el mismo tema.

Cuando un suscriptor solicita las publicaciones retenidas, la suscripción utilizada puede contener un comodín en el tema, en cuyo caso pueden coincidir varias publicaciones retenidas (en varios nodos del árbol de temas) y se pueden enviar varias publicaciones a la aplicación solicitante. Consulte la descripción de la llamada "MQSUBRQ-Solicitud de suscripción" en la página 777 para obtener más detalles.

Para obtener información sobre cómo interactúan las publicaciones retenidas con los niveles de suscripción, consulte Interceptación de publicaciones.

Si se utiliza esta opción y la publicación no se puede retener, el mensaje no se publica y la llamada falla con MQRC_PUT_NOT_RETAINED.

MQPMO_NOT_OWN_SUBS
Indica al gestor de colas que la aplicación no desea enviar ninguna de sus publicaciones a las suscripciones que posee. Las suscripciones se consideran propiedad de la misma aplicación si los manejadores de conexión son los mismos.

MQPMO_WARN_IF_NO_SUBS_MATCHED
Si ninguna suscripción coincide con la publicación, devuelva un código de terminación (*CompCode*) de MQCC_WARNING y el código de razón MQRC_NO_SUBS_MATCHES.

Si la operación de colocación devuelve MQRC_NO_SUBS_MISMA, la publicación no se ha entregado a ninguna suscripción. Sin embargo, si se especifica la opción MQPMO_RETAIN en la operación de colocación, el mensaje se conserva y se entrega a cualquier suscripción coincidente definida posteriormente.

Una suscripción sobre el tema coincide con la publicación si se cumple alguna de las condiciones siguientes:

- El mensaje se entrega a la cola de suscripción
- El mensaje se habría entregado a la cola de suscripción, pero un problema con la cola significa que el mensaje no se puede transferir a la cola y, en consecuencia, se ha colocado en la cola de mensajes no entregados o se ha descartado.
- Se define una salida de direccionamiento que suprime la entrega del mensaje a la suscripción

Una suscripción sobre el tema no coincide con la publicación si se cumple alguna de las condiciones siguientes:

- La suscripción tiene una serie de selección que no coincide con la publicación
- La suscripción ha especificado la opción MQSO_PUBLICATION_ON_REQUEST
- La publicación no se entrega porque se ha especificado la opción MQPMO_NOT_OWN_SUBS en la operación de colocación y la suscripción coincide con la identidad del publicador

Opciones de punto de sincronismo. Las opciones siguientes están relacionadas con la participación de la llamada MQPUT o MQPUT1 dentro de una unidad de trabajo:

MQPMO_SYNCPOINT
La solicitud es operar dentro de los protocolos normales de unidad de trabajo. El mensaje no está visible fuera de la unidad de trabajo hasta que se confirme la unidad de trabajo. Si la unidad de trabajo se restituye, se suprime el mensaje.

Si no se especifican MQPMO_SYNCPOINT y MQPMO_NO_SYNCPOINT, la inclusión de la solicitud de colocación en los protocolos de unidad de trabajo viene determinada por el entorno que ejecuta el gestor de colas y no por el entorno que ejecuta la aplicación. En z/OS, la solicitud de colocación está

dentro de una unidad de trabajo. En todos los demás entornos, la solicitud de colocación no está dentro de una unidad de trabajo.

Debido a estas diferencias, una aplicación que desea portar no debe permitir que esta opción sea la predeterminada; especifique MQPMO_SYNCPOINT o MQPMO_NO_SYNCPOINT explícitamente.

No especifique MQPMO_SYNCPOINT con MQPMO_NO_SYNCPOINT.

MQPMO_NO_SYNCPOINT
La solicitud es operar fuera de los protocolos normales de unidad de trabajo. El mensaje está disponible inmediatamente y no se puede suprimir restituir una unidad de trabajo.

Si no se especifican MQPMO_NO_SYNCPOINT y MQPMO_SYNCPOINT, la inclusión de la solicitud de colocación en protocolos de unidad de trabajo viene determinada por el entorno que ejecuta el gestor de colas y no por el entorno que ejecuta la aplicación. En z/OS, la solicitud de colocación está dentro de una unidad de trabajo. En todos los demás entornos, la solicitud de colocación no está dentro de una unidad de trabajo.

Debido a estas diferencias, una aplicación que desea portar no debe permitir que esta opción sea la predeterminada; especifique MQPMO_SYNCPOINT o MQPMO_NO_SYNCPOINT explícitamente.

No especifique MQPMO_NO_SYNCPOINT con MQPMO_SYNCPOINT.

Opciones de identificador de mensaje e identificador de correlación. Las opciones siguientes solicitan al gestor de colas que genere un nuevo identificador de mensaje o identificador de correlación:

MQPMO_NEW_MSG_ID
El gestor de colas sustituye el contenido del campo *MsgId* en MQMD por un nuevo identificador de mensaje. Este identificador de mensaje se envía con el mensaje y se devuelve a la aplicación en la salida de la llamada MQPUT o MQPUT1 .

La opción MQPMO_NEW_MSG_ID también se puede especificar cuando el mensaje se transfiere a una lista de distribución; consulte la descripción del campo *MsgId* en la estructura MQPMR para obtener detalles.

El uso de esta opción libera a la aplicación de la necesidad de restablecer el campo *MsgId* a MQMI_NONE antes de cada llamada MQPUT o MQPUT1 .

MQPMO_NEW_CORREL_ID
El gestor de colas sustituye el contenido del campo *CorrelId* en MQMD por un nuevo identificador de correlación. Este identificador de correlación se envía con el mensaje y se devuelve a la aplicación en la salida de la llamada MQPUT o MQPUT1 .

La opción MQPMO_NEW_CORREL_ID también se puede especificar cuando el mensaje se coloca en una lista de distribución; consulte la descripción del campo *CorrelId* en la estructura MQPMR para obtener detalles.

MQPMO_NEW_CORREL_ID es útil en situaciones en las que la aplicación requiere un identificador de correlación exclusivo.

Opciones de grupo y segmento. Las opciones siguientes están relacionadas con el proceso de mensajes en grupos y segmentos de mensajes lógicos. Lea las definiciones siguientes para ayudarle a comprender la opción.

Atención: No puede utilizar los mensajes segmentados ni agrupados con la publicación/ suscripción.

Mensaje físico
Es la unidad de información más pequeña que se puede colocar o eliminar de una cola; a menudo corresponde a la información especificada o recuperada en una sola llamada MQPUT, MQPUT1o MQGET. Cada mensaje físico tiene su propio descriptor de mensaje (MQMD). Generalmente, los mensajes físicos se distinguen por valores diferentes para el identificador de mensaje (campo*MsgId* en MQMD), aunque el gestor de colas no los impone.

Mensaje lógico
Un mensaje lógico es una única unidad de información de aplicación sólo para plataformas que no sonz/OS . En ausencia de restricciones del sistema, un mensaje lógico es el mismo que un mensaje físico. Pero cuando los mensajes lógicos son extremadamente grandes, las restricciones del sistema pueden hacer aconsejable o necesario dividir un mensaje lógico en dos o más mensajes físicos, denominados *segmentos*.

Un mensaje lógico que se ha segmentado consta de dos o más mensajes físicos que tienen el mismo identificador de grupo no nulo (campo*GroupId* en MQMD) y el mismo número de secuencia de mensaje (campo*MsgSeqNumber* en MQMD). Los segmentos se distinguen por valores diferentes para el desplazamiento de segmento (campo*Offset* en MQMD), que proporciona el desplazamiento de los datos en el mensaje físico desde el inicio de los datos en el mensaje lógico. Debido a que cada segmento es un mensaje físico, los segmentos de un mensaje lógico normalmente tienen identificadores de mensaje diferentes.

Un mensaje lógico que no se ha segmentado, pero para el que la aplicación emisora ha permitido la segmentación, también tiene un identificador de grupo no nulo, aunque en este caso sólo hay un mensaje físico con ese identificador de grupo si el mensaje lógico no pertenece a un grupo de mensajes. Los mensajes lógicos para los que la aplicación emisora ha inhibido la segmentación tienen un identificador de grupo nulo (MQGI_NONE), a menos que el mensaje lógico pertenezca a un grupo de mensajes.

Grupo de mensajes
Un grupo de mensajes es un conjunto de uno o más mensajes lógicos que tienen el mismo identificador de grupo no nulo. Los mensajes lógicos del grupo se distinguen por valores diferentes para el número de secuencia de mensaje, que es un entero en el rango de 1 a *n*, donde *n* es el número de mensajes lógicos del grupo. Si uno o varios de los mensajes lógicos están segmentados, hay más de *n* mensajes físicos en el grupo.

MQPMO_LOGICAL_ORDER
Esta opción indica al gestor de colas cómo la aplicación coloca los mensajes en grupos y segmentos de mensajes lógicos. Sólo se puede especificar en la llamada MQPUT; no es válida en la llamada MQPUT1.

Si se especifica MQPMO_LOGICAL_ORDER, indica que la aplicación utiliza llamadas MQPUT sucesivas para:

1. Poner los segmentos de cada mensaje lógico en el orden de desplazamiento de segmento creciente, empezando desde 0, sin huecos.
2. Poner todos los segmentos en un mensaje lógico antes de poner los segmentos en el siguiente mensaje lógico.
3. Poner los mensajes lógicos de cada grupo de mensajes en el orden de número de secuencia de mensaje creciente, empezando desde 1, sin huecos.IBM WebSphere MQ incrementa automáticamente el número de secuencia de mensajes.
4. Poner todos los mensajes lógicos en un grupo de mensajes antes de poner los mensajes lógicos en el siguiente grupo de mensajes.

Para obtener información detallada sobre MQPMO_LOGICAL_ORDER, consulte Ordenación lógica y física

Opciones de contexto. Las opciones siguientes controlan el proceso del contexto de mensaje:

MQPMO_NO_CONTEXT
Tanto el contexto de identidad como el contexto de origen se establecen para indicar que no hay contexto. Esto significa que los campos de contexto en MQMD se establecen en:

- Espacios en blanco para campos de caracteres
- Nulos para campos de bytes
- Ceros para campos numéricos

MQPMO_DEFAULT_CONTEXT
El mensaje debe tener asociada la información de contexto predeterminada, tanto para la identidad como para el origen. El gestor de colas establece los campos de contexto en el descriptor de mensaje como se indica a continuación:

Campo de MQMD	Valor utilizado
UserIdentifier	Se determina a partir del entorno si es posible; de lo contrario, se establece en blancos.
AccountingToken	Determinado por el entorno si es posible; en caso contrario, establecido en MQACT.
ApplIdentityData	Establecido en blancos.
PutApplType	Determinado a partir del entorno.
PutApplName	Se determina a partir del entorno si es posible; de lo contrario, se establece en blancos.
PutDate	Establézcalo en la fecha en la que se coloca el mensaje.
PutTime	Establézcalo en la hora en que se coloca el mensaje.
ApplOriginData	Establecido en blancos.

Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Estos son los valores y acciones predeterminados si no se especifica ninguna opción de contexto.

MQPMO_PASS_IDENTITY_CONTEXT
El mensaje tiene que tener información de contexto asociada. El contexto de identidad se toma del descriptor de contexto de cola especificado en el campo *Context* . El gestor de colas genera la información de contexto de origen de la misma forma que para MQPMO_DEFAULT_CONTEXT (consulte la tabla anterior para ver los valores). Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Para la llamada MQPUT, la cola debe haberse abierto con la opción MQOO_PASS_IDENTITY_CONTEXT (o una opción que lo implique). Para la llamada MQPUT1 , se realiza la misma comprobación de autorización que para la llamada MQOPEN con la opción MQOO_PASS_IDENTITY_CONTEXT.

MQPMO_PASS_ALL_CONTEXT
El mensaje tiene que tener información de contexto asociada. El contexto se toma del descriptor de contexto de cola especificado en el campo *Context* . Para obtener más información sobre el contexto de mensaje, consulte Control de la información de contexto.

Para la llamada MQPUT, la cola debe haberse abierto con la opción MQOO_PASS_ALL_CONTEXT (o una opción que lo implique). Para la llamada MQPUT1 , se realiza la misma comprobación de autorización que para la llamada MQOPEN con la opción MQOO_PASS_ALL_CONTEXT.

MQPMO_SET_IDENTITY_CONTEXT
El mensaje tiene que tener información de contexto asociada. La aplicación especifica el contexto de identidad en la estructura MQMD. El gestor de colas genera la información de contexto de origen de la misma forma que para MQPMO_DEFAULT_CONTEXT (consulte la tabla anterior para ver los valores). Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Para la llamada MQPUT, la cola debe haberse abierto con la opción MQOO_SET_IDENTITY_CONTEXT (o una opción que lo implique). Para la llamada MQPUT1 , se realiza la misma comprobación de autorización que para la llamada MQOPEN con la opción MQOO_SET_IDENTITY_CONTEXT.

MQPMO_SET_ALL_CONTEXT
El mensaje tiene que tener información de contexto asociada. La aplicación especifica la identidad, el origen y el contexto de usuario en la estructura MQMD. Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.

Para la llamada MQPUT, la cola debe haberse abierto con la opción MQOO_SET_ALL_CONTEXT. Para la llamada MQPUT1 , se realiza la misma comprobación de autorización que para la llamada MQOPEN con la opción MQOO_SET_ALL_CONTEXT.

Sólo puede especificar una de las opciones de contexto MQPMO_ * _CONTEXT. Si no especifica ninguno, se presupone MQPMO_DEFAULT_CONTEXT.

Opciones de propiedad. La siguiente opción está relacionada con las propiedades del mensaje:

MQPMO_MD_FOR_OUTPUT_ONLY
El parámetro de descriptor de mensaje sólo debe utilizarse para que la salida devuelva el descriptor de mensaje del mensaje que se ha colocado. Los campos de descriptor de mensaje asociados con los campos *NewMsgHandle*, *OriginalMsgHandle*, o ambos, de la estructura **MQPMO** se deben utilizar para la entrada.

Si no se proporciona un manejador de mensajes válido, la llamada falla con el código de razón **MQRC_MD_ERROR**.

Opciones de respuesta de colocación. Las opciones siguientes controlan la respuesta devuelta a una llamada MQPUT o MQPUT1 . Sólo puede especificar una de estas opciones. Si no se especifican MQPMO_ASYNC_RESPONSE y MQPMO_SYNC_RESPONSE, se presupone MQPMO_RESPONSE_AS_Q_DEF o MQPMO_RESPONSE_AS_TOPIC_DEF.

MQPMO_ASYNC_RESPONSE
La opción MQPMO_ASYNC_RESPONSE solicita que se complete una operación MQPUT o MQPUT1 sin que la aplicación espere a que el gestor de colas complete la llamada. La utilización de esta opción puede mejorar el rendimiento de la mensajería, especialmente para las aplicaciones que utilizan enlaces de cliente. Una aplicación puede comprobar periódicamente, utilizando el verbo MQSTAT, si se ha producido un error durante cualquier llamada asíncrona anterior.

Con esta opción, solo se garantiza que se completen los campos siguientes en el MQMD;

- ApplIdentityData
- PutApplType
- PutApplName
- ApplOriginData

Además, si se especifica MQPMO_NEW_MSG_ID o MQPMO_NEW_CORREL_ID como opciones, o ambos, el MsgId y el CorrelId devueltos también se completan. (MQPMO_NEW_MSG_ID puede especificarse implícitamente especificando un campo MsgId en blanco).

Sólo se completan los campos especificados anteriores. Otra información que normalmente se devolvería en la estructura MQMD o MQPMO no está definida.

Al solicitar una respuesta de transferencia asíncrona para MQPUT1, el nombre ResolvedQName y ResolvedQMgrdevuelto en la estructura MQOD no están definidos.

Cuando se solicita una respuesta de transferencia asíncrona para MQPUT o MQPUT1, un código CompCode y una razón MQCC_OK y MQRC_NONE no significan necesariamente que el mensaje se haya transferido correctamente a una cola. Al desarrollar una aplicación MQI que utiliza la respuesta de colocación asíncrona y requiere confirmación de que los mensajes se han colocado en una cola, debe comprobar los códigos CompCode y Reason de las operaciones de colocación y también utilizar MQSTAT para consultar información de errores asíncronos.

Aunque el éxito o el error de cada llamada MQPUT o MQPUT1 individual no se devuelven inmediatamente, el primer error que se ha producido bajo una llamada asíncrona se puede determinar más adelante mediante una llamada a MQSTAT.

Si un mensaje persistente bajo punto de sincronismo no se puede entregar utilizando la respuesta de transferencia asíncrona e intenta confirmar la transacción, la confirmación falla y la transacción se restituye con un código de terminación de MQCC_FAILED y una razón de MQRC_BACKED_OUT. La aplicación puede realizar una llamada a MQSTAT para determinar la causa de una anomalía anterior de MQPUT o MQPUT1 .

MQPMO_SYNC_RESPONSE
La especificación de este tipo de respuesta put garantiza que la operación MQPUT o MQPUT1 se emita siempre de forma síncrona. Si la operación de colocación es satisfactoria, se completan todos los campos de MQMD y MQPMO.

Esta opción garantiza una respuesta síncrona independientemente del valor de respuesta de colocación predeterminado definido en la cola o el objeto de tema.

MQPMO_RESPONSE_AS_Q_DEF
Si se especifica este valor para una llamada MQPUT, el tipo de respuesta de colocación utilizado se toma del valor DEFPRESP especificado en la cola cuando la aplicación la abrió por primera vez. Si una aplicación cliente está conectada a un gestor de colas en un nivel anterior a la versión 7.0, se comporta como si se hubiera especificado MQPMO_SYNC_RESPONSE.

Si se especifica esta opción para una llamada MQPUT1 , el valor del atributo DEFPRESP no se conoce antes de que se envíe la solicitud al servidor. De forma predeterminada, si la llamada MQPUT1 utiliza MQPMO_SYNCPOINT, se comporta como para MQPMO_ASYNC_RESPONSE, y si utiliza MQPMO_NO_SYNCPOINT, se comporta como para MQPMO_SYNC_RESPONSE. Sin embargo, puede alterar temporalmente este comportamiento predeterminado estableciendo la propiedad Put1DefaultAlwaysSync en el archivo de configuración del cliente, consulte Stanza CHANNELS del archivo de configuración del cliente.

MQPMO_RESPONSE_AS_TOPIC_DEF
MQPMO_RESPONSE_AS_TOPIC_DEF es un sinónimo de MQPMO_RESPONSE_AS_Q_DEF para su uso con objetos de tema.

Otras opciones. Las opciones siguientes controlan la comprobación de autorización, lo que sucede cuando el gestor de colas se está desactivando temporalmente y la resolución de nombres de colas y gestores de colas:

MQPMO_ALTERNATE_USER_AUTHORITY
MQPMO_ALTERNATE_USER_AUTHORITY indica que el campo *AlternateUserId* del parámetro *ObjDesc* de la llamada MQPUT1 contiene un identificador de usuario que se debe utilizar para validar la autorización para colocar mensajes en la cola. La llamada sólo puede realizarse correctamente si *AlternateUserId* tiene autorización para abrir la cola con las opciones especificadas, independientemente de si el identificador de usuario bajo el que se ejecuta la aplicación tiene autorización para hacerlo. (Sin embargo, esto no se aplica a las opciones de contexto especificadas, que siempre se comprueban con el identificador de usuario bajo el que se ejecuta la aplicación.)

Esta opción sólo es válida con la llamada MQPUT1 .

MQPMO_FAIL_IF_QUIESCING
Esta opción fuerza que la llamada MQPUT o MQPUT1 falle si el gestor de colas está en estado de desactivación temporal.

En z/OS, esta opción también fuerza que la llamada MQPUT o MQPUT1 falle si la conexión (para una aplicación CICS o IMS) está en estado de desactivación temporal.

La llamada devuelve el código de terminación MQCC_FAILED con el código de razón MQRC_Q_MGR_QUIESCING o MQRC_CONNECTION_QUIESCING.

MQPMO_RESOLVE_LOCAL_Q
Utilice esta opción para rellenar *ResolvedQName* en la estructura MQPMO con el nombre de la cola local a la que se coloca el mensaje y *ResolvedQMgrName* con el nombre del gestor de colas local que aloja la cola local. Para obtener más información sobre MQPMO_RESOLVE_LOCAL_Q, consulte el tema MQOO_RESOLVE_LOCAL_Q.

Si tiene autorización para transferir a una cola, tiene la autorización necesaria para especificar este distintivo en la llamada MQPUT; no es necesaria ninguna autorización especial.

Opción predeterminada. Si no necesita ninguna de las opciones descritas, utilice la opción siguiente:

MQPMO_NONE
Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados. MQPMO_NONE está definido para ayudar a la documentación del

programa; no está previsto que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

MQPMO_NONE es un campo de entrada. El valor inicial del campo *Options* es MQPMO_NONE.

Manejador OriginalMsg(MQHMSG)

Se trata de un descriptor de contexto opcional para un mensaje. Es posible que se haya recuperado anteriormente de una cola. El uso de este descriptor de contexto está sujeto al valor del campo *Action* ; consulte también NewMsgHandle.

La llamada **MQPUT** o **MQPUT1** no alterará el contenido del manejador de mensajes original.

Este es un campo de entrada. El valor inicial de este campo es **MQHM_NONE**. Este campo se ignora si la versión es menor que **MQPMO_VERSION_3**.

PubLevel (MQLONG)
El valor inicial de este campo es 9. El nivel de suscripción de destino de esta publicación. Sólo las suscripciones con el SubLevel más alto o inferior a este valor reciben esta publicación. Este valor debe estar en el rango de cero a 9; cero es el nivel más bajo. Sin embargo, si una publicación se ha retenido, ya no está disponible para los suscriptores en niveles superiores porque se vuelve a publicar en PubLevel 1.

Para obtener información, consulte Interceptación de publicaciones .

PutMsgRecFields (MQLONG)

Este campo contiene distintivos que indican qué campos MQPMR están presentes en los registros de mensajes de colocación proporcionados por la aplicación. Utilice *PutMsgRecFields* sólo cuando el mensaje se transfiera a una lista de distribución. El campo se ignora si *RecsPresent* es cero, o si *PutMsgRecOffset* y *PutMsgRecPtr* son cero.

Para los campos que están presentes, el gestor de colas utiliza para cada destino los valores de los campos del registro de mensajes de colocación correspondiente. Para los campos que están ausentes, el gestor de colas utiliza los valores de la estructura MQMD.

Utilice uno o varios de los distintivos siguientes para indicar qué campos están presentes en los registros de mensajes de colocación:

MQPMRF_MSG_ID
El campo de identificador de mensaje está presente.

ID MQPMRF_CORREL_ID
El campo de identificador de correlación está presente.

MQPMRF_ID_grupo
El campo identificador de grupo está presente.

MQPMRF_FEEDBACK
El campo de comentarios está presente.

MQPMRF_ACCOUNTING_TOKEN
El campo de señal de contabilidad está presente.

Si especifica este distintivo, especifique MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT en el campo *Options* ; si no se cumple esta condición, la llamada falla con el código de razón MQRC_PMO_RECORD_FLAGS_ERROR.

Si no hay campos MQPMR presentes, se puede especificar lo siguiente:

MQPMRF_NONE
No hay campos de registro de colocación de mensaje.

Si se especifica este valor, *RecsPresent* debe ser cero, o bien *PutMsgRecOffset* y *PutMsgRecPtr* deben ser cero.

MQPMRF_NONE está definido para ayudar a la documentación del programa. No se pretende que esta constante se utilice con ninguna otra, pero como su valor es cero, tal uso no se puede detectar.

Si *PutMsgRecFields* contiene distintivos que no son válidos, o se proporcionan registros de colocación de mensajes pero *PutMsgRecFields* tiene el valor MQPMRF_NONE, la llamada falla con el código de razón MQRC_PMO_RECORD_FLAGS_ERROR.

Este es un campo de entrada. El valor inicial de este campo es MQPMRF_NONE. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

PutMsgRecOffset (MQLONG)

Es el desplazamiento en bytes del primer registro de mensajes de colocación MQPMR desde el inicio de la estructura MQPMO. El desplazamiento puede ser positivo o negativo. *PutMsgRecOffset* sólo se utiliza cuando el mensaje se coloca en una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Cuando el mensaje se transfiere a una lista de distribución, se puede proporcionar una matriz de uno o más registros de mensajes de colocación MQPMR para especificar determinadas propiedades del mensaje para cada destino individualmente; estas propiedades son:

- Identificador del mensaje
- Identificador de correlación
- Identificador de grupo
- Valor de comentarios
- Señal de contabilidad

No es necesario especificar todas estas propiedades, pero independientemente del subconjunto que elija, especifique los campos en el orden correcto. Consulte la descripción de la estructura MQPMR para obtener más detalles.

Normalmente, debe haber tantos registros de mensajes de colocación como registros de objeto especificados por MQOD cuando se abre la lista de distribución; cada registro de mensajes de colocación proporciona las propiedades de mensaje para la cola identificada por el registro de objeto correspondiente. Las colas de la lista de distribución que no se pueden abrir deben tener asignados registros de mensajes en las posiciones adecuadas de la matriz, aunque las propiedades del mensaje se ignoran en este caso.

El número de registros de mensajes de colocación puede diferir del número de registros de objeto. Si hay menos registros de mensaje de colocación que registros de objeto, las propiedades de mensaje para los destinos que no tienen registros de mensaje de colocación se toman de los campos correspondientes en el MQMD del descriptor de mensaje. Si hay más registros de mensajes de colocación que registros de objeto, el exceso no se utiliza (aunque todavía debe ser posible acceder a ellos). Los registros de mensajes de colocación son opcionales, pero si se proporcionan, debe haber *RecsPresent* de ellos.

Proporcione los registros de mensajes de colocación de forma similar a los registros de objeto en MQOD, ya sea especificando un desplazamiento en *PutMsgRecOffset*, o especificando una dirección en *PutMsgRecPtr*; para obtener detalles sobre cómo hacerlo, consulte el campo *ObjectRecOffset* descrito en “MQOD-Descriptor de objeto” en la página 454.

No se puede utilizar más de uno de *PutMsgRecOffset* y *PutMsgRecPtr* ; la llamada falla con el código de razón MQRC_PUT_MSG_RECORDS_ERROR si ambos son distintos de cero.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

PutMsgRecPtr (MQPTR)

Es la dirección del primer registro de mensaje de colocación MQPMR. Utilice *PutMsgRecPtr* sólo cuando el mensaje se transfiera a una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Puede utilizar *PutMsgRecPtr* o *PutMsgRecOffset* para especificar los registros de mensajes de colocación, pero no ambos; consulte la descripción del campo *PutMsgRecOffset* anterior para obtener más detalles. Si no utiliza *PutMsgRecPtr*, establézcalo en el puntero nulo o en bytes nulos.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada, siendo el valor inicial la serie de bytes totalmente nula.

RecsPresent (MQLONG)

Es el número de registros de mensajes de colocación MQPMR o registros de respuesta MQRR que ha proporcionado la aplicación. Este número puede ser mayor que cero sólo si el mensaje se está colocando en una lista de distribución. Los registros de mensajes de colocación y los registros de respuesta son opcionales; la aplicación no necesita proporcionar ningún registro, o puede optar por proporcionar registros de un solo tipo. Sin embargo, si la aplicación proporciona registros de ambos tipos, debe proporcionar registros *RecsPresent* de cada tipo.

No es necesario que el valor de *RecsPresent* sea el mismo que el número de destinos de la lista de distribución. Si se proporcionan demasiados registros, el exceso no se utiliza; si se proporcionan demasiados pocos registros, se utilizan los valores predeterminados para las propiedades de mensaje para los destinos que no tienen registros de mensajes de colocación (consulte *PutMsgRecOffset*).

Si *RecsPresent* es menor que cero, o es mayor que cero pero el mensaje no se está colocando en una lista de distribución, la llamada falla con el código de razón MQRC_RECS_PRESENT_ERROR.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

ResolvedQMgrNombre (MQCHAR48)

Es el nombre del gestor de colas de destino después de que el gestor de colas local haya realizado la resolución de nombres. El nombre devuelto es el nombre del gestor de colas que es propietario de la cola identificada por *ResolvedQName*, y puede ser el nombre del gestor de colas local.

Si *ResolvedQName* es una cola compartida que es propiedad del grupo de compartición de colas al que pertenece el gestor de colas local, *ResolvedQMgrName* es el nombre del grupo de compartición de colas. Si la cola es propiedad de algún otro grupo de compartición de colas, *ResolvedQName* puede ser el nombre del grupo de compartición de colas o el nombre de un gestor de colas que es miembro del grupo de compartición de colas (la naturaleza del valor devuelto viene determinada por las definiciones de cola que existen en el gestor de colas local).

Sólo se devuelve un valor no en blanco si el objeto es una sola cola; si el objeto es una lista de distribución o un tema, el valor devuelto no está definido.

Se trata de un campo de salida. La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ResolvedQName (MQCHAR48)

Es el nombre de la cola de destino después de que el gestor de colas local haya realizado la resolución de nombres. El nombre devuelto es el nombre de una cola que existe en el gestor de colas identificado por *ResolvedQMgrName*.

Sólo se devuelve un valor no en blanco si el objeto es una sola cola; si el objeto es una lista de distribución o un tema, el valor devuelto no está definido.

Se trata de un campo de salida. La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

Desplazamiento de ResponseRec(MQLONG)

Es el desplazamiento en bytes del primer registro de respuesta MQRR desde el inicio de la estructura MQPMO. El desplazamiento puede ser positivo o negativo. *ResponseRecOffset* sólo se utiliza cuando el mensaje se coloca en una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Al colocar el mensaje en una lista de distribución, puede proporcionar una matriz de uno o varios registros de respuesta MQRR para identificar las colas a las que no se ha enviado correctamente el mensaje (campo*CompCode* en MQRR) y la razón de cada anomalía (campo*Reason* en MQRR). Es posible que el mensaje no se haya enviado porque la cola no se ha podido abrir o porque la operación de colocación ha fallado. El gestor de colas establece los registros de respuesta sólo cuando el resultado de la llamada es mixto (es decir, algunos mensajes se han enviado correctamente mientras que otros han fallado, o todos han fallado pero por razones diferentes); el código de razón MQRC_MULTIPLE_REASON de la llamada indica este caso. Si el mismo código de razón se aplica a todas las colas, esa razón se devuelve en el parámetro *Reason* de la llamada MQPUT o MQPUT1 y los registros de respuesta no se establecen.

Normalmente, hay tantos registros de respuesta como registros de objeto especificados por MQOD cuando se abre la lista de distribución; cuando es necesario, cada registro de respuesta se establece en el código de terminación y el código de razón para la colocación en la cola identificada por el registro de objeto correspondiente. Las colas de la lista de distribución que no se pueden abrir deben seguir teniendo registros de respuesta asignados en las posiciones adecuadas de la matriz, aunque se establecen en el código de terminación y el código de razón resultantes de la operación de apertura, en lugar de en la operación de colocación.

El número de registros de respuesta puede diferir del número de registros de objeto. Si hay menos registros de respuesta que registros de objeto, es posible que la aplicación no pueda identificar todos los destinos para los que ha fallado la operación de colocación, o las razones de las anomalías. Si hay más registros de respuesta que registros de objeto, el exceso no se utiliza (aunque todavía debe ser posible acceder a ellos). Los registros de respuesta son opcionales, pero si se proporcionan, debe haber *RecsPresent* de ellos.

Proporcione los registros de respuesta de forma similar a los registros de objeto en MQOD, ya sea especificando un desplazamiento en *ResponseRecOffset*, o especificando una dirección en *ResponseRecPtr*; para obtener detalles sobre cómo hacerlo, consulte el campo *ObjectRecOffset* descrito en "MQOD-Descriptor de objeto" en la página 454. Sin embargo, no utilice más de uno de *ResponseRecOffset* y *ResponseRecPtr*; la llamada falla con el código de razón MQRC_RESPONSE_RECORDS_ERROR si ambos son distintos de cero.

Para la llamada MQPUT1 , este campo debe ser cero. Esto se debe a que la información de respuesta (si se solicita) se devuelve en los registros de respuesta especificados por el descriptor de objeto MQOD.

Este es un campo de entrada. El valor inicial de este campo es 0. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

ResponseRecPtr (MQPTR)

Es la dirección del primer registro de respuesta MQRR. *ResponseRecPtr* sólo se utiliza cuando el mensaje se coloca en una lista de distribución. El campo se ignora si *RecsPresent* es cero.

Utilice *ResponseRecPtr* o *ResponseRecOffset* para especificar los registros de respuesta, pero no ambos; consulte la descripción del campo *ResponseRecOffset* anterior para obtener más detalles. Si no utiliza *ResponseRecPtr* , establézcalo en el puntero nulo o en bytes nulos.

Para la llamada MQPUT1 , este campo debe ser el puntero nulo o bytes nulos. Esto se debe a que la información de respuesta (si se solicita) se devuelve en los registros de respuesta especificados por el descriptor de objeto MQOD.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula. Este campo se ignora si *Version* es menor que MQPMO_VERSION_2.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada, siendo el valor inicial la serie de bytes totalmente nula.

StrucId (MQCHAR4)

Este es el identificador de estructura; el valor debe ser:

MQPMO_STRUC_ID
Identificador de la estructura de opciones de colocación de mensaje.

Para el lenguaje de programación C, también se define la constante MQPMO_STRUC_ID_ARRAY; tiene el mismo valor que MQPMO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQPMO_STRUC_ID.

Tiempo de espera (MQLONG)

Se trata de un campo reservado; su valor no es significativo. El valor inicial de este campo es -1.

Recuento de UnknownDest(MQLONG)

Es el número de mensajes que la llamada MQPUT o MQPUT1 actual ha enviado correctamente a las colas de la lista de distribución que se resuelven en colas remotas. Los mensajes que el gestor de colas retiene temporalmente en formato de lista de distribución cuentan como el número de destinos individuales que contienen esas listas de distribución. Este campo también se establece cuando se coloca un mensaje en una sola cola que no está en una lista de distribución.

Se trata de un campo de salida. El valor inicial de este campo es 0. Este campo no se establece si *Version* es menor que MQPMO_VERSION_1.

Este campo no está definido en z/OS porque las listas de distribución no están soportadas.

Versión (MQLONG)
Número de versión de la estructura.

El valor debe ser uno de los siguientes:

MQPMO_VERSION_1
Estructura de opciones de colocación de Version-1 .

Esta versión está soportada en todos los entornos.

MQPMO_VERSION_2
Estructura de opciones de colocación de Version-2 .

Esta versión está soportada en los entornos siguientes: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más clientes MQI de WebSphere MQ conectados a estos sistemas.

MQPMO_VERSION_3
Estructura de opciones de colocación de mensaje Version-3 .

Esta versión está soportada en todos los entornos.

Los campos que existen sólo en la versión más reciente de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQPMO_CURRENT_VERSION
Versión actual de la estructura de opciones de colocación de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es MQPMO_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQPMO

Tabla 529. Valores iniciales de los campos en MQPMO

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQPMO_STRUC_ID	'PMO¬'
Version	MQPMO_VERSION_1	1
Options	MQPMO_NONE	0
Timeout	Ninguna	-1

Tabla 529. Valores iniciales de los campos en MQPMO (continuación)

Nombre de campo	Nombre de constante	Valor de constante
Context	Ninguna	0
KnownDestCount	Ninguna	0
UnknownDestCount	Ninguna	0
InvalidDestCount	Ninguna	0
ResolvedQName	Ninguna	Serie nula o espacios en blanco
ResolvedQMgrName	Ninguna	Serie nula o espacios en blanco
RecsPresent	Ninguna	0
PutMsgRecFields	MQPMRF_NONE	0
PutMsgRecOffset	Ninguna	0
ResponseRecOffset	Ninguna	0
PutMsgRecPtr	Ninguna	Puntero nulo o bytes nulos
ResponseRecPtr	Ninguna	Puntero nulo o bytes nulos
OriginalMsgHandle	MQHM_NONE	0
NewMsgHandle	MQHM_NONE	0
Action	MQACTP_NEW	0
PubLevel	Ninguna	9

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQPMO_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQPMO MyPMO = {MQPMO_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

MQPMR-Registro de colocación de mensajes

La tabla siguiente resume los campos de la estructura.

Tabla 530. Campos en MQPMR		
Campo	**Descripción**	**Tema**
MsgId	Identificador del mensaje	MsgId
CorrelId	Identificador de correlación	CorrelId
GroupId	Identificador de grupo	GroupId
Feedback	Código de comentario o razón	FEEDBACK
AccountingToken	Señal de contabilidad	AccountingToken

Visión general de MQPMR

Disponibilidad: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más clientes WebSphere MQ conectados a estos sistemas.

Finalidad: Utilice la estructura MQPMR para especificar varias propiedades de mensaje para un único destino al colocar un mensaje en una lista de distribución. MQPMR es una estructura de entrada/salida para las llamadas MQPUT y MQPUT1 .

Conjunto de caracteres y codificación: los datos de MQPMR deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Uso: al proporcionar una matriz de estas estructuras en la llamada MQPUT o MQPUT1 , puede especificar valores diferentes para cada cola de destino en una lista de distribución. Algunos de los campos son sólo de entrada, otros son de entrada/salida.

Nota: Esta estructura es inusual en que no tiene un diseño fijo. Los campos de esta estructura son opcionales, y la presencia o ausencia de cada campo se indica mediante los distintivos del campo *PutMsgRecFields* en MQPMO. Los campos que están presentes ***deben aparecer en el siguiente orden***:

- *MsgId*
- *CorrelId*
- *GroupId*
- *Feedback*
- *AccountingToken*

Los campos ausentes no ocupan ningún espacio en el registro.

Puesto que MQPMR no tiene un diseño fijo, no se proporciona ninguna definición del mismo en los archivos de cabecera, COPY e INCLUDE para los lenguajes de programación soportados. El programador

de aplicaciones debe crear una declaración que contenga los campos necesarios para la aplicación y establecer los distintivos en *PutMsgRecFields* para indicar los campos que están presentes.

Campos para MQPMR

La estructura MQPMR contiene los campos siguientes; los campos se describen en **orden alfabético**:

AccountingToken (MQBYTE32)

Es la señal de contabilidad que se utilizará para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *AccountingToken* en MQMD para una colocación en una sola cola. Consulte la descripción de *AccountingToken* en "MQMD - Descriptor de mensaje" en la página 392 para obtener información sobre el contenido de este campo.

Si este campo no está presente, se utiliza el valor de MQMD.

Este es un campo de entrada.

CorrelId (MQBYTE24)

Es el identificador de correlación que se debe utilizar para el mensaje enviado a la cola con un nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *CorrelId* en MQMD para una colocación en una sola cola.

Si este campo no está presente en el registro MQPMR, o hay menos registros MQPMR que destinos, el valor de MQMD se utiliza para los destinos que no tienen un registro MQPMR que contenga un campo *CorrelId* .

Si se especifica MQPMO_NEW_CORREL_ID, se genera un *único* identificador de correlación nuevo y se utiliza para todos los destinos de la lista de distribución, independientemente de si tienen registros MQPMR. Esto es diferente de la forma en que se procesa MQPMO_NEW_MSG_ID (consulte el campo *MsgId*).

Es un campo de entrada/salida.

Feedback (MQLONG)

Este es el código de retorno que se debe utilizar para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *Feedback* en MQMD para una colocación en una sola cola.

Si este campo no está presente, se utiliza el valor de MQMD.

Este es un campo de entrada.

GroupId (MQBYTE24)

GroupId es el identificador de grupo que se utilizará para el mensaje enviado a la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *GroupId* en MQMD para una colocación en una sola cola.

Si este campo no está presente en el registro MQPMR, o hay menos registros MQPMR que destinos, el valor de MQMD se utiliza para los destinos que no tienen un registro MQPMR que contenga un campo *GroupId* . El valor se procesa tal como se documenta en Orden físico en una cola, pero con las diferencias siguientes:

- GroupId se crea a partir de QMName y una indicación de fecha y hora. Por lo tanto, para mantener un GroupId exclusivo, mantenga también los nombres de gestor de colas exclusivos. Tampoco vuelva a establecer los relojes en la máquina de gestores de colas.

- En los casos en los que se utilizaría un nuevo identificador de grupo, el gestor de colas genera un identificador de grupo diferente para cada destino (es decir, ningún destino tiene el mismo identificador de grupo).
- En los casos en los que se utilizaría el valor del campo, la llamada falla con el código de razón MQRC_GROUP_ID_ERROR

Es un campo de entrada/salida.

MsgId (MQBYTE24)

Este es el identificador de mensaje que se utilizará para el mensaje enviado a la cola con un nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 . Se procesa de la misma forma que el campo *MsgId* en MQMD para una colocación en una sola cola.

Si este campo no está presente en el registro MQPMR, o hay menos registros MQPMR que destinos, el valor de MQMD se utiliza para los destinos que no tienen un registro MQPMR que contenga un campo *MsgId* . Si ese valor es MQMI_NONE, se genera un nuevo identificador de mensaje para *cada* de esos destinos (es decir, dos de esos destinos no tienen el mismo identificador de mensaje).

Si se especifica MQPMO_NEW_MSG_ID, se generan nuevos identificadores de mensaje para todos los destinos de la lista de distribución, independientemente de si tienen registros MQPMR. Esto es diferente de la forma en que se procesa MQPMO_NEW_CORREL_ID (consulte el campo *CorrelId*).

Es un campo de entrada/salida.

Valores iniciales y declaraciones de lenguaje para MQPMR

No hay valores iniciales definidos para esta estructura, ya que no se proporcionan declaraciones de estructura en los archivos de cabecera, COPY e INCLUDE para los lenguajes de programación soportados. Las declaraciones de ejemplo muestran cómo declarar la estructura si se necesitan todos los campos.

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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).
```

Declaración PL/I

```
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 */
```

Declaración de Visual Basic

```
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
```

MQRFH - Cabecera de reglas y formato

En esta sección se describen las reglas y la cabecera de formato, los campos que contiene y los valores iniciales de dichos campos.

Visión general de MQRFH

Disponibilidad: todos los sistemas WebSphere MQ , además de los clientes MQI de WebSphere MQ conectados a estos sistemas.

Finalidad: la estructura MQRFH define el diseño de las reglas y la cabecera de formato. Utilice esta cabecera para enviar datos de serie en forma de pares de nombre/valor.

Nombre de formato: MQFMT_RF_HEADER.

Conjunto de caracteres y codificación: los campos de la estructura MQRFH (incluido *NameValueString*) están en el juego de caracteres y la codificación proporcionados por los campos *CodedCharSetId* y *Encoding* en la estructura de cabecera que precede a la MQRFH, o por los campos de la estructura MQMD si la MQRFH está al principio de los datos del mensaje de aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de cola.

Campos para MQRFH

La estructura MQRFH contiene los campos siguientes; los campos se describen en **orden alfabético**:

CodedCharSetId (MQLONG)

Especifica el identificador de juego de caracteres de los datos que siguen a *NameValueString*; no se aplica a los datos de tipo carácter de la propia estructura MQRFH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

MQCCSI_INHERIT
Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

MQCCSI_INHERIT no se puede utilizar si el valor del campo *PutApplType* en MQMD es MQAT_BROKER.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Encoding (MQLONG)

Especifica la codificación numérica de los datos que siguen a *NameValueString*; no se aplica a los datos numéricos de la propia estructura MQRFH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es MQENC_NATIVE.

Flags (MQLONG)

Se puede especificar lo siguiente:

MQRFH_NONE
Sin distintivos.

El valor inicial de este campo es MQRFH_NONE.

Format (MQCHAR8)

Especifica el nombre de formato de los datos que siguen a *NameValueString*.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *Format* en MQMD.

El valor inicial de este campo es MQFMT_NONE.

Serie NameValue(MQCHARn)

Es una serie de caracteres de longitud variable que contiene pares de nombre/valor con el formato:

```
name1 value1 name2 value2 name3 value3 ...
```

Cada nombre o valor debe estar separado del nombre o valor adyacente por uno o más caracteres en blanco; estos espacios en blanco no son significativos. Un nombre o valor puede contener espacios en blanco significativos añadiendo como prefijo y sufijo el nombre o valor con comillas dobles; todos los caracteres entre las comillas dobles abiertas y las comillas dobles de cierre coincidentes se tratan como significativos. En el ejemplo siguiente, el nombre es FAMOUS_WORDSy el valor es `Hello World`:

```
FAMOUS_WORDS "Hello World"
```

Un nombre o valor puede contener cualquier carácter que no sea el carácter nulo (que actúa como delimitador para *NameValueString*). Sin embargo, para ayudar a la interoperatividad, una aplicación puede restringir los nombres a los siguientes caracteres:

- Primer carácter: alfabético en mayúsculas o minúsculas (de la A a la Z, o de la a a la z), o subrayado.
- Caracteres subsiguientes: alfabético en mayúsculas o minúsculas, dígito decimal (de 0 a 9), subrayado, guión o punto.

Si un nombre o valor contiene una o más comillas dobles, el nombre o valor debe estar entre comillas dobles, y cada comillas dobles dentro de la serie debe duplicarse:

```
Famous_Words "The program displayed ""Hello World"""
```

Los nombres y valores distinguen entre mayúsculas y minúsculas, es decir, las letras minúsculas no se consideran iguales que las letras mayúsculas. Por ejemplo, FAMOUS_WORDS y `Famous_Words` son dos nombres diferentes.

La longitud en bytes de *NameValueString* es igual a *StrucLength* menos MQRFH_STRUC_LENGTH_FIXED. Para evitar problemas al convertir los datos de usuario en algunos entornos, haga que esta longitud sea un múltiplo de cuatro. Rellene *NameValueString* con espacios en blanco hasta esta longitud, o termine antes colocando un carácter nulo después del último carácter significativo de la serie. El carácter nulo y los bytes que le siguen, hasta la longitud especificada de *NameValueString*, se ignoran.

Nota: Debido a que la longitud de este campo no es fija, el campo se omite de las declaraciones de la estructura que se proporcionan para los lenguajes de programación soportados.

StrucId (MQCHAR4)

Este es el identificador de estructura; el valor debe ser:

MQRFH_STRUC_ID
Identificador de reglas y estructura de cabecera de formato.

Para el lenguaje de programación C, también se define la constante MQRFH_STRUC_ID_ARRAY; tiene el mismo valor que MQRFH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQRFH_STRUC_ID.

StrucLength (MQLONG)

Es la longitud en bytes de la estructura MQRFH, incluido el campo *NameValueString* al final de la estructura. La longitud *no* incluye los datos de usuario que siguen al campo *NameValueString* .

Para evitar problemas al convertir los datos de usuario en algunos entornos, *StrucLength* debe ser un múltiplo de cuatro.

La constante siguiente proporciona la longitud de la parte *fija* de la estructura, es decir, la longitud excluyendo el campo *NameValueString* :

MQRFH_STRUC_LENGTH_FIXED
Longitud de la parte fija de la estructura MQRFH.

El valor inicial de este campo es MQRFH_STRUC_LENGTH_FIXED.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQRFH_VERSION_1
Reglas de Version-1 y estructura de cabecera de formato.

El valor inicial de este campo es MQRFH_VERSION_1.

Valores iniciales y declaraciones de idioma para MQRFH

Tabla 531. Valores iniciales de los campos en MQRFH para MQRFH

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQRFH_STRUC_ID	'RFH¬'
Version	MQRFH_VERSION_1	1
StrucLength	MQRFH_STRUC_LENGTH_FIXED	32
Encoding	MQENC_NATIVE	Depende del entorno
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Espacios en blanco
Flags	MQRFH_NONE	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQRFH_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQRFH MyRFH = {MQRFH_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

MQRFH2 – Reglas y formato de la cabecera 2

En esta sección se describen las reglas y el formato de la cabecera 2, los campos que contiene y los valores iniciales de dichos campos.

Visión general de MQRFH2

Disponibilidad

Todos los sistemas WebSphere MQ , además de los clientes MQI de WebSphere MQ conectados a estos sistemas.

Finalidad

La cabecera MQRFH2 se basa en la cabecera MQRFH , pero permite que las series Unicode se transporten sin conversión, y puede transportar tipos de datos numéricos.

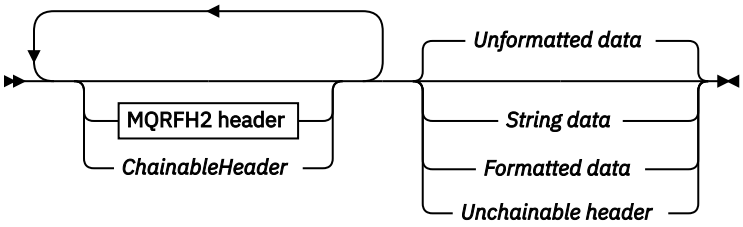
La estructura MQRFH2 define el formato de las reglas version-2 y la cabecera de formato. Utilice esta cabecera para enviar datos que se han codificado utilizando una sintaxis similar a XML. Un mensaje puede contener dos o más estructuras MQRFH2 en serie, con datos de usuario siguiendo opcionalmente la última estructura MQRFH2 de la serie.

Nombre de formato

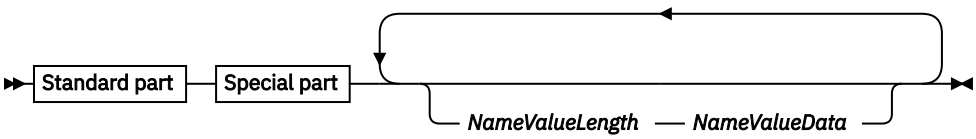
MQFMT_RF_HEADER_2

Syntax

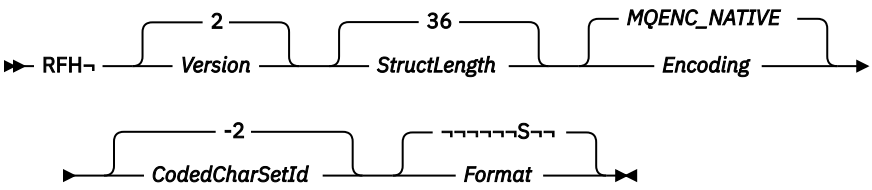
WebSphere MQ Message



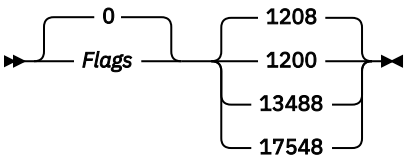
MQRFH2 header



Standard part



Special part



Juego de caracteres y codificación

Las reglas especiales se aplican al juego de caracteres y a la codificación utilizados para la estructura MQRFH2 :

- Los campos que no son *NameValueData* están en el juego de caracteres y la codificación proporcionados por los campos *CodedCharSetId* y *Encoding* en la estructura de cabecera que precede a MQRFH2, o por los campos en la estructura MQMD si MQRFH2 está al principio de los datos del mensaje de aplicación.

  El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de cola.

  Cuando se especifica MQGMO_CONVERT en la llamada MQGET , el gestor de colas convierte los campos MQRFH2 , distintos de *NameValueData*, en el juego de caracteres y la codificación solicitados.
- *NameValueData* está en el juego de caracteres proporcionado por el campo *NameValueCCSID* . Sólo los juegos de caracteres Unicode listados son válidos para *NameValueCCSID* ; consulte la descripción de *NameValueCCSID* para obtener más detalles.

  Algunos juegos de caracteres tienen una representación que depende de la codificación. Si *NameValueCCSID* es uno de estos juegos de caracteres, *NameValueData* debe estar en la misma codificación que los otros campos de MQRFH2.

  Cuando se especifica MQGMO_CONVERT en la llamada MQGET , el gestor de colas convierte *NameValueData* a la codificación solicitada, pero no cambia su juego de caracteres.

Campos para MQRFH2

La estructura MQRFH2 contiene los campos siguientes; los campos se describen en **orden alfabético**:

CodedCharSetId (MQLONG)

Especifica el identificador de juego de caracteres de los datos que siguen al último campo *NameValueData* ; no se aplica a los datos de tipo carácter de la propia estructura MQRFH2 .

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

MQCCSI_INHERIT
Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

MQCCSI_INHERIT no se puede utilizar si el valor del campo *PutApplType* en MQMD es MQAT_BROKER.

El valor inicial de este campo es MQCCSI_INHERIT.

Encoding (MQLONG)

Especifica la codificación numérica de los datos que siguen al último campo *NameValueData* ; no se aplica a los datos numéricos de la propia estructura MQRFH2 .

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es MQENC_NATIVE.

Flags (MQLONG)

El valor inicial de este campo es MQRFH_NONE. MQRFH_NONE debe especificarse.

MQRFH_NONE
Sin distintivos.

MQRFH_INTERNAL
La cabecera MQRFH2 contiene propiedades establecidas internamente.

MQRFH_INTERNAL es para uso del gestor de colas.

Los 16 bits principales, MQRFH_FLAGS_RESTRICTED_MASK, están reservados para los distintivos de los conjuntos de gestores de colas. Los distintivos que un usuario puede establecer se definen en los 16 bits inferiores.

Format (MQCHAR8)

Especifica el nombre de formato de los datos que siguen al último campo *NameValueData* .

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *Format* en MQMD.

El valor inicial de este campo es MQFMT_NONE.

NameValueCCSID (MQLONG)

Especifica el identificador de juego de caracteres codificado de los datos en el campo *NameValueData* . Es diferente del juego de caracteres de las otras series de la estructura MQRFH2 y puede ser diferente del juego de caracteres de los datos (si los hay) que siguen al último campo *NameValueData* al final de la estructura.

NameValueCCSID debe tener uno de los valores siguientes:

CCSID	Significado
1200	UCS-2 abierto
13488	UCS-2 2.0 subconjunto
17584	UCS-2 2.1 subconjunto (incluye el símbolo de euro)
1208	UTF-8

Para los juegos de caracteres UCS-2 , la codificación (orden de bytes) de *NameValueData* debe ser la misma que la codificación de los otros campos de la estructura MQRFH2 . Los caracteres sustitutos (X'D800'a X'DFFF') no están soportados.

Nota: Si *NameValueCCSID* no tiene uno de los valores listados anteriormente y la estructura MQRFH2 requiere conversión en la llamada MQGET, la llamada se completa con el código de razón MQRC_SOURCE_CCSID_ERROR y el mensaje se devuelve sin convertir.

El valor inicial de este campo es 1208.

Datos de NameValue(MQCHARn)
NameValueData es un campo de longitud variable que contiene una carpeta que contiene pares nombre/valor de propiedades de mensaje. Una carpeta es una serie de caracteres de longitud variable que contiene datos codificados utilizando una sintaxis similar a XML. La longitud en bytes de la serie de caracteres la proporciona el campo *NameValueLength* que precede al campo *NameValueData* . La longitud debe ser un múltiplo de cuatro.

Los campos *NameValueLength* y *NameValueData* son opcionales, pero si están presentes deben aparecer como un par y ser adyacentes. El par de campos se puede repetir tantas veces como sea necesario, por ejemplo:

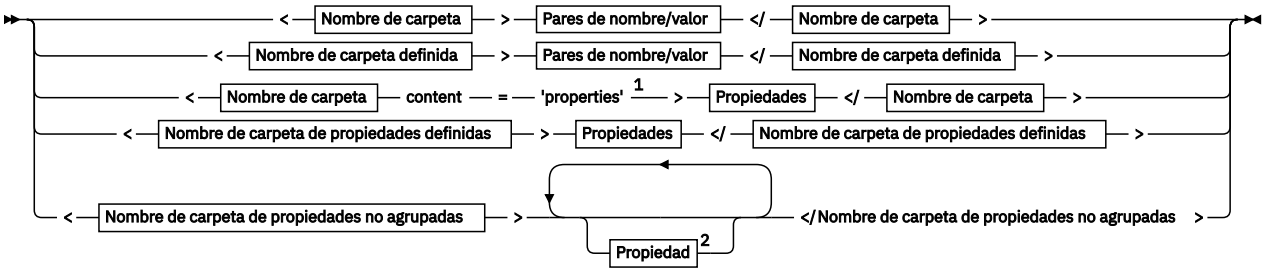
```
length1 data1 length2 data2 length3 data3
```

NameValueData no se convierte al juego de caracteres especificado en la llamada MQGET . Incluso si el mensaje se recupera con la opción MQGMO_CONVERT en vigor *NameValueData* permanece en su juego de caracteres original. Sin embargo, *NameValueData se* convierte a la codificación especificada en la llamada MQGET .

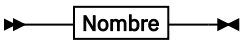
Nota: Puesto que estos campos son opcionales, se omiten de las declaraciones de la estructura que se proporcionan para los distintos lenguajes de programación soportados.

Nota: Los términos "definidos" y "reservados" se utilizan en el diagrama de sintaxis. "Definido" significa que IBM WebSphere MQutiliza el nombre. "Reservado" significa que el nombre está reservado para su uso futuro por parte de WebSphere MQ.

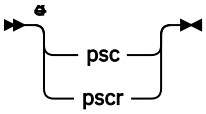
NameValueData sintaxis



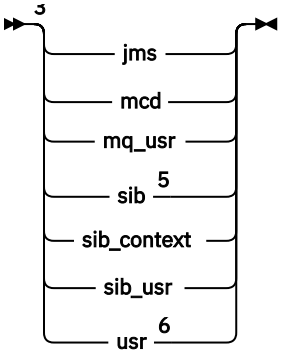
Nombre de carpeta



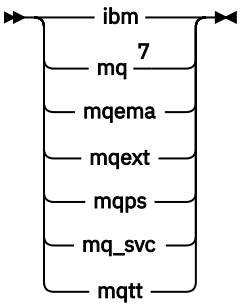
Nombre de carpeta definida



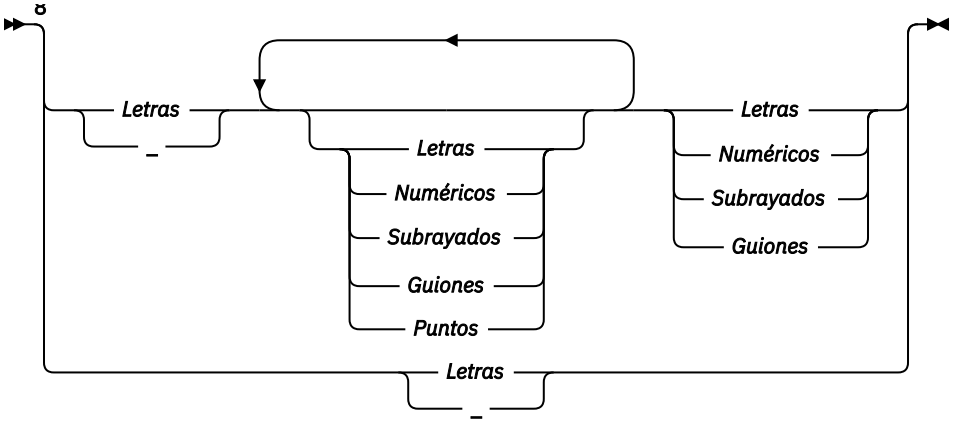
Nombre de carpeta de propiedades definidas



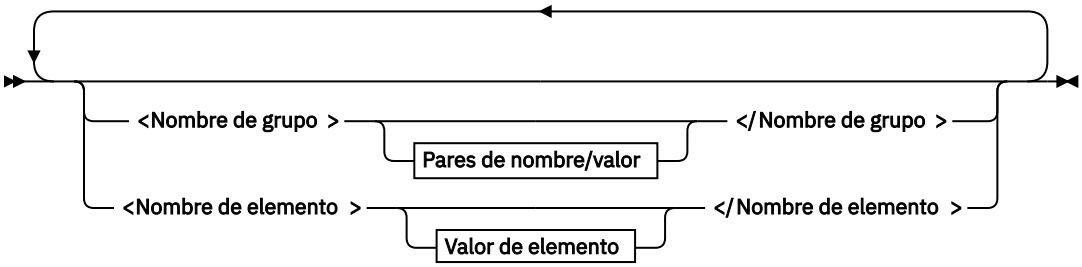
Nombre de carpeta de propiedades no agrupadas



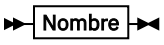
Nombre



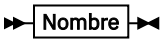
Pares de nombre/valor



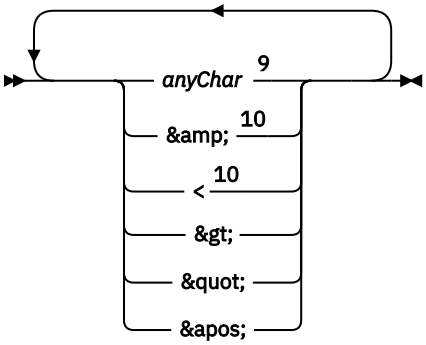
Nombre de grupo



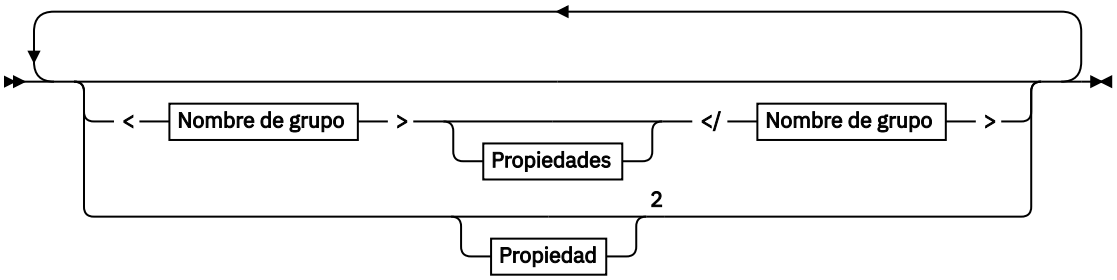
Nombre de elemento



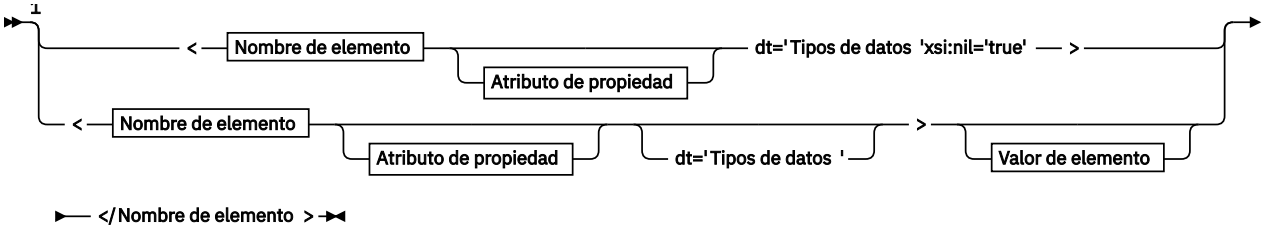
Valor de elemento



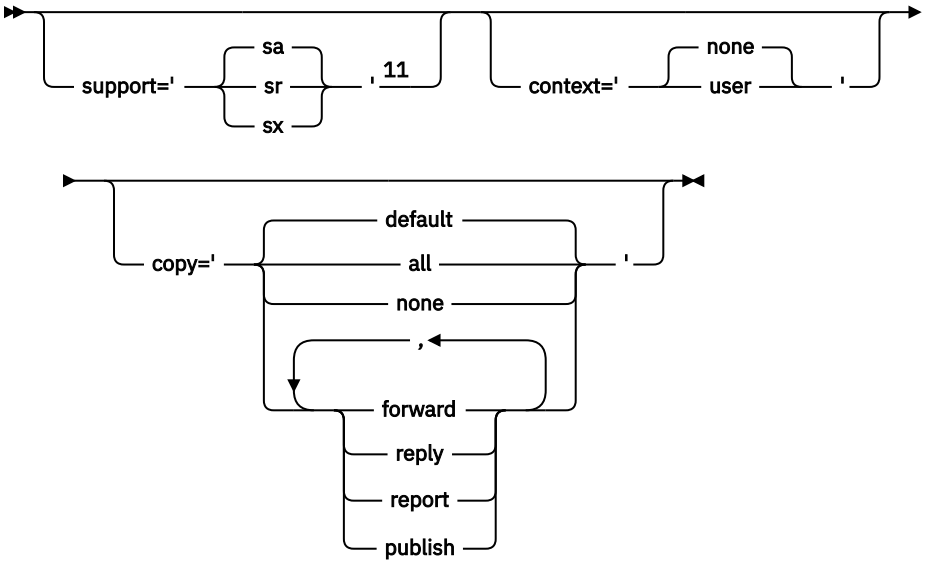
Propiedades



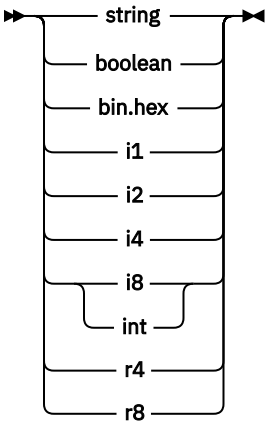
Propiedad



Atributo de propiedad



Tipos de datos



Notas:

[1] Las comillas dobles o las comillas simples son válidas.

[2] No utilice un nombre de propiedad no válido; consulte “Nombre de propiedad no válido” en la página 521. Utilice un nombre de propiedad reservado sólo para su finalidad definida; consulte “Nombres de propiedad definidos” en la página 521.

[3] El nombre debe estar en minúsculas.

[4] Solo se da soporte a una carpeta `psc` y `pscr` .

[5] Sólo las propiedades de la primera cabecera MQRFH2 son significativas. WebSphere Application Server Service Integration Bus ignora las carpetas sib, sib_contexty sib_usr en las cabeceras MQRFH2 posteriores.

[6] No debe haber más de una carpeta de `usr` en un MQRFH2. Las propiedades de la carpeta `usr` no deben aparecer más de una vez.

[7] Sólo son significativas las propiedades de la primera carpeta `mq` . Si la carpeta es UTF-8, sólo se da soporte a caracteres UTF-8 de un solo byte. El único carácter de espacio en blanco es Unicode U+0020.

[8] Los caracteres válidos se definen en la especificación XML W3C y constan esencialmente de categorías Unicode `Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm,` y `Nd`.

[9] Todos los caracteres son significativos. Los espacios en blanco iniciales y finales forman parte del valor del elemento.

[10] No utilice un carácter no válido; consulte “Caracteres no válidos” en la página 521. Utilice una secuencia de escape, en lugar de estos caracteres no válidos.

[11] El atributo de propiedad de soporte sólo es válido en la carpeta `mq`

Nombre de carpeta

NameValueData contiene una única carpeta. Para crear varias carpetas, cree varios campos *NameValueData* . Puede crear varios campos *NameValueData* en una única cabecera MQRFH2 dentro de un mensaje. De forma alternativa, puede crear varias cabeceras MQRFH2 encadenadas, cada una de las cuales contiene varios campos *NameValueData* .

El orden de las cabeceras MQRFH2 y el orden de los campos *NameValueData* no hacen ninguna diferencia en el contenido lógico de una carpeta. Si la misma carpeta está presente más de una vez en un mensaje, la carpeta se analiza como un todo. Si se produce la misma propiedad en varias instancias de la misma carpeta, se analiza como una lista.

Un análisis correcto de un MQRFH2 no se ve afectado por las formas alternativas en que una carpeta se puede almacenar físicamente en un mensaje.

Cuatro carpetas no siguen esta regla. Sólo se analiza la primera instancia de la carpeta `mq`, `sib`, `sib_contexty sib_usr` .

Si la misma propiedad aparece más de una vez en el contenido combinado de las cabeceras MQRFH2 encadenadas, sólo se analiza la primera instancia de la propiedad. Si una propiedad se establece utilizando una llamada de API, como por ejemplo MQSETMP, y se añade a un MQRFH2 directamente mediante una aplicación, la llamada de API tiene prioridad.

Un nombre de carpeta es el nombre de una carpeta que contiene pares o grupos de nombre/valor. Los grupos y los pares de nombre/valor se pueden mezclar en el mismo nivel en el árbol de carpetas; consulte Figura 1 en la página 510. No combine un nombre de grupo y un nombre de elemento; consulte Figura 2 en la página 511

```
<group1><nvp1>value</nvp1></group1><group2><nvp2>value</nvp2></group2>
<group3><nvp1>value</nvp1></group3><nvp3>value</nvp3>
```

Figura 1. Usos correctos de grupos y pares de nombre/valor

```
<group1><nvp1>value</nvp1>value</group1>
```

Figura 2. Uso incorrecto de grupos y pares de nombre/valor

No utilice un nombre de carpeta no válido o reservado; consulte “Nombre de vía de acceso no válido” en la página 521 y “Carpeta reservada o nombre de carpeta de propiedades” en la página 520. Utilice un nombre de carpeta definido sólo para su finalidad definida; consulte “Nombre de carpeta definida” en la página 512.

Si añade el atributo `'content=properties'` al código de nombre de carpeta, la carpeta se convierte en una carpeta de propiedades; consulte Figura 3 en la página 511.

```
<myFolder></myfolder>
<myPropertyFolder contents='properties'></myPropertyFolder>
```

Figura 3. Ejemplo de una carpeta y una carpeta de propiedades

Los nombres de carpeta distinguen entre mayúsculas y minúsculas. Los nombres de carpeta y los nombres de carpeta de propiedades comparten el mismo espacio de nombres. Deben tener nombres diferentes. `Folder1` en Figura 4 en la página 511 debe ser un nombre diferente a `Folder2` en Figura 5 en la página 511.

```
<Folder1><NVP1>value</NVP1></Folder1>
```

Figura 4. Espacio de nombres `Folder1`

```
<Folder2 content='properties'><Property1>value</Property1></Folder2>
```

Figura 5. Espacio de nombres `Folder2`

Los grupos, las propiedades y los pares de nombre/valor en carpetas diferentes tienen espacios de nombres diferentes. `Property1` en Figura 5 en la página 511 es una propiedad diferente de `Property1` en Figura 6 en la página 511.

```
<Folder3 content='properties'><Property1>value</Property1></Folder3>
```

Figura 6. Espacio de nombres `Folder3`

Las carpetas de propiedades son diferentes a las carpetas que no son de propiedades en dos aspectos importantes:

1. Las carpetas de propiedades contienen propiedades y las carpetas que no son de propiedades contienen pares de nombre/valor. Las carpetas difieren ligeramente, sintácticamente.
2. Utilice las interfaces definidas, como las propiedades MQI o las propiedades de mensaje JMS, para acceder a las propiedades de mensaje. Las interfaces garantizan que las carpetas de propiedades de MQRFH2 estén bien formadas. Una carpeta de propiedades bien formada es interoperable entre gestores de colas en distintas plataformas y distintos releases.

La propiedad de mensaje MQI es una forma sólida de leer y escribir un MQRFH2, y evita las dificultades de analizar un MQRFH2 correctamente.

Nombre de carpeta definida

Un nombre de carpeta definido es el nombre de una carpeta que está reservada para que la utilice WebSphere MQu otro producto. No cree una carpeta con el mismo nombre y no añada sus propios pares de nombre/valor a las carpetas. Las carpetas definidas son `psc` y `pscr`.

La publicación/suscripción en cola utiliza `psc` y `pscr` .

Un mensaje segmentado colocado con `MQMF_SEGMENT` o `MQMF_SEGMENTATION_ALLOWED` no puede contener un `MQRFH2` con un nombre de carpeta definido. El `MQPUT` falla con el código de razón 2443, `MQRC_SEGMENTATION_NOT_ALLOWED`.

Nombre de carpeta de propiedades definidas

Un nombre de carpeta de propiedades definido es el nombre de una carpeta de propiedades utilizada por IBM WebSphere MQu otro producto. Para ver los nombres de las carpetas y su contenido, consulte Carpetas de propiedades. Los nombres de carpeta de propiedades definidas son un subconjunto de todos los nombres de carpeta reservados por WebSphere MQ; consulte "Carpeta reservada o nombre de carpeta de propiedades" en la página 520.

Cualquier elemento almacenado en una carpeta de propiedades definida es una propiedad. Un elemento almacenado en una carpeta de propiedades definida no debe tener un atributo `content='properties'` .

Solo puede añadir propiedades a las carpetas de propiedades definidas `usr`, `mq_usr`y `sib_usr`. En otras carpetas de propiedades, como `mq` y `sib`, WebSphere MQ ignora o descarta las propiedades que no reconoce.

La descripción de cada carpeta de propiedades definida lista las propiedades que IBM WebSphere MQ ha definido que pueden utilizar los programas de aplicación. A algunas de las propiedades se accede indirectamente estableciendo u obteniendo una propiedad JMS, y a otras se accede directamente utilizando las llamadas MQI `MQSETMP` y `MQINQMP` .

Las carpetas de propiedades definidas también contienen otras propiedades que IBM WebSphere MQ ha reservado, pero a las que las aplicaciones no tienen acceso. Los nombres de las propiedades reservadas no se listan. No hay propiedades reservadas presentes en las carpetas de propiedades `usr`, `mq_usr`y `sib_usr` . Pero no cree propiedades con nombres de propiedad no válidos; consulte "Nombre de propiedad no válido" en la página 521.

Carpetas de propiedades

jms

`jms` contiene campos de cabecera JMS y propiedades JMSX que no se pueden expresar completamente en `MQMD`. La carpeta `jms` siempre está presente en un JMS `MQRFH2`.

Tabla 532. `jms` nombre de propiedad, sinónimo, tipo de datos y carpeta

Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
`JMSDestination`	`jms.Dst`	`string`	`<jms><Dst>`*`destination`*`</Dst></jms>`
`JMSExpiration`	`jms.Exp`	`i8`	`<jms><Exp>`*`expiration`*`</Exp></jms>`
`Correlación JMS`	`jms.Cid`	`string`	`<jms><Cid>`*`correlationId`*`</Cid></jms>`
`JMSDelivery`	`jms.Dlv`	`i4`	`<jms><Dlv>`*`delivery`*`</Dlv></jms>`

Tabla 532. jms nombre de propiedad, sinónimo, tipo de datos y carpeta (continuación)			
Sinónimo de propiedad	**Nombre de propiedad**	**Tipo de datos**	**Carpeta**
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>
JMSXGroupSeq	jms.Seq	i4	<jms><Seq>*messageSequenceNo*</Seq></jms>

No añada sus propias propiedades en la carpeta jms.

mcd

mcd contiene propiedades que describen el formato del mensaje. Por ejemplo, la propiedad de dominio de servicio de mensajes Msd identifica un mensaje JMS como JMSTextMessage, JMSBytesMessage, JMSStreamMessage, JMSMapMessage, JMSObjectMessage o nulo.

La carpeta mcd siempre está presente en un mensaje JMS que contiene un MQRFH2.

Siempre está presente en un mensaje que contiene un MQRFH2 enviado desde WebSphere Message Broker. Describe el dominio, formato, tipo y conjunto de mensajes de un mensaje.

Tabla 533. mcd nombre de propiedad, sinónimo, tipo de datos y carpeta			
Sinónimo de propiedad	**Nombre de propiedad**	**Tipo de datos**	**Carpeta**
	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>

No añada sus propias propiedades en la carpeta mcd.

mq_usr

mq_usr contiene propiedades definidas por la aplicación que no están expuestas como propiedades definidas por el usuario JMS. Las propiedades que no cumplen los requisitos de JMS se pueden colocar en esta carpeta.

Puede crear propiedades en la carpeta mq_usr . Las propiedades que crea en mq_usr son como las propiedades que crea en carpetas nuevas con el atributo content='properties' .

sib

sib contiene las propiedades de mensaje del sistema del bus de integración de servicios (WAS/SIB) de WebSphere Application Server. Las propiedades sib no se exponen como propiedades JMS en las aplicaciones JMS de IBM WebSphere MQ porque no son de los tipos soportados. Por ejemplo, algunas propiedades sib no se pueden exponer como propiedades JMS porque son matrices de bytes. Algunas propiedades sib se exponen a las aplicaciones WAS/SIB como propiedades JMS_IBM_* ; estas incluyen las propiedades de vías de acceso de direccionamiento inverso y de reenvío.

No añada sus propias propiedades en la carpeta sib.

sib_context

sib_context contiene propiedades de mensaje del sistema WAS/SIB que no están expuestas a aplicaciones de usuario WAS/SIB o como propiedades JMS. sib_context contiene propiedades de seguridad y transaccionales que se utilizan para los servicios web.

No añada sus propias propiedades en la carpeta sib_context.

sib_usr

sib_usr contiene propiedades de mensaje de usuario WAS/SIB que no están expuestas como propiedades de usuario JMS porque no son de tipos soportados. sib_usr está expuesto a aplicaciones WAS/SIB en la interfaz SIMessage ; consulte Desarrollo de integración de servicios.

El tipo de una propiedad sib_usr debe ser bin.hexy el valor debe estar en el formato correcto. Si una aplicación IBM WebSphere MQ escribe un elemento con tipo bin.hex en la carpeta con un formato incorrecto, la aplicación recibe un IOException. Si el tipo de datos de la propiedad no es bin.hex , la aplicación recibe un ClassCastException.

No intente que las propiedades de usuario JMS estén disponibles para WAS/SIB utilizando esta carpeta; en su lugar, utilice la carpeta usr .

Puede crear propiedades en la carpeta sib_usr .

usr

usr contiene propiedades JMS definidas por la aplicación asociadas al mensaje. La carpeta usr solo está presente si una aplicación ha establecido una propiedad definida por la aplicación.

usr es la carpeta de propiedades predeterminada. Si una propiedad se establece sin un nombre de carpeta, se coloca en la carpeta usr .

Tabla 534. usr nombre de propiedad, sinónimo, tipo de datos y carpeta.

Los valores de propiedad de servicios web se describen en MQRFH2 Valores SOAP

Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
	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>

Puede crear propiedades en la carpeta usr .

Un mensaje segmentado colocado con MQMF_SEGMENT o MQMF_SEGMENTATION_ALLOWED no puede contener un MQRFH2 con un nombre de carpeta de propiedades definido. El MQPUT falla con el código de razón 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Nombre de carpeta de propiedades no agrupadas

ibm

ibm contiene propiedades que sólo utiliza IBM WebSphere MQ.

Tabla 535. ibm nombre de propiedad, sinónimo, tipo de datos y carpeta

Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
	ibm.rfp	string	<ibm><rfp>fingerprint</rfp></ibm>

No añada sus propias propiedades en la carpeta ibm.

mq

mq contiene propiedades que sólo utiliza IBM WebSphere MQ.

Las restricciones siguientes se aplican a las propiedades de la carpeta mq :

- MQsólo actúa sobre las propiedades de la primera carpeta mq significativa del mensaje; las propiedades de cualquier otra carpeta mq del mensaje se ignoran.
- Sólo se permiten caracteres UTF-8 de un solo byte en la carpeta. Un carácter de varios bytes en la carpeta puede hacer que falle el análisis y que se rechace el mensaje.
- No utilice series de escape en la carpeta. Una serie de escape se trata como el valor real del elemento.
- Sólo el carácter Unicode U+0020 se trata como un espacio en blanco dentro de la carpeta. Todos los demás caracteres se tratan como significativos y pueden hacer que falle el análisis de la carpeta y que se rechace el mensaje.

Si el análisis de la carpeta mq falla, o si la carpeta no observa estas restricciones, el mensaje se rechaza con el código de razón 2527, MQRC_RFH_RESTRICTED_FORMAT_ERR.

No añada sus propias propiedades en la carpeta mq.

mqema

mqema contiene propiedades que sólo utiliza WebSphere Application Server. La carpeta se ha sustituido por mqext.

No añada sus propias propiedades en la carpeta mqema.

mqext

mqext contiene propiedades que sólo utiliza WebSphere Application Server. La carpeta sólo está presente si la aplicación ha establecido al menos una de las propiedades definidas de IBM .

Tabla 536. mqext nombre de propiedad, sinónimo, tipo de datos y carpeta

Sinónimo de propiedad	Nombre de propiedad	Tipo de datos	Carpeta
JMSArmCorrelator	mqext.Arm	string	<mqext><Arm>*armCorrelator*</Arm></mqext>

Tabla 536. mqext nombre de propiedad, sinónimo, tipo de datos y carpeta (continuación)			
Sinónimo de propiedad	**Nombre de propiedad**	**Tipo de datos**	**Carpeta**
JMSRMCorrelator	mqext.Wrm	string	<mqext><Wrm>*wrmCorrelator*</Wrm></mqext>

No añada sus propias propiedades en la carpeta mqext.

mqps

mqps contiene propiedades que solo son utilizadas por la publicación/suscripción de IBM WebSphere MQ. La carpeta sólo está presente si la aplicación ha establecido al menos una de las propiedades de publicación/suscripción integradas.

Tabla 537. mqps nombre de propiedad, sinónimo, tipo de datos y carpeta			
Sinónimo de propiedad	**Nombre de propiedad**	**Tipo de datos**	**Carpeta**
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>

No añada sus propias propiedades en la carpeta mqps.

mq_svc

mq_svc contiene propiedades utilizadas por SupportPac MA93.

No añada sus propias propiedades en la carpeta mq_svc.

mqtt

mqtt contiene propiedades utilizadas por IBM WebSphere MQ Telemetry

Tabla 538. mqtt nombre de propiedad, sinónimo, tipo de datos y carpeta			
Sinónimo de propiedad	**Nombre de propiedad**	**Tipo de datos**	**Carpeta**
	mqtt.clientId	string	<mqtt><clientId>*topicString*</clientId></mqtt>
	mqtt.qos	i4	<mqtt><qos>*qualityOfService*</qos></mqtt>
	mqtt.msgid	string	<mqtt><msgid>*messageIdentifier*</msgid></mqtt>

No añada sus propias propiedades en la carpeta mqtt.

Un mensaje segmentado colocado con MQMF_SEGMENT o MQMF_SEGMENTATION_ALLOWED no puede contener un MQRFH2 con un nombre de carpeta de propiedades no agrupado. El MQPUT falla con el código de razón 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Pares de nombre/valor

En el diagrama de sintaxis, "Pares de nombre/valor" describe el contenido de una carpeta ordinaria. Una carpeta ordinaria contiene grupos y elementos. Un elemento es un par nombre/valor. Un grupo contiene elementos y otros grupos.

En términos de árboles, los elementos son nodos hoja y los grupos son nodos internos. Un nodo interno y la carpeta, que es el nodo raíz, pueden contener una mezcla de nodos internos y nodos hoja. Un nodo no puede ser un nodo interno y un nodo de hoja al mismo tiempo; consulte Figura 2 en la página 511.

Propiedades

En el diagrama de sintaxis, "Propiedades" describe el contenido de una carpeta de propiedades. Una carpeta de propiedades contiene grupos y propiedades. Una propiedad es un par nombre/valor con un atributo de tipo de datos opcional. Un grupo contiene propiedades y otros grupos.

En términos de árboles, las propiedades son nodos hoja y los grupos son nodos internos. Un nodo interno y la carpeta de propiedades, que es el nodo raíz, pueden contener una mezcla de nodos internos y nodos hoja. Un nodo no puede ser un nodo interno y un nodo de hoja al mismo tiempo; consulte Figura 2 en la página 511.

Propiedad

Una propiedad de mensaje es un par de nombre/valor en una carpeta de propiedades. Opcionalmente, puede incluir un atributo de tipo de datos y un atributo de propiedad; para ver un ejemplo, consulte Figura 7 en la página 517. Si se omite el atributo de tipo de datos, el tipo de propiedad es string.

```
<pf><p1 dt='i8' >value</p1></pf>
```

Figura 7. Atributo Tipo de datos

El nombre de una propiedad de mensaje es su nombre completo de vía de acceso, con la sintaxis <> de tipo XML, sustituida por puntos. Por ejemplo, myPropertyFolder1.myGroup1.myGroup2.myProperty1 se correlaciona con una serie *NameValueData* en Figura 8 en la página 518. La serie se formatea para facilitar la lectura.

```
<myPropertyFolder1>
    <myGroup1>
        <myGroup2>
            <myProperty1>value</myProperty1>
        </myGroup2>
    </myGroup1>
</myPropertyFolder1>
```

Figura 8. Correlación de nombre de propiedad único

Una carpeta de propiedades puede contener varias propiedades. Por ejemplo, las propiedades en Figura 9 en la página 518 se correlacionan con la carpeta de propiedades en Figura 10 en la página 518

```
myPropertyFolder1.myProperty4
myPropertyFolder1.myGroup1.myGroup2.myProperty1
myPropertyFolder1.myGroup1.myGroup2.myProperty2
myPropertyFolder1.myGroup1.myProperty3
```

Figura 9. Varias propiedades con el mismo nombre raíz

```
<myPropertyFolder1>
    <myProperty4>value</myProperty4>
    <myGroup1>
        <myGroup2>
            <myProperty1>value</myProperty1>
            <myProperty2>value</myProperty2>
        </myGroup2>
        <myProperty3>value</myProperty3>
    </myGroup1>
</myPropertyFolder1>
```

Figura 10. Correlación de varios nombres de propiedad

Nombre

Un nombre debe empezar por una *Carta* o un *Subrayado*. No debe contener un *Colón*, no debe terminar en un *Periodo* y contener solo *Cartas*, *Numerales*, *Subrayados*, *Guiones*y *Puntos*. Los caracteres válidos se definen en la especificación XML W3C y constan esencialmente de categorías Unicode `Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm,` y `Nd`.

La vía de acceso completa de una propiedad o par nombre/valor no debe romper la regla descrita en “Nombre de vía de acceso no válido” en la página 521. Las vías de acceso están restringidas a 4095 bytes, no deben contener caracteres de compatibilidad Unicode y no deben empezar por la serie `XML`.

Nombre de grupo

Un nombre de grupo tiene la misma sintaxis que un nombre. Los nombres de grupo son opcionales. Las propiedades y los pares de nombre/valor se pueden colocar en la raíz de una carpeta. Utilice grupos si ayuda a organizar las propiedades y los pares de nombre/valor.

Nombre de elemento

Un nombre de elemento tiene la misma sintaxis que un nombre.

Valor de elemento

Un valor de elemento incluye todos los espacios en blanco entre el código <*Element name*> y el </*Element name*>. No utilice los dos caracteres < y & en un valor. Sustituya entonces por < y `&`.

Atributo de propiedad

Los campos de descriptor de propiedad de correlación de atributos de propiedad: las correlaciones son las siguientes:

Soporte

sa
MQPD_SUPPORT_OPTIONAL

sr
MQPD_SUPPORT_REQUIRED

sx
MQPD_SUPPORT_REQUIRED_IF_LOCAL

Contexto

none
MQPD_NO_CONTEXT

user
MQPD_USER _CONTEXT

CopyOptions

forward
MQPD_COPY_FORWARD

reply
MQPD_COPY_REPLY

report
MQPD_COPY_REPORT

publish
MQPD_COPY_PUBLISH

all
MQPD_COPY_ALL

No utilice `all` en combinación con otras opciones.

default
MQPD_COPY_DEFAULT

No utilice `default` en combinación con otras opciones. `default` es el mismo que `forward` + `report` + `publish`

none
MQPD_COPY_NONE

No utilice none en combinación con otras opciones.

Los atributos de la propiedad Soporte sólo son aplicables a las propiedades de la carpeta `mq` .

Los atributos de propiedad Contexto y CopyOptions son aplicables a todas las carpetas de propiedades.

Tipo de datos

Los tipos de datos de MQRFH2 se correlacionan con los tipos de propiedad de mensaje como se indica a continuación:

Tabla 539. Correlaciones de tipos de datos

Tipo de datos de MQRFH2	Tipo de propiedad de mensaje
bin.hex	MQBYTE[]
boolean	MQBOOL

Tabla 539. Correlaciones de tipos de datos (continuación)	
Tipo de datos de MQRFH2	**Tipo de propiedad de mensaje**
i1	MQINT8
i2	MQINT16
i4	MQINT32
i8	MQINT64
r4	MQFLOAT32
r8	MQFLOAT64
string	MQCHAR[]

Se presupone que cualquier elemento sin un tipo de datos es del tipo `string`.

Un valor nulo se indica mediante el atributo de elemento `xsi:nil='true'`. No utilice el atributo `xsi:nil='false'` para valores no nulos. Por ejemplo, la propiedad siguiente tiene un valor nulo:

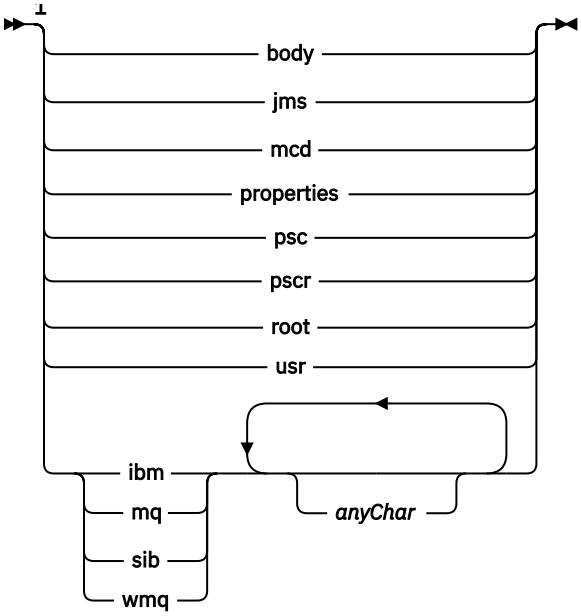
```
<NullProperty
xsi:nil='true'></NullProperty>
```

Una propiedad de serie de bytes o caracteres puede tener un valor vacío. Un valor vacío se representa mediante un elemento MQRFH2 con un valor de elemento de longitud cero. Por ejemplo, la propiedad siguiente tiene un valor vacío:

```
<EmptyProperty></EmptyProperty>
```

Carpeta reservada o nombre de carpeta de propiedades

Restrinja el nombre de una carpeta o carpeta de propiedades para que no empiece con ninguna de las series siguientes. Los prefijos están reservados para los nombres de carpeta o propiedad creados por IBM.

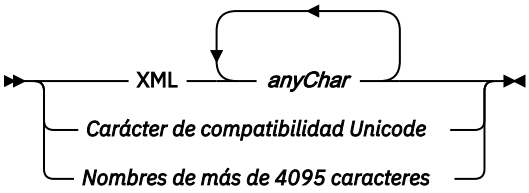


Notas:

[1] Un nombre de propiedad o carpeta reservado contiene cualquier combinación de letras minúsculas y mayúsculas.

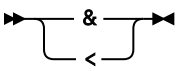
Nombre de vía de acceso no válido

Restrinja la vía de acceso completa de un par nombre/valor o una propiedad para que no incluya ninguna de las series siguientes.



Caracteres no válidos

Utilice siempre las secuencias de escape & y < en lugar de los literales "&" y "<".

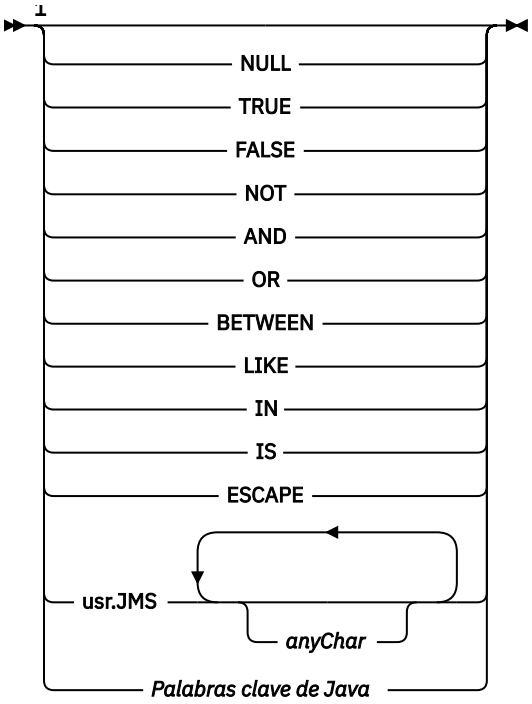


Nombres de propiedad definidos

Los nombres de propiedad definidos son los nombres de las propiedades definidas por WebSphere MQ, u otros productos, y utilizadas por IBM WebSphere MQ y las aplicaciones de usuario. Las propiedades definidas sólo existen en las carpetas de propiedades definidas. Los nombres de propiedad definidos se describen en la descripción de las carpetas de propiedades; consulte Carpetas de propiedades.

Nombre de propiedad no válido

No construya nombres de propiedad que coincidan con la regla siguiente. La regla se aplica a la vía de acceso de propiedad completa que nombra una propiedad y no sólo al nombre del elemento de propiedad.



Notas:

[1] Un nombre de propiedad no válido puede contener cualquier combinación de mayúsculas y minúsculas.

NameValueLongitud (MQLONG)
La longitud del campo `NameValueData` correspondiente

Especifica la longitud en bytes de los datos del campo *`NameValueData`* . *`NameValueLength`* debe ser un múltiplo de cuatro.

Nota: Los campos *`NameValueLength`* y *`NameValueData`* son opcionales, pero si están presentes deben aparecer como un par y ser adyacentes. El par de campos se puede repetir tantas veces como sea necesario, por ejemplo:

```
length1 data1 length2 data2 length3 data3
```

Puesto que estos campos son opcionales, se omiten de las declaraciones de la estructura que se proporcionan para los distintos lenguajes de programación soportados.

StrucId (MQCHAR4)

Este es el identificador de estructura; el valor debe ser:

MQRFH_STRUC_ID
Identificador de reglas y estructura de cabecera de formato.

Para el lenguaje de programación C, también se define la constante MQRFH_STRUC_ID_ARRAY; tiene el mismo valor que MQRFH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQRFH_STRUC_ID.

StrucLength (MQLONG)

Es la longitud en bytes de la estructura MQRFH2 , incluidos los campos *`NameValueLength`* y *`NameValueData`* al final de la estructura. Es válido para que haya varios pares de campos *`NameValueLength`* y *`NameValueData`* al final de la estructura, en la secuencia:

```
length1, data1, length2, data2, ...
```

`StrucLength` no incluye ningún dato de usuario que pueda seguir al último campo *`NameValueData`* al final de la estructura.

Para evitar problemas con la conversión de los datos de usuario en algunos entornos, *`StrucLength`* debe ser un múltiplo de cuatro.

La constante siguiente proporciona la longitud de la parte *fija* de la estructura, es decir, la longitud excluyendo los campos *`NameValueLength`* y *`NameValueData`* :

MQRFH_STRUC_LENGTH_FIXED_2
Longitud de la parte fija de la estructura MQRFH2 .

El valor inicial de este campo es MQRFH_STRUC_LENGTH_FIXED_2.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQRFH_VERSION_2
Version-2 reglas y estructura de cabecera de formato.

El valor inicial de este campo es MQRFH_VERSION_2.

Valores iniciales y declaraciones de idioma para MQRFH2

Tabla 540. Valores iniciales de los campos en MQRFH2 para MQRFH2

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQRFH_STRUC_ID	'RFH¬'
Version	MQRFH_VERSION_2	2
StrucLength	MQRFH_STRUC_LENGTH_FIXED_2	36
Encoding	MQENC_NATIVE	Depende del entorno
CodedCharSetId	MQCCSI_INHERIT	-2
Format	MQFMT_NONE	Espacios en blanco
Flags	MQRFH_NONE	0
NameValueCCSID	Ninguna	1208

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQRFH2_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQRFH2 MyRFH2 = {MQRFH2_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

MQRMH - Cabecera de mensaje de referencia

La tabla siguiente resume los campos de la estructura.

Tabla 541. Campos en MQRMH		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
StrucLength	Longitud total de MQRMH, incluidas las series situadas al final de los campos de longitud fija, pero no los datos generales	StrucLength
Encoding	Codificación numérica de los datos generales	Codificación
CodedCharSetId	Identificador del juego de caracteres de los datos generales	CodedCharSetId
Format	Nombre de formato de los datos generales	Formato
Flags	Distintivos de mensajes de referencia	Distintivos
ObjectType	Tipo de objeto	ObjectType
ObjectInstanceId	Identificador de la instancia de objeto	ObjectInstanceId
SrcEnvLength	Longitud de los datos del entorno origen	SrcEnvLongitud
SrcEnvOffset	Desplazamiento de los datos del entorno origen	SrcEnvDesplazamiento
SrcNameLength	Longitud del nombre de objeto origen	SrcNameLongitud
SrcNameOffset	Desplazamiento del nombre de objeto origen	SrcNameDesplazamiento
DestEnvLength	Longitud de los datos del entorno destino	DestEnvLongitud
DestEnvOffset	Desplazamiento de los datos del entorno destino	DestEnvDesplazamiento
DestNameLength	Longitud del nombre de objeto destino	DestNameLongitud
DestNameOffset	Desplazamiento del nombre de objeto destino	DestNameDesplazamiento
DataLogicalLength	Longitud de los datos generales	DataLogicalLength
DataLogicalOffset	Desplazamiento bajo de los datos generales	DataLogicalOffset
DataLogicalOffset2	Desplazamiento alto de los datos generales	DataLogicalOffset2

Visión general de MQRMH

Disponibilidad: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más clientes WebSphere MQ conectados a estos sistemas.

Finalidad: la estructura MQRMH define el formato de una cabecera de mensaje de referencia. Esta cabecera se utiliza con salidas de canal de mensajes escritas por el usuario para enviar cantidades extremadamente grandes de datos (denominadas *datos masivos*) de un gestor de colas a otro. La diferencia en comparación con la mensajería normal es que los datos masivos no se almacenan en una cola; en su lugar, sólo se almacena en la cola una *referencia* a los datos masivos. Esto reduce la posibilidad de que los recursos de MQ se agoten en un pequeño número de mensajes extremadamente grandes.

Nombre de formato: MQFMT_REF_MSG_HEADER.

Conjunto y codificación de caracteres: los datos de caracteres en MQRMH, y las series a las que se dirigen los campos de desplazamiento, deben estar en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas *CodedCharSetId* . Los datos numéricos en MQRMH

deben estar en la codificación de máquina nativa; esto viene dado por el valor de MQENC_NATIVE para el lenguaje de programación C.

Establezca el juego de caracteres y la codificación de MQRMH en los campos *CodedCharSetId* y *Encoding* en:

- El MQMD (si la estructura MQRMH está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQRMH (todos los demás casos).

Uso: una aplicación coloca un mensaje que consta de una MQRMH, pero omitiendo los datos masivos. Cuando un agente de canal de mensajes (MCA) lee el mensaje de la cola de transmisión, se invoca una salida de mensaje proporcionada por el usuario para procesar la cabecera de mensaje de referencia. La salida puede añadir al mensaje de referencia los datos masivos identificados por la estructura MQRMH, antes de que el MCA envíe el mensaje a través del canal al siguiente gestor de colas.

En el extremo receptor, debe existir una salida de mensajes que espere mensajes de referencia. Cuando se recibe un mensaje de referencia, la salida debe crear el objeto a partir de los datos masivos que siguen a la MQRMH en el mensaje y, a continuación, pasar el mensaje de referencia sin los datos masivos. El mensaje de referencia puede ser recuperado posteriormente por una aplicación que lee el mensaje de referencia (sin los datos masivos) de una cola.

Normalmente, la estructura MQRMH es todo lo que está en el mensaje. Sin embargo, si el mensaje está en una cola de transmisión, una o más cabeceras adicionales preceden a la estructura MQRMH.

También se puede enviar un mensaje de referencia a una lista de distribución. En este caso, la estructura MQDH y sus registros relacionados preceden a la estructura MQRMH cuando el mensaje está en una cola de transmisión.

Nota: No envíe un mensaje de referencia como un mensaje segmentado, porque la salida de mensaje no puede procesarlo correctamente.

Conversión de datos: A efectos de conversión de datos, la conversión de la estructura MQRMH incluye la conversión de los datos del entorno de origen, el nombre del objeto de origen, los datos del entorno de destino y el nombre del objeto de destino. Cualquier otro byte dentro de *StrucLength* bytes del inicio de la estructura se descarta o tiene valores no definidos después de la conversión de datos. Los datos masivos se convierten siempre que se cumpla lo siguiente:

- Los datos masivos están presentes en el mensaje cuando se realiza la conversión de datos.
- El campo *Format* de MQRMH tiene un valor distinto de MQFMT_NONE.
- Existe una salida de conversión de datos escrita por el usuario con el nombre de formato especificado.

Sin embargo, tenga en cuenta que normalmente los datos masivos *no* están presentes en el mensaje cuando el mensaje está en una cola y que, como resultado, los datos masivos se convierten mediante la opción MQGMO_CONVERT.

Campos para MQRMH

La estructura MQRMH contiene los campos siguientes; los campos se describen en **orden alfabético**:

CodedCharSetId (MQLONG)

Especifica el identificador de juego de caracteres de los datos masivos; no se aplica a los datos de tipo carácter de la propia estructura MQRMH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Se puede utilizar el siguiente valor especial:

MQCCSI_INHERIT
Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

No utilice MQCCSI_INHERIT si el valor del campo *PutApplType* en MQMD es MQAT_BROKER.

Este valor está soportado en los entornos siguientes: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más clientes WebSphere MQ conectados a estos sistemas.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Longitud DataLogical(MQLONG)

El campo *DataLogicalLength* especifica la longitud de los datos masivos a los que hace referencia la estructura MQRMH.

Si los datos masivos están realmente presentes en el mensaje, los datos empiezan en un desplazamiento de *StrucLength* bytes desde el inicio de la estructura MQRMH. La longitud de todo el mensaje menos *StrucLength* proporciona la longitud de los datos masivos presentes.

Si los datos están presentes en el mensaje, *DataLogicalLength* especifica la cantidad de datos que son relevantes. El caso normal es que *DataLogicalLength* tenga el mismo valor que la longitud de los datos presentes en el mensaje.

Si la estructura MQRMH representa los datos restantes en el objeto (empezando por el desplazamiento lógico especificado), puede utilizar el valor cero para *DataLogicalLength*, siempre que los datos masivos no estén realmente presentes en el mensaje.

Si no hay datos presentes, el final de MQRMH coincide con el final del mensaje.

El valor inicial de este campo es 0.

Desplazamiento DataLogical(MQLONG)

Este campo especifica el desplazamiento bajo de los datos masivos desde el inicio del objeto del que forman parte los datos masivos. El desplazamiento de los datos masivos desde el inicio del objeto se denomina *desplazamiento lógico*. *No* es el desplazamiento físico de los datos masivos desde el inicio de la estructura MQRMH; ese desplazamiento lo proporciona *StrucLength*.

Para permitir que se envíen objetos grandes utilizando mensajes de referencia, el desplazamiento lógico se divide en dos campos, y el desplazamiento lógico real viene dado por la suma de estos dos campos:

- *DataLogicalOffset* representa el resto obtenido cuando el desplazamiento lógico se divide por 1 000 000 000. Por lo tanto, es un valor comprendido entre 0 y 999.999.999.
- *DataLogicalOffset2* representa el resultado obtenido cuando el desplazamiento lógico se divide por 1 000 000 000. Por lo tanto, es el número de múltiplos completos de 1 000 000 000 000 que existen en el desplazamiento lógico. El número de múltiplos está en el rango de 0 a 999.999.999.

El valor inicial de este campo es 0.

DataLogicalOffset2 (MQLONG)

Este campo especifica el desplazamiento alto de los datos masivos desde el inicio del objeto del que forman parte los datos masivos. Es un valor comprendido entre 0 y 999.999.999. Consulte *DataLogicalOffset* para obtener detalles.

El valor inicial de este campo es 0.

DestEnvLongitud (MQLONG)

Es la longitud de los datos de entorno de destino. Si este campo es cero, no hay datos de entorno de destino y se ignora *DestEnvOffset* .

DestEnvDesplazamiento (MQLONG)

Este campo especifica el desplazamiento de los datos de entorno de destino desde el inicio de la estructura MQRMH. El creador del mensaje de referencia puede especificar los datos del entorno de destino, si el creador los conoce. Por ejemplo, en Windows los datos de entorno de destino pueden ser la vía de acceso del directorio del objeto donde se van a almacenar los datos masivos. Sin embargo,

si el creador no conoce los datos de entorno de destino, es responsabilidad de la salida de mensajes proporcionada por el usuario determinar la información de entorno necesaria.

La longitud de los datos de entorno de destino la proporciona *DestEnvLength*; si esta longitud es cero, no hay datos de entorno de destino y se ignora *DestEnvOffset* . Si está presente, los datos de entorno de destino deben residir completamente dentro de *StrucLength* bytes desde el inicio de la estructura.

Las aplicaciones no deben suponer que los datos del entorno de destino son contiguos a ninguno de los datos direccionado por los campos *SrcEnvOffset*, *SrcNameOffset* y *DestNameOffset* .

El valor inicial de este campo es 0.

DestNameLongitud (MQLONG)

Longitud del nombre de objeto de destino. Si este campo es cero, no hay ningún nombre de objeto de destino y se ignora *DestNameOffset* .

DestNameDesplazamiento (MQLONG)

Este campo especifica el desplazamiento del nombre de objeto de destino desde el inicio de la estructura MQRMH. El nombre de objeto de destino puede ser especificado por el creador del mensaje de referencia, si el creador conoce esos datos. Sin embargo, si el creador no conoce el nombre del objeto de destino, es responsabilidad de la salida de mensajes proporcionada por el usuario identificar el objeto que se va a crear o modificar.

La longitud del nombre de objeto de destino la proporciona *DestNameLength*; si esta longitud es cero, no hay ningún nombre de objeto de destino y se ignora *DestNameOffset* . Si está presente, el nombre de objeto de destino debe residir completamente dentro de *StrucLength* bytes desde el inicio de la estructura.

Las aplicaciones no deben suponer que el nombre de objeto de destino es contiguo a ninguno de los datos a los que se dirigen los campos *SrcEnvOffset*, *SrcNameOffset*y *DestEnvOffset* .

El valor inicial de este campo es 0.

Encoding (MQLONG)

Especifica la codificación numérica de los datos masivos; no se aplica a los datos numéricos de la propia estructura MQRMH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es MQENC_NATIVE.

Flags (MQLONG)

Estos son distintivos de mensaje de referencia. Se definen los distintivos siguientes:

MQRMHF_LAST
Este distintivo indica que el mensaje de referencia representa o contiene la última parte del objeto referenciado.

MQRMHF_NOT_LAST
El mensaje de referencia no contiene ni representa la última parte del objeto. MQRMHF_NOT_LAST ayuda a la documentación del programa. No se pretende que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

El valor inicial de este campo es MQRMHF_NOT_LAST.

Format (MQCHAR8)

Especifica el nombre de formato de los datos masivos.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *Format* en MQMD.

El valor inicial de este campo es MQFMT_NONE.

ObjectInstanceId (MQBYTE24)

Utilice este campo para identificar una instancia específica de un objeto. Si no es necesario, establézcalo en el valor siguiente:

MQOII_NONE
No se ha especificado ningún identificador de instancia de objeto. El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQOII_NONE_ARRAY; tiene el mismo valor que MQOII_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_OBJECT_INSTANCE_ID_LENGTH. El valor inicial de este campo es MQOII_NONE.

ObjectType (MQCHAR8)

Es un nombre que la salida de mensajes puede utilizar para reconocer los tipos de mensajes de referencia a los que da soporte. El nombre debe cumplir las mismas reglas que el campo *Format* descrito anteriormente.

El valor inicial de este campo es de 8 blancos.

SrcEnvLongitud (MQLONG)

La longitud de los datos de entorno de origen. Si este campo es cero, no hay datos de entorno de origen y se ignora *SrcEnvOffset* .

El valor inicial de este campo es 0.

SrcEnvDesplazamiento (MQLONG)

Este campo especifica el desplazamiento de los datos de entorno de origen desde el inicio de la estructura MQRMH. El creador del mensaje de referencia puede especificar los datos de entorno de origen, si el creador los conoce. Por ejemplo, en Windows los datos de entorno de origen pueden ser la vía de acceso del directorio del objeto que contiene los datos masivos. Sin embargo, si el creador no conoce los datos de entorno de origen, la salida de mensaje proporcionada por el usuario debe determinar la información de entorno necesaria.

La longitud de los datos de entorno de origen la proporciona *SrcEnvLength*; si esta longitud es cero, no hay datos de entorno de origen y se ignora *SrcEnvOffset* . Si está presente, los datos de entorno de origen deben residir completamente dentro de *StrucLength* bytes desde el inicio de la estructura.

Las aplicaciones no deben suponer que los datos de entorno se inician inmediatamente después del último campo fijo de la estructura o que son contiguos con cualquiera de los datos direccionados por los campos *SrcNameOffset*, *DestEnvOffset* y *DestNameOffset* .

El valor inicial de este campo es 0.

SrcNameLongitud (MQLONG)

Longitud del nombre de objeto de origen. Si este campo es cero, no hay ningún nombre de objeto de origen y se ignora *SrcNameOffset* .

El valor inicial de este campo es 0.

SrcNameDesplazamiento (MQLONG)

Este campo especifica el desplazamiento del nombre de objeto de origen desde el inicio de la estructura MQRMH. El nombre de objeto de origen puede ser especificado por el creador del mensaje de referencia, si el creador conoce esos datos. Sin embargo, si el creador no conoce el nombre del objeto de origen, la salida de mensaje proporcionada por el usuario debe identificar el objeto al que se debe acceder.

La longitud del nombre de objeto de origen la proporciona *SrcNameLength*; si esta longitud es cero, no hay ningún nombre de objeto de origen y se ignora *SrcNameOffset* . Si está presente, el nombre

de objeto de origen debe residir completamente dentro de *StrucLength* bytes desde el inicio de la estructura.

Las aplicaciones no deben suponer que el nombre de objeto de origen es contiguo a ninguno de los datos a los que se dirigen los campos *SrcEnvOffset*, *DestEnvOffset*y *DestNameOffset* .

El valor inicial de este campo es 0.

StrucId (MQCHAR4)

Este es el identificador de estructura; el valor debe ser:

MQRMH_STRUC_ID
Identificador de la estructura de cabecera de mensaje de referencia.

Para el lenguaje de programación C, la constante MQRMH_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQRMH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQRMH_STRUC_ID.

StrucLength (MQLONG)
La longitud total de MQRMH, incluidas las series al final de los campos fijos, pero no los datos masivos.

El valor inicial de este campo es cero.

Versión (MQLONG)

El número de versión de la estructura. El valor debe ser:

MQRMH_VERSION_1
Estructura de cabecera de mensaje de referencia Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQRMH_CURRENT_VERSION
Versión actual de la estructura de cabecera de mensaje de referencia.

El valor inicial de este campo es MQRMH_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQRMH

Tabla 542. Valores iniciales de campos en MQRMH para MQRMH

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQRMH_STRUC_ID	'RMH¬'
Version	MQRMH_VERSION_1	1
StrucLength	Ninguna	0
Encoding	MQENC_NATIVE	Depende del entorno
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Espacios en blanco
Flags	MQRMHF_NOT_LAST	0
ObjectType	Ninguna	Espacios en blanco
ObjectInstanceId	MQOII_NONE	Nulos
SrcEnvLength	Ninguna	0
SrcEnvOffset	Ninguna	0
SrcNameLength	Ninguna	0
SrcNameOffset	Ninguna	0

Tabla 542. Valores iniciales de campos en MQRMH para MQRMH (continuación)		
Nombre de campo	**Nombre de constante**	**Valor de constante**
DestEnvLength	Ninguna	0
DestEnvOffset	Ninguna	0
DestNameLength	Ninguna	0
DestNameOffset	Ninguna	0
DataLogicalLength	Ninguna	0
DataLogicalOffset	Ninguna	0
DataLogicalOffset2	Ninguna	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQRMH_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQRMH MyRMH = {MQRMH_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

MQRR-Registro de respuesta

La tabla siguiente resume los campos de la estructura.

Tabla 543. Campos en MQRR

Campo	Descripción	Tema
CompCode	Código de terminación para la cola	CompCode
Reason	Código de razón para la cola	Razón

Visión general de MQRR

Disponibilidad: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más clientes WebSphere MQ conectados a estos sistemas.

Finalidad: Utilice la estructura MQRR para recibir el código de terminación y el código de razón resultantes de la operación de apertura o colocación para una única cola de destino, cuando el destino es una lista de distribución. MQRR es una estructura de salida para las llamadas MQOPEN, MQPUT y MQPUT1 .

Conjunto de caracteres y codificación: los datos de MQRR deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Uso: al proporcionar una matriz de estas estructuras en las llamadas MQOPEN y MQPUT, o en la llamada MQPUT1 , puede determinar los códigos de terminación y los códigos de razón para todas las colas de una lista de distribución cuando el resultado de la llamada se mezcla, es decir, cuando la llamada es satisfactoria para algunas colas de la lista pero falla para otras. El código de razón MQRC_MULTIPLE_REASON de la llamada indica que el gestor de colas ha establecido los registros de respuesta (si los proporciona la aplicación).

Campos para MQRR

La estructura MQRR contiene los campos siguientes; los campos se describen en **orden alfabético**:

CompCode (MQLONG)

Es el código de terminación resultante de la operación de apertura o colocación para la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 .

Siempre es un campo de salida. El valor inicial de este campo es MQCC_OK.

Razón (MQLONG)

Este es el código de razón resultante de la operación de apertura o colocación para la cola con el nombre especificado por el elemento correspondiente en la matriz de estructuras MQOR proporcionada en la llamada MQOPEN o MQPUT1 .

Siempre es un campo de salida. El valor inicial de este campo es MQRC_NONE.

Valores iniciales y declaraciones de lenguaje para MQRR

Tabla 544. Valores iniciales de campos en MQRR para MQRR

Nombre de campo	Nombre de constante	Valor de constante
CompCode	MQCC_OK	0
Reason	MQRC_NONE	0

Notas:

1. En el lenguaje de programación C, la variable de macroMQRR_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQRR MyRR = {MQRR_DEFAULT};
```

Declaración C

```
typedef struct tagMQRR MQRR;
struct tagMQRR {
  MQLONG  CompCode;  /* Completion code for queue */
  MQLONG  Reason;    /* Reason code for queue */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
dcl
 1 MQRR based,
  3 CompCode fixed bin(31), /* Completion code for queue */
  3 Reason   fixed bin(31); /* Reason code for queue */
```

Declaración de Visual Basic

```
Type MQRR
  CompCode As Long 'Completion code for queue'
  Reason   As Long 'Reason code for queue'
End Type
```

MQSCO-Opciones de configuración SSL

La tabla siguiente resume los campos de la estructura.

Tabla 545. Entrada de campos MQSCO.

Lista de campos en MQSCO, por versión, con enlaces a los temas que describen los campos.

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
KeyRepository	Ubicación de repositorio de claves	KeyRepository
CryptoHardware	Detalles de hardware de cifrado	CryptoHardware
AuthInfoRecCount	Número de registros MQAIR presentes	AuthInfoRecCount
AuthInfoRecOffset	Desplazamiento del primer registro MQAIR desde el inicio de MQSCO	AuthInfoRecOffset
AuthInfoRecPtr	Dirección del primer registro MQAIR	AuthInfoRecPtr
Nota: Los dos campos siguientes se ignoran si *Version* es menor que MQSCO_VERSION_2.		
KeyResetCount	Número de restablecimiento de clave secreta SSL	Recuento deKeyReset
FipsRequired	Utilizar algoritmos criptográficos certificados por FIPS en WebSphere MQ	“FipsRequired (MQLONG)” en la página 538
Nota: El campo siguiente se ignora si *Version* es menor que MQSCO_VERSION_3.		
EncryptionPolicySuiteB	Utilizar solo algoritmos criptográficos Suite B	EncryptionPolicySuiteB
Nota: El campo siguiente se ignora si *Version* es menor que MQSCO_VERSION_4.		
CertificateValPolicy	Política de validación de certificado	PolíticaCertificateVal

Referencia relacionada

“MQCNO - Opciones de conexión” en la página 297
La tabla siguiente resume los campos de la estructura.

“Visión general de MQSCO” en la página 536
Disponibilidad: clientes AIX, HP-UX, IBM i, Solaris, Linux y Windows .

“Campos para MQSCO” en la página 536

“Valores iniciales y declaraciones de lenguaje para MQSCO” en la página 540

Visión general de MQSCO

Disponibilidad: clientes AIX, HP-UX, IBM i, Solaris, Linux y Windows .

Finalidad: La estructura MQSCO (junto con los campos SSL de la estructura MQCD) permite a una aplicación que se ejecuta como un cliente MQI de WebSphere MQ especificar opciones de configuración que controlan el uso de SSL para la conexión de cliente cuando el protocolo de canal es TCP/IP. La estructura es un parámetro de entrada en la llamada MQCONNX.

Si el protocolo de canal para el canal de cliente no es TCP/IP, se ignora la estructura MQSCO.

Conjunto de caracteres y codificación: los datos de MQSCO mustis se encuentran en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE.

Campos para MQSCO

La estructura MQSCO contiene los campos siguientes; los campos se describen en **orden alfabético**:

AuthInfoRecCount (MQLONG)

Es el número de registros de información de autenticación (MQAIR) a los que se dirigen los campos *AuthInfoRecPtr* o *AuthInfoRecOffset* . Para obtener más información, consulte “MQAIR-Registro de información de autenticación” en la página 252. El valor debe ser mayor o igual que cero. Si el valor no es válido, la llamada falla con el código de razón MQRC_AUTH_INFO_REC_COUNT_ERROR.

Este es un campo de entrada. El valor inicial de este campo es 0.

AuthInfoRecOffset (MQLONG)

Es el desplazamiento en bytes del primer registro de información de autenticación desde el inicio de la estructura MQSCO. El desplazamiento puede ser positivo o negativo. El campo se ignora si *AuthInfoRecCount* es cero.

Puede utilizar *AuthInfoRecOffset* o *AuthInfoRecPtr* para especificar los registros MQAIR, pero no ambos; consulte la descripción del campo *AuthInfoRecPtr* para obtener más detalles.

Este es un campo de entrada. El valor inicial de este campo es 0.

AuthInfoRecPtr (PMQAIR)

Es la dirección del primer registro de información de autenticación. El campo se ignora si *AuthInfoRecCount* es cero.

Puede proporcionar la matriz de registros MQAIR de una de estas dos maneras:

- Utilizando el campo de puntero *AuthInfoRecPtr*

  En este caso, la aplicación puede declarar una matriz de registros MQAIR separada de la estructura MQSCO y establecer *AuthInfoRecPtr* en la dirección de la matriz.

  Considere la posibilidad de utilizar *AuthInfoRecPtr* para lenguajes de programación que den soporte al tipo de datos de puntero de una forma que sea portable a distintos entornos (por ejemplo, el lenguaje de programación C).

- Utilizando el campo de desplazamiento *AuthInfoRecOffset*

  En este caso, la aplicación debe declarar una estructura compuesta que contenga una MQSCO seguida de la matriz de registros MQAIR y establecer *AuthInfoRecOffset* en el desplazamiento del primer registro de la matriz desde el inicio de la estructura MQSCO. Asegúrese de que este valor es correcto y tiene un valor que se puede acomodar en un MQLONG (el lenguaje de programación más restrictivo es COBOL, para el que el rango válido es de -999 999 999 a +999 999 999 999).

  Considere la posibilidad de utilizar *AuthInfoRecOffset* para lenguajes de programación que no dan soporte al tipo de datos de puntero, o que implementan el tipo de datos de puntero de una forma que no es portable a entornos diferentes (por ejemplo, el lenguaje de programación COBOL).

Sea cual sea la técnica que elija, sólo se puede utilizar uno de *AuthInfoRecPtr* y *AuthInfoRecOffset* ; la llamada falla con el código de razón MQRC_AUTH_INFO_REC_ERROR si ambos son distintos de cero.

Este es un campo de entrada. El valor inicial de este campo es el puntero nulo en los lenguajes de programación que dan soporte a punteros y, de lo contrario, una serie de bytes totalmente nula.

Nota: En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Política CertificateVal(MQLONG)

Este campo especifica qué tipo de política de validación de certificados se utiliza. El campo se puede establecer en uno de los valores siguientes:

MQ_CERT_VAL_POLICY_ANY
Aplique cada una de las políticas de validación de certificados soportadas por la biblioteca de sockets seguros. Acepte la cadena de certificados si alguna de las políticas considera válida la cadena de certificados.

MQ_CERT_VAL_POLICY_RFC5280
Aplique sólo la política de validación de certificados compatible con RFC5280 . Este valor proporciona una validación más estricta que el valor ANY, pero rechaza algunos certificados digitales más antiguos.

El valor inicial de este campo es MQ_CERT_VAL_POLICY_ANY

CryptoHardware (MQCHAR256)
Este campo proporciona detalles de configuración para el hardware criptográfico conectado al sistema cliente.

Establezca el campo en una serie con el formato siguiente, o déjelo en blanco o nulo:

```
GSK_PKCS11=<the PKCS #11 driver path and file name>;<the PKCS #11
token label>;<the PKCS #11 token password>;<symmetric cipher setting>;
```

Para utilizar hardware criptográfico que se ajuste a la interfaz PKCS #11 , por ejemplo, se deben especificar las series de contraseña IBM 4960 o IBM 4764, la vía de acceso del controlador PKCS #11 , la etiqueta de señal PKCS #11 y la contraseña de señal PKCS #11 , cada una de las cuales termina con un punto y coma.

La vía de acceso del controlador PKCS #11 es una vía de acceso absoluta a la biblioteca compartida que proporciona soporte para la tarjeta PKCS #11 . El nombre del archivo de controlador PKCS #11 es el nombre de la biblioteca compartida. Un ejemplo del valor necesario para la vía de acceso PKCS #11 y el nombre de archivo es:

```
/usr/lib/pkcs11/PKCS11_API.so
```

La etiqueta de señal PKCS #11 debe estar totalmente en minúsculas. Si ha configurado el hardware con una etiqueta de señal en mayúsculas o en mayúsculas, vuelva a configurarlo con esta etiqueta en minúsculas.

Si no es necesaria ninguna configuración de hardware criptográfico, establezca el campo en blanco o nulo.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellena el valor con blancos hasta la longitud del campo. Si el valor no es válido, o se produce una anomalía cuando se utiliza para configurar el hardware de cifrado, la llamada falla con el código de razón MQRC_CRYPTO_HARDWARE_ERROR.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_SSL_CRYPTO_HARDWARE_LENGTH. El valor inicial de este campo es la serie nula en C y los caracteres en blanco en otros lenguajes de programación.

EncryptionPolicySuiteB(MQLONG)

Este campo especifica si se utiliza la criptografía compatible con Suite B y qué nivel de fuerza se emplea. El valor puede ser uno o varios de los siguientes:

- MQ_SUITE_B_NONE

  No se utiliza la criptografía compatible con Suite B.
- MQ_SUITE_B_128_BIT

  Se utiliza la seguridad de potencia de 128 bits de Suite B.
- MQ_SUITE_B_192_BIT

  Se utiliza la seguridad de potencia de 192 bits de la suite B.

Nota: El uso de MQ_SUITE_B_NONE con cualquier otro valor en este campo no es válido.

FipsRequired (MQLONG)
WebSphere MQ se puede configurar con hardware criptográfico para que los módulos de criptografía utilizados sean los proporcionados por el producto de hardware; estos pueden estar certificados por FIPS a un nivel determinado en función del producto de hardware criptográfico en uso. Utilice este campo para especificar que sólo se utilicen algoritmos certificados por FIPS si la criptografía se proporciona en el software proporcionado por WebSphere MQ.

Cuando se instala WebSphere MQ , también se instala una implementación de criptografía SSL que proporciona algunos módulos certificados por FIPS.

Los valores pueden ser:

MQSSL_FIPS_NO
Este es el valor predeterminado. Cuando se establece en este valor:

- Se puede utilizar cualquier CipherSpec soportada en una plataforma determinada.
- Si se ejecuta sin utilizar el hardware de cifrado, las siguientes CipherSpecs se ejecutan utilizando la criptografía certificada FIPS 140-2 en las plataformas WebSphere MQ :
  - TLS_RSA_WITH_3DES_EDE_CBC_SHA
  - TLS_RSA_WITH_AES_128_CBC_SHA
  - TLS_RSA_WITH_AES_256_CBC_SHA

MQSSL_FIPS_YES
Cuando se establece en este valor, a menos que esté utilizando hardware criptográfico para realizar la criptografía, puede estar seguro de que

- Solo se pueden utilizar algoritmos criptográficos con certificado FIPS en la CipherSpec que se aplica a esta conexión de cliente.
- Las conexiones de canal SSL de entrada y salida sólo son satisfactorias si se utiliza una de las siguientes especificaciones de cifrado:
  - TLS_RSA_WITH_3DES_EDE_CBC_SHA
  - TLS_RSA_WITH_AES_128_CBC_SHA
  - TLS_RSA_WITH_AES_256_CBC_SHA

Notas:

1. La CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA está en desuso.
2. Siempre que sea posible, si se configuran CipherSpecs sólo para FIPS, el cliente MQI rechaza las conexiones que especifican una CipherSpec no FIPS con MQRC_SSL_INITIALIZATION_ERROR. WebSphere MQ no garantiza rechazar todas las conexiones de este tipo y es responsabilidad del usuario determinar si la configuración es compatible con WebSphere MQ.

distributed *KeyRepository (MQCHAR256)*

Este campo sólo es relevante para los clientes MQI de WebSphere MQ que se ejecutan en sistemas UNIX, Linuxy Windows . Especifica la ubicación del archivo de base de datos de claves en el que se almacenan las claves y los certificados. El archivo de base de datos de claves debe tener un nombre de archivo con el formato `zzz.kdb` , donde `zzz` es seleccionable por el usuario. El campo *KeyRepository* contiene la vía de acceso a este archivo, junto con la raíz del nombre de archivo (todos los caracteres del nombre de archivo hasta, pero sin incluir, el `.kdb`final). El sufijo de archivo `.kdb` se añade automáticamente.

Cada archivo de base de datos de claves tiene un *archivo de ocultación de contraseña*asociado. Esto contiene las contraseñas codificadas que se utilizan para permitir el acceso programático a la base de datos de claves. El archivo de ocultación de contraseña debe residir en el mismo directorio y tener la misma raíz de archivo que la base de datos de claves, y debe finalizar con el sufijo `.sth`.

Por ejemplo, si el campo *KeyRepository* tiene el valor `/xxx/yyy/key` , el archivo de base de datos de claves debe ser `/xxx/yyy/key.kdb` y el archivo de ocultación de contraseña debe ser `/xxx/yyy/key.sth` , donde `xxx` y `yyy` representan nombres de directorio.

Si el valor es más corto que la longitud del campo, termine el valor con un carácter nulo o rellena el valor con blancos hasta la longitud del campo. El valor no se comprueba; si hay un error al acceder al repositorio de claves, la llamada falla con el código de razón MQRC_KEY_REPOSITORY_ERROR.

Para ejecutar una conexión SSL desde un cliente MQI de WebSphere MQ , establezca *KeyRepository* en un nombre de archivo de base de datos de claves válido.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_SSL_KEY_REPOSITORY_LENGTH. El valor inicial de este campo es la serie nula en C y los caracteres en blanco en otros lenguajes de programación.

KeyResetCount (MQLONG)
Representa el número total de bytes no cifrados enviados y recibidos dentro de una conversación SSL o TLS antes de que se renegocie la clave secreta.

El número de bytes incluye la información de control que envía el MCA.

Si especifica una cuenta de restablecimiento de clave secreta SSL o TLS entre 1 byte y 32 Kb, los canales SSL o TLS utilizarán una cuenta de restablecimiento de clave secreta de 32 Kb. Esto es para evitar el coste de proceso de restablecimientos de clave excesivos que se producirían para valores de restablecimiento de clave secreta SSL o TLS pequeños.

Este es un campo de entrada. El valor es un número comprendido entre 0 y 999 999 999, con un valor por omisión de 0. Utilice un valor de 0 para indicar que las claves secretas nunca se renegocian.

StrucId (MQCHAR4)

Este es el identificador de estructura; el valor debe ser:

MQSCO_ID_ESTRUCTURA
Identificador para la estructura de opciones de configuración SSL.

Para el lenguaje de programación C, también se define la constante MQSCO_STRUC_ID_ARRAY; tiene el mismo valor que MQSCO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQSCO_STRUC_ID.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQSCO_VERSION_1
Estructura de opciones de configuración SSL de Version-1 .

MQSCO_VERSION_2
Estructura de opciones de configuración SSL de Version-2 .

MQSCO_VERSION_3
Estructura de opciones de configuración SSL de Version-3 .

MQSCO_VERSION_4
Version-4 Estructura de opciones de configuración SSL.

La constante siguiente especifica el número de versión de la versión actual:

MQSCO_VERSIÓN_ACTUAL
Versión actual de la estructura de opciones de configuración SSL.

Siempre es un campo de entrada. El valor inicial de este campo es MQSCO_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQSCO

Tabla 546. Valores iniciales de los campos en MQSCO.

Descripción de los campos en MQSCO y sus valores iniciales

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQSCO_STRUC_ID	'SCO¬'
Version	MQSCO_CURRENT_VERSION	1
KeyRepository	Ninguna	Serie nula o espacios en blanco
CryptoHardware	Ninguna	Serie nula o espacios en blanco
AuthInfoRecCount	Ninguna	0
AuthInfoRecOffset	Ninguna	0
AuthInfoRecPtr	Ninguna	Puntero nulo o bytes nulos
KeyResetCount	MQSCO_RESET_COUNT_DEFAULT	0
FipsRequired	MQSSL_FIPS_NO	0
EncryptionPolicySuiteB	MQ_SUITE_B_NONE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE	1, 0, 0, 0
CertificateValPolicy	MQ_CERT_VAL_POLICY_DEFAULT	0

Notes:

1. The symbol ¬ represents a single blank character.
2. In the C programming language, the macro variable MQSCO_DEFAULT contains the values listed above. Use it in the following way to provide initial values for the fields in the structure:

```
MQSCO MySCO = {MQSCO_DEFAULT};
```

Declaración C

```
typedef struct tagMQSCO MQSCO;
struct tagMQSCO {
   MQCHAR4    StrucId;                   /* Structure identifier */
   MQLONG     Version;                   /* Structure version number */
   MQCHAR256  KeyRepository;             /* Location of SSL 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 */
```

declaración COBOL

```
**   MQSCO structure
  10 MQSCO.
**    Structure identifier
   15 MQSCO-STRUCID                  PIC X(4).
**    Structure version number
   15 MQSCO-VERSION                  PIC S9(9) BINARY.
**    Location of SSL 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
```

Declaración PL/I

```
dcl
 1 MQSCO based,
  3 StrucId                   char(4),       /* Structure identifier */
  3 Version                   fixed bin(31), /* Structure version number */
  3 KeyRepository             char(256),     /* Location of SSL 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 */
```

Declaración de Visual Basic

```
Type MQSCO
  StrucId           As String*4   'Structure identifier'
  Version           As Long       'Structure version number'
  KeyRepository     As String*256 'Location of SSL 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
```

MQSD - Descriptor de suscripción

La tabla siguiente resume los campos de la estructura.

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Options	Opciones	Opciones
ObjectName	Nombre de objeto	ObjectName
AlternateUserId	ID de usuario alternativo	AlternateUserId
AlternateSecurityId	ID de seguridad alternativo	AlternateSecurityId
SubExpiry	Caducidad de suscripción	SubExpiry
ObjectString	Serie de objeto	ObjectString
SubName	Nombre de la suscripción	SubName
SubUserData	Datos de usuario de suscripción	SubUserData
SubCorrelId	ID de correlación de suscripción	SubCorrelId
PubPriority	Prioridad de publicación	PubPriority
PubAccountingToken	Señal de contabilidad de publicación	PubAccountingToken
PubAppIdentityData	Datos de identidad de aplicación de publicación	PubAppIdentityData
SelectionString	Serie que proporciona criterios de selección	SelectionString
SubLevel	Nivel de suscripción	SubLevel
ResObjectString	Nombre de objeto largo	ResObjectString

Visión general de MQSD

Disponibilidad: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS, más clientes MQI de WebSphere MQ conectados a estos sistemas.

Finalidad: la estructura MQSD se utiliza para especificar detalles sobre la suscripción que se está realizando.

La estructura es un parámetro de entrada/salida en la llamada MQSUB. Para obtener más información, consulte Notas de uso de MQSUB.

Suscripciones gestionadas: si una aplicación no tiene ninguna necesidad específica de utilizar una cola determinada como destino para las publicaciones que coinciden con su suscripción, puede utilizar la característica de suscripción gestionada. Si una aplicación opta por utilizar una suscripción gestionada, el gestor de colas informa al suscriptor sobre el destino donde se envían los mensajes publicados, proporcionando un descriptor de objeto como salida de la llamada MQSUB. Para obtener más información, consulte Hobj (MQHOBJ)-entrada/salida.

Cuando se elimina la suscripción, el gestor de colas también se compromete a limpiar los mensajes que no se han recuperado del destino gestionado, en las situaciones siguientes:

- Cuando se elimina la suscripción-mediante el uso de MQCLOSE con MQCO_REMOVE_SUB-y se cierra el Hobj gestionado.
- Implícita significa que cuando se pierde la conexión con una aplicación utilizando una suscripción no duradera (MQSO_NON_DURABLE)
- Por caducidad cuando se elimina una suscripción porque ha caducado y el Hobj gestionado está cerrado.

Debe utilizar suscripciones gestionadas con suscripciones no duraderas, para que se pueda realizar esta limpieza, y para que los mensajes de las suscripciones no duraderas cerradas no ocupen espacio en el gestor de colas. Las suscripciones duraderas también pueden utilizar destinos gestionados.

Versión: La versión actual de MQSD es MQSD_VERSION_1.

Conjunto de caracteres y codificación: los datos de MQSD deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos para MQSD

La estructura MQSD contiene los campos siguientes; los campos se describen en orden alfabético:

AlternateSecurityId (MQBYTE40)

Este es un identificador de seguridad que se transfiere con el AlternateUserId al servicio de autorizaciones para permitir que se realicen las comprobaciones de autorización correspondientes.

AlternateSecurityId sólo se utiliza si se especifica MQSO_ALTERNATE_USER_AUTHORITY y el campo AlternateUserId no está completamente en blanco hasta el primer carácter nulo o el final del campo.

Al volver de una llamada MQSUB utilizando MQSO_RESUME, este campo no se modifica.

Consulte la descripción de “AlternateSecurityId (MQBYTE40)” en la página 455 en el tipo de datos MQOD para obtener más información.

AlternateUserId (MQCHAR12)

Si especifica MQSO_ALTERNATE_USER_AUTHORITY, este campo contiene un identificador de usuario alternativo que se utiliza para comprobar la autorización para la suscripción y para la salida en la cola de destino (especificada en el parámetro *Hobj* de la llamada MQSUB), en lugar del identificador de usuario bajo el que se ejecuta actualmente la aplicación.

Si es satisfactorio, el identificador de usuario especificado en este campo se registra como el identificador de usuario propietario de la suscripción en lugar del identificador de usuario con el que se ejecuta actualmente la aplicación.

Si se especifica MQSO_ALTERNATE_USER_AUTHORITY y este campo está completamente en blanco hasta el primer carácter nulo o el final del campo, la suscripción sólo puede realizarse correctamente si no se necesita ninguna autorización de usuario para suscribirse a este tema con las opciones especificadas o la cola de destino para la salida.

Si no se especifica MQSO_ALTERNATE_USER_AUTHORITY, este campo se ignora.

Existen las siguientes diferencias en los entornos indicados:

- En z/OS, solo se utilizan los primeros 8 caracteres del ID AlternateUserpara comprobar la autorización para la suscripción. Sin embargo, el identificador de usuario actual debe estar autorizado para especificar este identificador de usuario alternativo concreto; para esta comprobación se utilizan los 12 caracteres del identificador de usuario alternativo. El identificador de usuario sólo debe contener caracteres permitidos por el gestor de seguridad externa.

Al volver de una llamada MQSUB utilizando MQSO_RESUME, este campo no se modifica.

Este es un campo de entrada. La longitud de este campo la proporciona MQ_USER_ID_LENGTH. El valor inicial de este campo es la serie nula en C y 12 caracteres en blanco en otros lenguajes de programación.

ObjectName (MQCHAR48)

Es el nombre del objeto de tema tal como se ha definido en el gestor de colas local.

El nombre puede contener los siguientes caracteres:

- Caracteres alfabéticos en mayúsculas (de la A a la Z)
- Caracteres alfabéticos en minúsculas (de la a a la z)
- Dígitos numéricos (de 0 a 9)
- Punto (.), barra inclinada (/), subrayado (_), porcentaje (%)

El nombre no debe contener espacios en blanco iniciales o intercalados, pero puede contener espacios en blanco finales. Utilice un carácter nulo para indicar el final de los datos significativos en el nombre; el nulo y los caracteres que le siguen se tratan como espacios en blanco. Las restricciones siguientes se aplican en los entornos indicados:

- En sistemas que utilizan EBCDIC Katakana, no se pueden utilizar caracteres en minúsculas.
- En z/OS:
  - Evite los nombres que empiezan o terminan con un guión bajo; no pueden ser procesados por las operaciones y los paneles de control.
  - El carácter de porcentaje tiene un significado especial para RACF. Si se utiliza RACF como gestor de seguridad externa, los nombres no deben contener el porcentaje. Si lo hacen, estos nombres no se incluyen en ninguna comprobación de seguridad cuando se utilizan perfiles genéricos RACF .
- En IBM i, los nombres que contienen caracteres en minúsculas, barras inclinadas o porcentaje deben estar entre comillas cuando se especifican en los mandatos. Estas comillas no se deben especificar para nombres que aparecen como campos en estructuras o como parámetros en llamadas.

ObjectName se utiliza para formar el nombre de tema completo.

El nombre completo del tema se puede crear a partir de dos campos diferentes: *ObjectName* y *ObjectString*. Para obtener detalles sobre cómo se utilizan estos dos campos, consulte “Utilización de series de tema” en la página 558.

Si no se puede encontrar el objeto identificado por el campo *ObjectName* , la llamada falla con el código de razón MQRC_UNKNOWN_OBJECT_NAME incluso si hay una serie especificada en *ObjectString*.

Al volver de una llamada MQSUB utilizando la opción MQSO_RESUME, este campo no cambia.

La longitud de este campo la proporciona MQ_TOPIC_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, el nombre del objeto de tema al que se ha suscrito no se puede cambiar. Este campo y el campo *ObjectString* se pueden omitir. Si se proporcionan, deben resolverse en el mismo nombre de tema completo. Si no lo hacen, la llamada falla con MQRC_TOPIC_NOT_ALTERABLE.

ObjectString (MQCHARV)

Es el nombre de objeto largo que se va a utilizar.

ObjectString se utiliza para formar el nombre de tema completo.

El nombre completo del tema se puede crear a partir de dos campos diferentes: *ObjectName* y *ObjectString*. Para obtener detalles sobre cómo se utilizan estos dos campos, consulte "Utilización de series de tema" en la página 558.

La longitud máxima de *ObjectString* es 10240.

Si *ObjectString* no se especifica correctamente, según la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón MQRC_OBJECT_STRING_ERROR.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQCHARV.

Si hay comodines en *ObjectString* , la interpretación de dichos comodines se puede controlar utilizando las opciones de comodín especificadas en el campo Opciones de MQSD.

Al volver de una llamada MQSUB utilizando la opción MQSO_RESUME, este campo no cambia. El nombre de tema completo utilizado se devuelve en el campo *ResObjectString* si se proporciona un almacenamiento intermedio.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, el nombre largo del objeto de tema al que se ha suscrito no se puede cambiar. Este campo y el campo *ObjectName* se pueden omitir. Si se proporcionan, deben resolverse en el mismo nombre de tema completo o la llamada falla con MQRC_TOPIC_NOT_ALTERABLE.

Opciones (MQLONG)
Esto proporciona opciones para controlar la acción de la llamada MQSUB.

Debe especificar al menos una de las opciones siguientes:

- MQSO_ALTER
- MQSO_RESUME
- MQSO_CREATE

Los valores que especifique para las opciones se pueden utilizar de las maneras siguientes:

- Se pueden añadir los valores. No añada la misma constante más de una vez.
- Los valores se pueden combinar utilizando la operación OR a nivel de bit, si el lenguaje de programación da soporte a operaciones a nivel de bit.

Las combinaciones que no son válidas se anotan en este tema; cualquier otra combinación es válida.

Opciones de acceso o creación: las opciones de acceso y creación controlan si se crea una suscripción o si se devuelve o modifica una suscripción existente. Debe especificar al menos una de estas opciones. La tabla muestra combinaciones válidas de opciones de acceso y creación.

Combinación de opciones	Notas
MQSO_CREATE	Crea una suscripción si no existe. Esta combinación falla si existe la suscripción.
MQSO_RESUME	Reanuda una suscripción existente. Esta combinación falla si no existe ninguna suscripción.
MQSO_CREATE + MQSO_RESUME	Crea una suscripción si no existe una y reanuda una que coincida, si existe. Esta combinación es útil cuando se utiliza en una aplicación que se ejecuta varias veces.
MQSO_ALTER (consulte la nota)	Reanuda una suscripción existente, alterando los campos para que coincidan con los especificados en MQSD. Esta combinación falla si no existe ninguna suscripción.
MQSO_CREATE + MQSO_ALTER (consulte la nota)	Crea una suscripción si no existe una y reanuda una coincidente, si existe, alterando los campos para que coincidan con los especificados en el MQSD. Esta combinación es útil cuando se utiliza en una aplicación que desea asegurarse de que su suscripción está en un estado determinado antes de continuar.
Nota: Las opciones que especifican MQSO_ALTER también pueden especificar MQSO_RESUME, pero esta combinación no tiene ningún efecto adicional para especificar solo MQSO_ALTER. MQSO_ALTER implica MQSO_RESUME, porque llamar a MQSUB para modificar una suscripción implica que la suscripción también se reanudará. Sin embargo, lo contrario no es cierto: reanudar una suscripción no implica que deba ser alterada.	

MQSO_CREATE

Cree una nueva suscripción para el tema especificado. Si existe una suscripción que utiliza el mismo *SubName* , la llamada falla con MQRC_SUB_ALREADY_EXISTS. Esta anomalía se puede evitar combinando la opción MQSO_CREATE con MQSO_RESUME. *SubName* no siempre es necesario. Para obtener más detalles, consulte la descripción de ese campo.

La combinación de MQSO_CREATE con MQSO_RESUME devuelve un manejador a una suscripción preexistente para el *SubName* especificado si se encuentra una; si no hay ninguna suscripción existente, se crea una nueva utilizando todos los campos proporcionados en el MQSD.

MQSO_CREATE también se puede combinar con MQSO_ALTER con un efecto similar.

MQSO_RESUME

Devuelve un descriptor de contexto a una suscripción preexistente que coincide con la especificada por *SubName*. No se realizan cambios en los atributos de suscripciones coincidentes y se devuelven en la salida de la estructura MQSD. Sólo se utilizan los siguientes campos MQSD: StrucId, Version, Options, AlternateUserId y AlternateSecurityId y SubName.

La llamada falla con el código de razón MQRC_NO_SUBSCRIPTION si no existe una suscripción que coincida con el nombre de suscripción completo. Esta anomalía se puede evitar combinando la opción MQSO_CREATE con MQSO_RESUME.

El ID de usuario de la suscripción es el ID de usuario que ha creado la suscripción, o si posteriormente ha sido alterado por un ID de usuario diferente, es el ID de usuario de la modificación correcta más reciente. Si se utiliza un ID AlternateUsery se permite el uso de ID de usuario alternativo para dicho usuario, el ID de usuario alternativo se registra como el ID de usuario que ha creado la suscripción en lugar del ID de usuario bajo el que se ha realizado la suscripción.

Si existe una suscripción coincidente que se ha creado sin la opción MQSO_ANY_USERID, y el ID de usuario de la suscripción es distinto del de la aplicación que solicita un descriptor de contexto para la suscripción, la llamada falla con el código de razón MQRC_IDENTITY_MISMATCH.

Si existe una suscripción coincidente y se está utilizando actualmente, la llamada falla con MQRC_SUBSCRIPTION_IN_USE.

Si la suscripción especificada en SubName no es una suscripción válida para reanudar o modificar desde una aplicación, la llamada falla con MQRC_INVALID_SUBSCRIPTION.

MQSO_RESUME está implícito en MQSO_ALTER por lo que no es necesario combinarlo con esa opción. Sin embargo, la combinación de las dos opciones no provoca un error.

MQSO_ALTER

Devuelve un descriptor de contexto a una suscripción preexistente con el nombre de suscripción completo que coincide con el especificado por el nombre en *SubName*. Los atributos de la suscripción que son diferentes de los especificados en el MQSD se modifican en la suscripción a menos que no se permita la modificación para dicho atributo. Los detalles se anotan en la descripción de cada atributo y se resumen en la tabla siguiente. Si intenta modificar un atributo que no se puede cambiar, o modificar una suscripción que ha establecido la opción MQSO_IMMUTABLE, la llamada falla con el código de razón que se muestra en la tabla siguiente.

La llamada falla con el código de razón MQRC_NO_SUBSCRIPTION si no existe una suscripción que coincida con el nombre de suscripción completo. Puede evitar esta anomalía combinando la opción MQSO_CREATE con MQSO_ALTER.

La combinación de MQSO_CREATE con MQSO_ALTER devuelve un manejador a una suscripción preexistente para el *SubName* especificado si se encuentra una; si no hay ninguna suscripción existente, se crea una nueva utilizando todos los campos proporcionados en el MQSD.

El ID de usuario de la suscripción es el ID de usuario que ha creado la suscripción o, si posteriormente se modifica mediante un ID de usuario diferente, es el ID de usuario de la modificación más reciente y satisfactoria. Si se utiliza un ID AlternateUsery se permite el uso de ID de usuario alternativo para ese usuario, el ID de usuario alternativo se registra como el ID de usuario que ha creado la suscripción en lugar del ID de usuario bajo el que se ha realizado la suscripción.

Si existe una suscripción coincidente que se ha creado sin la opción MQSO_ANY_USERID y el ID de usuario de la suscripción es diferente del de la aplicación que solicita un descriptor de contexto para la suscripción, la llamada falla con el código de razón MQRC_IDENTITY_MISMATCH.

Si existe una suscripción coincidente y se está utilizando actualmente, la llamada falla con MQRC_SUBSCRIPTION_IN_USE.

Si la suscripción especificada en SubName no es una suscripción válida para reanudar o modificar desde una aplicación, la llamada falla con MQRC_INVALID_SUBSCRIPTION.

La tabla siguiente muestra la capacidad de MQSO_ALTER para modificar los valores de atributo en MQSD y MQSUB.

Tabla 547. Atributos en MQSD y MQSUB que se pueden modificar

Descriptor de tipo de datos o llamada de función	Nombre de campo	¿Se puede modificar este atributo utilizando MQSO_ALTER	Código de razón
MQSD	Opciones de durabilidad	No	MQRC_DURABILITY_NOT_ALTERABLE
MQSD	Opciones de destino	Sí	Ninguna
MQSD	Opciones de registro	Sí (consulte la nota “1” en la página 548)	MQRC_GROUPING_NOT_ALTERABLE si intenta modificar MQSO_GROUP_SUB
MQSD	Opciones de publicación	Sí (consulte la nota “2” en la página 548)	Ninguna
MQSD	Opciones de comodín	No	MQRC_TOPIC_NOT_ALTERABLE
MQSD	Otras opciones	No (consulte la nota “3” en la página 548)	Ninguna

Tabla 547. Atributos en MQSD y MQSUB que se pueden modificar (continuación)

Descriptor de tipo de datos o llamada de función	Nombre de campo	¿Se puede modificar este atributo utilizando MQSO_ALTER	Código de razón
MQSD	ObjectName	No	MQRC_TOPIC_NOT_ALTERABLE
MQSD	AlternateUserId	No (consulte la nota "4" en la página 548)	Ninguna
MQSD	AlternateSecurityId	No (consulte la nota "4" en la página 548)	Ninguna
MQSD	SubExpiry	Sí	Ninguna
MQSD	ObjectString	No	MQRC_TOPIC_NOT_ALTERABLE
MQSD	SubName	No (consulte la nota "5" en la página 548)	Ninguna
MQSD	SubUserData	Sí	Ninguna
MQSD	SubCorrelId	Sí (consulte la nota "6" en la página 548)	MQRC_GROUPING_NOT_ALTERABLE cuando está en una suscripción agrupada
MQSD	PubPriority	Sí	Ninguna
MQSD	PubAccountingToken	Sí	Ninguna
MQSD	PubApplIdentityData	Sí	Ninguna
MQSD	SubLevel	No	MQRC_SUBLEVEL_NOT_ALTERABLE
MQSUB	Hobj	Sí (consulte la nota "6" en la página 548)	MQRC_GROUPING_NOT_ALTERABLE cuando está en una suscripción agrupada

Notas:

1. No se puede modificar MQSO_GROUP_SUB.
2. MQSO_NEW_PUBLICATIONS_ONLY no se puede modificar porque no forma parte de la suscripción
3. Estas opciones no forman parte de la suscripción
4. Este atributo no forma parte de la suscripción
5. Este atributo es la identidad de la suscripción que se está alterando
6. Modificable excepto cuando forma parte de un subgrupo agrupado (MQSO_GROUP_SUB)

Opciones de durabilidad: Las opciones siguientes controlan la durabilidad de la suscripción. Sólo puede especificar una de estas opciones. Si está alterando una suscripción existente utilizando la opción MQSO_ALTER, no puede cambiar la durabilidad de la suscripción. Al volver de una llamada MQSUB utilizando MQSO_RESUME, se establece la opción de durabilidad adecuada.

MQSO_DURABLE
Solicite que la suscripción a este tema permanezca hasta que se elimine explícitamente utilizando MQCLOSE con la opción MQCO_REMOVE_SUB. Si esta suscripción no se elimina explícitamente, permanecerá incluso después de que se cierre esta conexión de aplicaciones con el gestor de colas.

Si se solicita una suscripción duradera a un tema que está definido como que no permite suscripciones duraderas, la llamada falla con MQRC_DURABILITY_NOT_ALLOWED.

MQSO_NON_DURABLE
Solicite que la suscripción a este tema se elimine cuando se cierre la conexión de las aplicaciones con el gestor de colas, si todavía no se ha eliminado explícitamente. MQSO_NON_DURABLE es lo contrario de la opción MQSO_DURABLE y se define para ayudar a la documentación del programa. Es el valor predeterminado si no se especifica ninguno.

Opciones de destino: La siguiente opción controla el destino al que se envían las publicaciones de un tema al que se ha suscrito. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar el destino utilizado para las publicaciones de la suscripción. Al volver de una llamada MQSUB utilizando MQSO_RESUME, esta opción se establece si procede.

MQSO_GESTIONADO

Solicite que el gestor de colas gestione el destino al que se envían las publicaciones.

El descriptor de objeto devuelto en *Hobj* representa una cola gestionada del gestor de colas y se utiliza con llamadas MQGET, MQCB, MQINQ o MQCLOSE posteriores.

No se puede proporcionar un descriptor de contexto de objeto devuelto desde una llamada MQSUB anterior en el parámetro *Hobj* cuando no se especifica MQSO_MANAGED.

MQSO_NO_MULTICAST

Solicite que el destino al que se envían las publicaciones no sea una dirección de grupo de multidifusión. Esta opción sólo es válida cuando se combina con la opción MQSO_MANAGED. Cuando se proporciona un descriptor de contexto para una cola en el parámetro *Hobj* , no se puede utilizar la multidifusión para esta suscripción y la opción no es válida.

Si el tema está definido para permitir sólo suscripciones de multidifusión, utilizando el valor `MCAST(ONLY)` , la llamada falla con el código de razón MQRC_MULTICAST_REQUIRED.

Opción de ámbito: La opción siguiente controla el ámbito de la suscripción que se está realizando. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, esta opción de ámbito de suscripción no se puede cambiar. Al volver de una llamada MQSUB utilizando MQSO-RESUME, se establece la opción de ámbito adecuada.

MQSO_SCOPE_QMGR

Esta suscripción sólo se realiza en el gestor de colas local. No se distribuye ninguna suscripción de proxy a otros gestores de colas de la red. Sólo las publicaciones que se publican en este gestor de colas se envían a este suscriptor. Esto altera temporalmente cualquier comportamiento establecido utilizando el atributo de tema SUBSCOPE.

Nota: Si no se establece, el ámbito de suscripción lo determina el atributo de tema SUBSCOPE.

Opciones de registro: Las opciones siguientes controlan los detalles del registro que se realiza en el gestor de colas para esta suscripción. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, estas opciones de registro se pueden cambiar. Al volver de una llamada MQSUB utilizando MQSO_RESUME, se establecen las opciones de registro adecuadas.

SUB_GRUPO_MQSO

Esta suscripción se va a agrupar con otras suscripciones del mismo SubLevel utilizando la misma cola y especificando el mismo ID de correlación para que cualquier publicación a temas que provoque que se proporcione más de un mensaje de publicación al grupo de suscripciones, debido a que se está utilizando un conjunto solapado de series de tema, sólo hace que se entregue un mensaje a la cola. Si no se utiliza esta opción, cada suscripción exclusiva (identificada mediante SubName) que coincida se proporciona con una copia de la publicación, lo que podría significar que más de una copia de la publicación se puede colocar en la cola compartida por un número de suscripciones.

Sólo la suscripción más significativa del grupo se proporciona con una copia de la publicación. La suscripción más significativa se basa en el nombre de tema completo hasta el punto en el que se encuentra un comodín. Si se utiliza una mezcla de esquemas de comodín dentro del grupo, sólo es importante la posición del comodín. Se recomienda no combinar diferentes esquemas de comodín dentro de un grupo de suscripciones que comparten la misma cola.

Al crear una nueva suscripción agrupada, todavía debe tener un SubNameexclusivo, pero si coincide con el nombre de tema completo de una suscripción existente en el grupo, la llamada falla con MQRC_DUPLICATE_GROUP_SUB.

Si la suscripción más significativa del grupo también especifica MQSO_NOT_OWN_PUBS y se trata de una publicación de la misma aplicación, no se entrega ninguna publicación a la cola.

Al modificar una suscripción realizada con esta opción, los campos que implican la agrupación, Hobj en la llamada MQSUB (que representa la cola y el nombre del gestor de colas) y el ID SubCorrelno se pueden cambiar. El intento de modificarlos hace que la llamada falle con MQRC_GROUPING_NOT_ALTERABLE.

Esta opción se debe combinar con MQSO_SET_CORREL_ID con un ID de SubCorrelque no esté establecido en MQCI_NONE y no se pueda combinar con MQSO_MANAGED.

MQSO_ANY_USERID

Cuando se especifica MQSO_ANY_USERID, la identidad del suscriptor no está restringida a un ID de usuario único. Esto permite que cualquier usuario modifique o reanude la suscripción cuando disponga de la autoridad adecuada. Sólo puede tener la suscripción un único usuario a la vez. Un intento de reanudar el uso de una suscripción actualmente en uso por otra aplicación hace que la llamada falle con MQRC_SUBSCRIPTION_IN_USE.

Para añadir esta opción a una suscripción existente, la llamada MQSUB (utilizando MQSO_ALTER) debe proceder del mismo ID de usuario que la propia suscripción original.

Si una llamada MQSUB hace referencia a una suscripción existente con MSQO_ANY_USERID establecido y el ID de usuario difiere de la suscripción original, la llamada sólo será satisfactoria si el nuevo ID de usuario tiene autorización para suscribirse al tema. Al finalizar correctamente, las futuras publicaciones de este suscriptor se colocan en la cola de suscriptores con el nuevo ID de usuario establecido en el mensaje de publicación.

No especifique MQSO_ANY_USERID y MQSO_FIXED_USERID. Si no se especifica ninguno, el valor predeterminado es MQSO_FIXED_USERID.

MQSO_FIXED_USERID

Cuando se especifica MQSO_FIXED_USERID, sólo el último ID de usuario que modifica la suscripción puede modificar o reanudar la suscripción. Si la suscripción no se ha modificado, es el ID de usuario que ha creado la suscripción.

Si un verbo MQSUB hace referencia a una suscripción existente con MQSO_ANY_USERID establecido y altera la suscripción utilizando MQSO_ALTER para utilizar la opción MQSO_FIXED_USERID, el ID de usuario de la suscripción se fija ahora en este nuevo ID de usuario. La llamada sólo es satisfactoria si el nuevo ID de usuario tiene autoridad para suscribirse al tema.

Si un ID de usuario distinto del registrado como propietario de una suscripción intenta reanudar o modificar una suscripción MQSO_FIXED_USERID, la llamada falla con MQRC_IDENTITY_MISMATCH. El ID de usuario propietario de una suscripción se puede ver mediante el mandato DISPLAY SBSTATUS.

No especifique MQSO_ANY_USERID y MQSO_FIXED_USERID. Si no se especifica ninguno, el valor predeterminado es MQSO_FIXED_USERID.

Opciones de publicación: Las opciones siguientes controlan la forma en que se envían las publicaciones a este suscriptor. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, estas opciones de publicación se pueden cambiar.

MQSO_NOT_OWN_PUBS
Indica al intermediario que la aplicación no desea ver ninguna de sus propias publicaciones. Se considera que las publicaciones se originan en la misma aplicación si los manejadores de conexión son los mismos. Al volver de una llamada MQSUB utilizando MQSO_RESUME, esta opción se establece si procede.

MQSO_NEW_PUBLICATIONS_ONLY
No se enviarán publicaciones retenidas actualmente, cuando se cree esta suscripción, sólo se enviarán nuevas publicaciones. Esta opción sólo se aplica cuando se especifica MQSO_CREATE. Cualquier cambio posterior en una suscripción no altera el flujo de publicaciones y, por lo tanto, las publicaciones retenidas en un tema, ya se habrán enviado al suscriptor como nuevas publicaciones.

Si se especifica esta opción sin MQSO_CREATE, la llamada falla con MQRC_OPTIONS_ERROR. Al volver de una llamada MQSUB utilizando MQSO_RESUME, esta opción no se establece incluso si la suscripción se ha creado utilizando esta opción.

Si no se utiliza esta opción, los mensajes retenidos anteriormente se envían a la cola de destino proporcionada. Si esta acción falla debido a un error, ya sea MQRC_RETAINED_MSG_Q_ERROR o MQRC_RETAINED_NOT_DELIVERED, la creación de la suscripción falla.

MQSO_PUBLICATIONS_ON_REQUEST
El establecimiento de esta opción indica que el suscriptor solicitará información específicamente cuando sea necesario. El gestor de colas no envía mensajes no solicitados al suscriptor. La publicación

retenida (o posiblemente varias publicaciones si se especifica un comodín en el tema) se envía al suscriptor cada vez que se realiza una llamada MQSUBRQ utilizando el descriptor de contexto Hsub de una llamada MQSUB anterior. No se envían publicaciones como resultado de la llamada MQSUB utilizando esta opción. Al volver de una llamada MQSUB utilizando MQSO_RESUME, esta opción se establece si procede.

Esta opción no es válida en combinación con un SubLevel mayor que 1.

Opciones de lectura anticipada: Las opciones siguientes controlan si los mensajes no persistentes se envían a una aplicación antes de que la aplicación los solicite.

MQSO_READ_AHEAD_AS_Q_DEF
Si la llamada MQSUB utiliza un descriptor de contexto gestionado, el atributo de lectura anticipada predeterminado de la cola de modelo asociada con el tema al que se ha suscrito determina si los mensajes se envían a la aplicación antes de que la aplicación los solicite.

Este es el valor predeterminado.

MQSO_NO_READ_AHEAD
Si la llamada MQSUB utiliza un descriptor de contexto gestionado, los mensajes no se envían a la aplicación antes de que la aplicación los solicite.

MQSO_READ_AHEAD
Si la llamada MQSUB utiliza un descriptor de contexto gestionado, es posible que se envíen mensajes a la aplicación antes de que la aplicación los solicite.

Nota:

Las siguientes notas se aplican a las opciones de lectura anticipada:

1. Sólo se puede especificar una de estas opciones. Si se especifican MQOO_READ_AHEAD y MQOO_NO_READ_AHEAD, se devuelve el código de razón MQRC_OPTIONS_ERROR. Estas opciones sólo son aplicables si se especifica MQSO_MANAGED.
2. No son aplicables para MQSUB cuando se pasa una cola que se ha abierto anteriormente. Es posible que la lectura anticipada no esté habilitada cuando se solicite. Las opciones MQGET utilizadas en la primera llamada MQGET pueden impedir que se habilite la lectura anticipada. Además, la lectura anticipada está inhabilitada cuando el cliente se está conectando a un gestor de colas donde la lectura anticipada no está soportada. Si la aplicación no se ejecuta como un cliente WebSphere MQ , estas opciones se ignoran.

Opciones de comodín: Las opciones siguientes controlan cómo se interpretan los comodines en la serie proporcionada en el campo ObjectString de MQSD. Sólo puede especificar una de estas opciones. Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, estas opciones de comodín no se pueden cambiar. Al volver de una llamada MQSUB utilizando MQSO_RESUME, se establece la opción de comodín adecuada.

MQSO_WILDCARD_CHAR
Los comodines sólo operan en caracteres dentro de la serie de tema.

El comportamiento definido por MQSO_WILDCARD_CHAR se muestra en la tabla siguiente.

Carácter especial	**Comportamiento**
Barra inclinada (/)	Sin significación, sólo otro carácter
Los caracteres de asterisco (*)	Comodín, cero o más caracteres
Un signo de interrogación (?)	Comodín, 1 carácter
Signo de porcentaje (%)	Carácter de escape para permitir que los caracteres (*), (?) o (%) se utilicen en una serie y no se interpreten como un carácter especial, por ejemplo, (% *), (%?) o (%%).

Por ejemplo, la publicación en el tema siguiente:

```
/level0/level1/level2/level3/level4
```

coincide con los suscriptores utilizando los temas siguientes:

```
*
/*
/ level0/level1/level2/level3/*
/ level0/level1/*/level3/level4
/ level0/level1/le?el2/level3/level4
```

Nota: Este uso de comodines proporciona exactamente el significado proporcionado en WebSphere MQ V6 y WebSphere MB V6 cuando se utilizan mensajes con formato MQRFH1 para publicación/suscripción. Se recomienda que no se utilice para las aplicaciones recién escritas y que solo se utilice para las aplicaciones que se ejecutaban anteriormente en esa versión y que no se han modificado para utilizar el comportamiento de comodín predeterminado tal como se describe en MQSO_WILDCARD_TOPIC.

MQSO_WILDCARD_TOPIC

Los comodines sólo operan en elementos de tema dentro de la serie de tema. Este es el comportamiento predeterminado si no se elige ninguno.

El comportamiento necesario para MQSO_WILDCARD_TOPIC se muestra en la tabla siguiente:

Carácter especial	**Comportamiento**
(/)	Separador de nivel de tema
signo de almohadilla (#)	Comodín: nivel de varios temas
Signo más (+)	Comodín: nivel de tema único

Notas:

Los caracteres (+) y (#) no se tratan como comodines si se mezclan con otros caracteres (incluidos ellos mismos) dentro de un nivel de tema. En la serie siguiente, los caracteres (#) y (+) se tratan como caracteres ordinarios.

```
level0/level1/#+/level3/level#
```

Por ejemplo, la publicación en el tema siguiente:

```
/level0/level1/level2/level3/level4
```

coincide con los suscriptores utilizando los temas siguientes:

```
#
/#
/ level0/level1/level2/level3/#
/ level0/level1/+/level3/level4
```

Nota: Este uso de comodines proporciona el significado que se proporciona en WebSphere Message Broker Versión 6 cuando se utilizan mensajes con formato MQRFH2 para la publicación/suscripción.

Otras opciones: Las opciones siguientes controlan la forma en que se emite la llamada de API en lugar de la suscripción. Al volver de una llamada MQSUB utilizando MQSO_RESUME, estas opciones no se modifican. Consulte “AlternateUserId (MQCHAR12)” en la página 543 para obtener más detalles.

MQSO_ALTERNATE_USER_AUTHORITY

El campo AlternateUserId contiene un identificador de usuario para utilizarlo para validar esta llamada MQSUB. La llamada sólo puede realizarse correctamente si este ID AlternateUserestá autorizado para abrir el objeto con las opciones de acceso especificadas, independientemente de si el identificador de usuario bajo el que se ejecuta la aplicación está autorizado para hacerlo.

MQSO_SET_CORREL_ID

La suscripción debe utilizar el identificador de correlación proporcionado en el campo *SubCorrelId* . Si no se especifica esta opción, el gestor de colas crea automáticamente un identificador de correlación en el momento de la suscripción y se devuelve a la aplicación en el campo *SubCorrelId* . Para obtener más información, consulte "SubCorrelId (MQBYTE24)" en la página 555 .

Esta opción no se puede combinar con MQSO_MANAGED.

MQSO_SET_IDENTITY_CONTEXT

La suscripción es para utilizar la señal de contabilidad y los datos de identidad de aplicación proporcionados en los campos *PubAccountingToken* y *PubApplIdentityData* .

Si se especifica esta opción, se realiza la misma comprobación de autorización que si se accediera a la cola de destino mediante una llamada MQOPEN con MQOO_SET_IDENTITY_CONTEXT, excepto en el caso en que se utilice también la opción MQSO_MANAGED, en cuyo caso no hay comprobación de autorización en la cola de destino.

Si no se especifica esta opción, las publicaciones enviadas a este suscriptor tienen información de contexto predeterminada asociada a ellos, de la manera siguiente:

Campo de MQMD	Valor utilizado
UserIdentifier	El ID de usuario asociado a la suscripción en el momento en que se realizó la suscripción.
AccountingToken	Se determina a partir del entorno, si es posible; se establece en MQACT_NONE si no es así.
ApplIdentityData	Establecer en blancos

Esta opción sólo es válida con MQSO_CREATE y MQSO_ALTER. Si se utiliza con MQSO_RESUME, los campos *PubAccountingToken* y *PubApplIdentityData* se ignoran, por lo que esta opción no tiene ningún efecto.

Si se altera una suscripción sin utilizar esta opción donde anteriormente la información de contexto de identidad proporcionada por la suscripción, se genera la información de contexto predeterminada para la suscripción alterada.

Si una suscripción que permite que distintos ID de usuario la utilicen con la opción MQSO_ANY_USERID, se reanuda con un ID de usuario diferente, se genera contexto de identidad predeterminado para el nuevo ID de usuario que es propietario ahora de la suscripción y las publicaciones posteriores se entregan conteniendo el nuevo contexto de identidad.

MQSO_FAIL_IF_QUIESCING

La llamada MQSUB falla si el gestor de colas está en estado de desactivación temporal. En z/OS, para una aplicación CICS o IMS , esta opción también fuerza que la llamada MQSUB falle si la conexión está en estado de desactivación temporal.

Señal PubAccounting(MQBYTE32)

Este es el valor que estará en el campo *AccountingToken* del Descriptor de mensaje (MQMD) de todos los mensajes de publicación que coincidan con esta suscripción. *AccountingToken* forma parte del contexto de identidad del mensaje. Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje. Para obtener más información sobre el campo *AccountingToken* en MQMD, consulte "AccountingToken (MQBYTE32)" en la página 396

Puede utilizar el siguiente valor especial para el campo *PubAccountingToken* :

MQACT_NONE

No se ha especificado ninguna señal de contabilidad.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, la constante MQACT_NONE_ARRAY también está definida; tiene el mismo valor que MQACT_NONE, pero es una matriz de caracteres en lugar de una serie.

Si no se especifica la opción MQSO_SET_IDENTITY_CONTEXT, el gestor de colas genera la señal de contabilidad como información de contexto predeterminada y este campo es un campo de salida que contiene el *AccountingToken* que se establecerá en cada mensaje publicado para esta suscripción.

Si se especifica la opción MQSO_SET_IDENTITY_CONTEXT, el usuario está generando la señal de contabilidad y este campo es un campo de entrada que contiene el *AccountingToken* que se debe establecer en cada publicación para esta suscripción.

La longitud de este campo la proporciona MQ_ACCOUNTING_TOKEN_LENGTH. El valor inicial de este campo es MQACT_NONE.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar el valor de *AccountingToken* en cualquier mensaje de publicación futuro.

Al volver de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en el *AccountingToken* actual que se utiliza para la suscripción.

PubApplIdentityData (MQCHAR32)

Es el valor que se encuentra en el campo *ApplIdentityData* del Descriptor de mensaje (MQMD) de todos los mensajes de publicación que coinciden con esta suscripción. *ApplIdentityData* forma parte del contexto de identidad del mensaje. Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje. Para obtener más información sobre el campo *ApplIdentityData* en MQMD, consulte "Datos de ApplIdentity(MQCHAR32)" en la página 398

Si no se especifica la opción MQSO_SET_IDENTITY_CONTEXT, el *ApplIdentityData* que se establece en cada mensaje publicado para esta suscripción está en blanco, como información de contexto predeterminada.

Si se especifica la opción MQSO_SET_IDENTITY_CONTEXT, el usuario está generando *PubApplIdentityData* y este campo es un campo de entrada que contiene el *ApplIdentityData* que se debe establecer en cada publicación para esta suscripción.

La longitud de este campo la proporciona MQ_APPL_IDENTITY_DATA_LENGTH. El valor inicial de este campo es la serie nula en C y 32 caracteres en blanco en otros lenguajes de programación.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar el *ApplIdentityData* de cualquier mensaje de publicación futuro.

Al volver de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en el *ApplIdentityData* actual que se utiliza para la suscripción.

PubPriority (MQLONG)

Este es el valor que estará en el campo *Priority* del Descriptor de mensaje (MQMD) de todos los mensajes de publicación que coincidan con esta suscripción. Para obtener más información sobre el campo *Priority* en MQMD, consulte "Priority (MQLONG)" en la página 422.

El valor debe ser mayor o igual que cero, donde cero es la prioridad más baja. También se pueden utilizar los valores especiales siguientes:

MQPRI_PRIORITY_AS_Q_DEF

Cuando se proporciona una cola de suscripción en el campo *Hobj* de la llamada MQSUB, y no es un descriptor de contexto gestionado, la prioridad del mensaje se toma del atributo *DefPriority* de esta cola. Si la cola es una cola de clúster o hay más de una definición en la vía de acceso de resolución de nombre de cola, la prioridad se determina cuando el mensaje de publicación se coloca en la cola tal como se describe para "Priority (MQLONG)" en la página 422.

Si la llamada MQSUB utiliza un descriptor de contexto gestionado, la prioridad del mensaje se toma del atributo *DefPriority* de la cola modelo asociada al tema al que se ha suscrito.

MQPRI_PRIORITY_AS_PUBLISHED

La prioridad del mensaje es la prioridad de la publicación original. Es el valor inicial del campo.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar el *Priority* de cualquier mensaje de publicación futuro.

Al volver de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en la prioridad actual que se utiliza para la suscripción.

Serie ResObject(MQCHARV)
Es el nombre de objeto largo después de que el gestor de colas resuelva el nombre proporcionado en *ObjectName*.

Si el nombre de objeto largo se proporciona en *ObjectString* y no se proporciona nada en *ObjectName*, el valor devuelto en este campo es el mismo que el proporcionado en *ObjectString*.

Si este campo se omite (es decir, ResObjectString.VSBufSize es cero), no se devuelve *ResObjectString* , pero la longitud se devuelve en ResObjectString.VSLength. Si la longitud es menor que la serie ResObjectcompleta, se trunca y devuelve tantos caracteres situados más a la derecha como caben en la longitud proporcionada.

Si *ResObjectString* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón MQRC_RES_OBJECT_STRING_ERROR.

SelectionString (MQCHARV)
Es la serie utilizada para proporcionar los criterios de selección utilizados al suscribirse a mensajes de un tema.

Este campo de longitud variable se devolverá en la salida de una llamada MQSUB utilizando la opción MQSO_RESUME, si se proporciona un almacenamiento intermedio, y también hay una longitud de almacenamiento intermedio positiva en VSBufSize. Si no se proporciona ningún almacenamiento intermedio en la llamada, sólo se devolverá la longitud de la serie de selección en el campo VSLength de MQCHARV. Si el búfer proporcionado es inferior al espacio necesario para devolver el campo, solo se devuelven VSBufSize bytes en el búfer.

Si *SelectionString* se especifica incorrectamente, según la descripción de cómo utilizar la estructura “MQCHARV-Serie de longitud variable” en la página 275 , o si supera la longitud máxima, la llamada falla con el código de razón MQRC_SELECTION_STRING_ERROR.

El uso de SelectionString se describe en Selectores .

StrucId (MQCHAR4)

Este es el identificador de estructura; el valor debe ser:

MQSD_STRUC_ID
Identificador de la estructura del descriptor de suscripción.

Para el lenguaje de programación C, también se define la constante MQSD_STRUC_ID_ARRAY; tiene el mismo valor que MQSD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQSD_STRUC_ID.

SubCorrelId (MQBYTE24)
Este campo contiene un identificador de correlación común a todas las publicaciones que coinciden con esta suscripción.

Atención: un identificador de correlación sólo se puede pasar entre gestores de colas en un clúster de publicación/suscripción, no en una jerarquía.

Todas las publicaciones enviadas para que coincidan con esta suscripción contienen este identificador de correlación en el descriptor de mensaje. Si varias suscripciones obtienen sus publicaciones de la misma cola, el uso de MQGET por identificador de correlación sólo permite obtener publicaciones para una suscripción específica. Este identificador de correlación puede ser generado por el gestor de colas o por el usuario.

Si no se especifica la opción MQSO_SET_CORREL_ID, el gestor de colas genera el identificador de correlación y este campo es un campo de salida que contiene el identificador de correlación que se establecerá en cada mensaje publicado para esta suscripción. El identificador de correlación generado consta de un identificador de producto de 4 bytes (AMQX o CSQM en ASCII o EBCDIC) seguido de una implementación específica de producto de una serie exclusiva.

Si se especifica la opción MQSO_SET_CORREL_ID, el usuario genera el identificador de correlación y este campo es un campo de entrada que contiene el identificador de correlación que se debe establecer en cada publicación para esta suscripción. En este caso, si el campo contiene MQCI_NONE, el identificador de correlación que se establece en cada mensaje publicado para esta suscripción es el identificador de correlación creado por la colocación original del mensaje.

Si la opción MQSO_GROUP_SUB se ha especificado y el identificador de correlación especificado es el mismo que una suscripción agrupada existente que utiliza la misma cola y una serie de tema que se solapa, sólo la suscripción más significativa del grupo se proporciona con una copia de la publicación.

La longitud de este campo la proporciona MQ_CORREL_ID_LENGTH. El valor inicial de este campo es MQCI_NONE.

Si está alterando una suscripción existente utilizando la opción MQSO_ALTER, y este campo es un campo de entrada, el identificador de correlación de suscripción se puede cambiar, a menos que la suscripción sea una suscripción agrupada, es decir, se ha creado utilizando la opción MQSO_GROUP_SUB, en cuyo caso el identificador de correlación de suscripción no se puede cambiar.

Al volver de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en el identificador de correlación actual para la suscripción.

SubExpiry (MQLONG)

Este es el tiempo expresado en décimas de segundo tras el cual caduca la suscripción. No habrá más publicaciones que coincidan con esta suscripción después de que haya pasado este intervalo. Tan pronto como caduca una suscripción, las publicaciones ya no se envían a la cola. Sin embargo, las publicaciones que ya están allí no se ven afectadas de ninguna manera. *SubExpiry* no tiene ningún efecto en la caducidad de la publicación.

Se reconoce el siguiente valor especial:

MQEI_UNLIMITED
La suscripción tiene un tiempo de caducidad ilimitado.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, se puede cambiar la caducidad de la suscripción.

Al volver de una llamada MQSUB utilizando la opción MQSO_RESUME, este campo se establece en la caducidad original de la suscripción y no en el tiempo de caducidad restante.

SubLevel (MQLONG)
Éste es el nivel asociado a la suscripción. Las publicaciones sólo se entregan a esta suscripción si está en el conjunto de suscripciones con el valor de SubLevel más alto menor o igual que el valor de PubLevel utilizado en el momento de la publicación. Sin embargo, si una publicación se ha retenido, ya no está disponible para los suscriptores en niveles superiores porque se vuelve a publicar en PubLevel 1.

El valor debe estar en el rango de cero a 9. Cero sería el nivel más bajo.

El valor inicial de este campo es 1.

Para obtener más información, consulte Interceptación de publicaciones .

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, el SubLevel no se puede cambiar.

No se permite combinar un SubLevel con un valor mayor que 1 con la opción MQSO_PUBLICATIONS_ON_REQUEST.

Al volver de una llamada MQSUB utilizando MQSO_RESUME, este campo se establece en el nivel actual que se utiliza para la suscripción.

Datos de SubUser(MQCHARV)

Especifica los datos de usuario de suscripción. Los datos proporcionados en la suscripción en este campo se incluirán como la propiedad de mensaje MQSubUserData de cada publicación que se envíe a esta suscripción.

La longitud máxima de *SubUserData* es 10240.

Si *SubUserData* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV , o si supera la longitud máxima, la llamada falla con el código de razón MQRC_SUB_USER_DATA_ERROR.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQCHARV.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, los datos de usuario de suscripción se pueden cambiar.

Este campo de longitud variable se devuelve en la salida de una llamada MQSUB utilizando la opción MQSO_RESUME, si se proporciona un almacenamiento intermedio y hay una longitud de almacenamiento intermedio positiva en *VSBufLen*. Si no se proporciona ningún almacenamiento intermedio en la llamada, sólo se devuelve la longitud de la fecha de usuario de suscripción en el campo *VSLength* de MQCHARV. Si el almacenamiento intermedio proporcionado es menor que el espacio necesario para devolver el campo, sólo se devuelven *VSBufLen* bytes en el almacenamiento intermedio proporcionado.

SubName (MQCHARV)

Especifica el nombre de suscripción. Este campo sólo es necesario si *Options* especifica la opción MQSO_DURABLE, pero si se proporciona también lo utilizará el gestor de colas para MQSO_NON_DURABLE.

Si se especifica, *SubName* debe ser exclusivo dentro del gestor de colas, porque es el método utilizado para identificar la suscripción.

La longitud máxima de *SubName* es 10240.

Este campo sirve para dos propósitos. Para una suscripción MQSO_DURABLE, utilice este campo para identificar una suscripción para poder reanudarla después de que se haya creado si ha cerrado el descriptor de contexto de la suscripción (utilizando la opción MQCO_KEEP_SUB) o se ha desconectado del gestor de colas. Esto se realiza utilizando la llamada MQSUB con la opción MQSO_RESUME. También se visualiza en la vista administrativa de suscripciones en el campo SUBNAME en DISPLAY SBSTATUS.

Si *SubName* se especifica incorrectamente, según la descripción de cómo utilizar la estructura MQCHARV , se deja de lado cuando es necesario (es decir, *SubName*.*VSLength* es cero), o si supera la longitud máxima, la llamada falla con el código de razón MQRC_SUB_NAME_ERROR.

Este es un campo de entrada. Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQCHARV.

Si se modifica una suscripción existente utilizando la opción MQSO_ALTER, el nombre de suscripción no se puede cambiar, porque es el campo de identificación utilizado para encontrar la suscripción referenciada. No se cambia en la salida de una llamada MQSUB con la opción MQSO_RESUME.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQSD_VERSION_1
Version-1 Estructura del descriptor de suscripción.

La constante siguiente especifica el número de versión de la versión actual:

MQSD_CURRENT_VERSION
Versión actual de la estructura del descriptor de suscripción.

Siempre es un campo de entrada. El valor inicial de este campo es MQSD_VERSION_1.

Utilización de series de tema

Un tema se construye a partir del subtema identificado en un objeto de tema y un subtema proporcionado por una aplicación. Puede utilizar subtema como nombre de tema o combinarlos para formar un nuevo nombre de tema.

En un programa MQI, MQOPENcrea el nombre de tema completo. Está compuesto por los campos utilizados en las llamadas de MQI de publicación/suscripción en el orden listado:

1. El atributo **TOPICSTR** del objeto de tema, nombrado en el campo **ObjectName**.
2. El parámetro **ObjectString** que define el subtema que la aplicación suministra.

La serie de tema resultante se devuelve en el parámetro **ResObjectString**.

Se considera que estos campos están presentes si el primer carácter de cada campo no es un valor en blanco o un carácter nulo y la longitud de campo es mayor que cero. Si sólo uno de los campos está presente, se utiliza sin modificaciones como nombre del tema. Si ninguno de los campos tiene un valor, la llamada falla con el código de razón MQRC_UNKNOWN_OBJECT_NAME o MQRC_TOPIC_STRING_ERROR si el nombre de tema completo no es válido.

Si ambos campos están presentes, se inserta un carácter '/' entre los dos elementos del nombre de tema combinado resultante.

En la Tabla 548 en la página 558, se muestran ejemplos de concatenación de series de tema:

Tabla 548. Ejemplo de concatenación de series de tema

TOPICSTR	ObjectString	Nombre de tema completo	Comentario
Fútbol/Resultados	' '	Fútbol/Resultados	El TOPICSTR se utiliza solo
' '	Fútbol/Resultados	Fútbol/Resultados	ObjectString se utiliza solo
Fútbol	Resultados	Fútbol/Resultados	Se añade un carácter ' /' en el punto de concatenación
Fútbol	/Resultados	Fútbol//Resultados	Se produce un 'nodo vacío ' entre las dos series
/Fútbol	Resultados	/Fútbol/Resultados	El tema empieza con un 'nodo vacío '

El carácter '/' se considera un carácter especial, proporcionando estructura al nombre de tema completo en Árboles de temas , y no debe utilizarse por ninguna otra razón, ya que la estructura del árbol de temas se ve afectada. El tema "/Football" no es el mismo que el tema "Football".

Los siguientes caracteres comodín son caracteres especiales:

- signo más '+'

- signo de número '#'
- asterisco '*'
- signo de interrogación '?'

Estos caracteres no se consideran no válidos, sin embargo, debe asegurarse de entender cómo se utilizan. Es posible que prefiera no utilizar estos caracteres en las series de tema al publicar. La publicación en una serie de tema con '#' o '+' mezclado con otros caracteres (incluidos ellos mismos) dentro de un nivel de tema, se puede suscribir con cualquier esquema comodín. La publicación en una serie de tema con '#' o '+' como único carácter entre dos caracteres '/' produce una serie de tema a la que una aplicación no puede suscribirse explícitamente utilizando el esquema de comodín MQSO_WILDCARD_TOPIC. El resultado de esta situación es que la aplicación obtiene más publicaciones de lo previsto.

Fragmento de código de ejemplo

Este fragmento de código, extraído del programa de ejemplo Ejemplo 2: Publicador en un tema variable, combina un objeto de tema con una serie de tema variable.

```
MQOD     td = {MQOD_DEFAULT};  /* Object Descriptor            */
td.ObjectType = MQOT_TOPIC;    /* Object is a topic            */
td.Version = MQOD_VERSION_4;   /* Descriptor needs to be V4    */
strncpy(td.ObjectName, topicName,  MQ_TOPIC_NAME_LENGTH);
td.ObjectString.VSPtr = topicString;
td.ObjectString.VSLength = (MQLONG)strlen(topicString);
td.ResObjectString.VSPtr = resTopicStr;
td.ResObjectString.VSBufSize = sizeof(resTopicStr)-1;
MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
```

Valores iniciales y declaraciones de lenguaje para MQSD

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQSD_STRUC_ID	'SD¬¬'
Version	MQSD_VERSION_1	1
Options	MQSO_NON_DURABLE	0
ObjectName	Ninguna	Serie nula o espacios en blanco
AlternateUserId	Ninguna	Serie nula o espacios en blanco
AlternateSecurityId	MQSID_NONE	Nulos
SubExpiry	MQEI_UNLIMITED	-1
ObjectString	Ninguna	Nombres y valores definidos para MQCHARV
SubName	Ninguna	Nombres y valores definidos para MQCHARV
SubUserData	Ninguna	Nombres y valores definidos para MQCHARV
SubCorrelId	MQCI_NONE	Nulos
PubPriority	MQPRI_PRIORITY_AS_Q_DEF	-3
PubAccountingToken	MQACT_NONE	Nulos
PubApplIdentityData	Ninguna	Serie nula o espacios en blanco
Selection String	Ninguna	Nombres y valores definidos para MQCHARV
SubLevel	Ninguna	1

Nombre de campo	Nombre de constante	Valor de constante
ResObjectString	Ninguna	Nombres y valores definidos para MQCHARV

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQSD_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQSD MySD = {MQSD_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
** 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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

MQSMPO-Establecer opciones de propiedad de mensaje

La tabla siguiente resume los campos de la estructura.

Tabla 549. Campos en MQSMPO		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Options	Opciones	Opciones
ValueEncoding	Codificación de valor de propiedad	ValueEncoding
ValueCCSID	Juego de caracteres de valor de propiedad	ValueCCSID

Visión general de MQSMPO

Disponibilidad: Todos los sistemas WebSphere MQ y clientes WebSphere MQ .

Finalidad: la estructura **MQSMPO** permite a las aplicaciones especificar opciones que controlan cómo se establecen las propiedades de los mensajes. La estructura es un parámetro de entrada en la llamada **MQSETMP** .

Juego de caracteres y codificación: Los datos de **MQSMPO** deben estar en el juego de caracteres de la aplicación y la codificación de la aplicación (**MQENC_NATIVE**).

Campos para MQSMPO

La estructura MQSMPO contiene los campos siguientes; los campos se describen en **orden alfabético**:

Opciones (MQLONG)

Opciones de ubicación: Las opciones siguientes están relacionadas con la ubicación relativa de la propiedad en comparación con el cursor de propiedad:

MQSMPO_SET_FIRST
Establece el valor de la primera propiedad que coincide con el nombre especificado, o si no existe, añade una nueva propiedad después de todas las demás propiedades con una jerarquía coincidente.

MQSMPO_SET_PROP_UNDER_CURSOR
Establece el valor de la propiedad a la que apunta el cursor de propiedad. La propiedad a la que apunta el cursor de propiedad es la que se ha consultado por última vez utilizando la opción MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje en una llamada MQGET, o cuando se especifica el descriptor de mensaje en el campo *MsgHandle* de la estructura MQGMO o MQPMO en una llamada MQPUT.

Si esta opción se utiliza cuando todavía no se ha establecido el cursor de propiedad o si se ha suprimido el puntero de propiedad para el cursor de propiedad, la llamada falla con el código de terminación MQCC_FAILED y el código de razón MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_SET_PROP_BEFORE_CURSOR
Establece una nueva propiedad antes de la propiedad a la que apunta el cursor de propiedad. La propiedad a la que apunta el cursor de propiedad es la que se ha consultado por última vez utilizando la opción MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje en una llamada MQGET, o cuando se especifica el descriptor de mensaje en el campo *MsgHandle* de la estructura MQGMO o MQPMO en una llamada MQPUT.

Si esta opción se utiliza cuando todavía no se ha establecido el cursor de propiedad o si se ha suprimido el puntero de propiedad para el cursor de propiedad, la llamada falla con el código de terminación MQCC_FAILED y el código de razón MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_SET_PROP_AFTER_CURSOR
Establece una nueva propiedad después de la propiedad a la que apunta el cursor de propiedad. La propiedad a la que apunta el cursor de propiedad es la que se ha consultado por última vez utilizando la opción MQIMPO_INQ_FIRST o MQIMPO_INQ_NEXT.

El cursor de propiedad se restablece cuando se reutiliza el descriptor de mensaje en una llamada MQGET, o cuando se especifica el descriptor de mensaje en el campo *MsgHandle* de la estructura MQGMO o MQPMO en una llamada MQPUT.

Si esta opción se utiliza cuando todavía no se ha establecido el cursor de propiedad o si se ha suprimido el puntero de propiedad para el cursor de propiedad, la llamada falla con el código de terminación MQCC_FAILED y el código de razón MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_APPEND_PROPERTY
Hace que se añada una nueva propiedad después de todas las demás propiedades con una jerarquía coincidente. Si existe al menos una propiedad que coincide con el nombre especificado, se añade una nueva propiedad al final después del final de la lista de propiedades.

Esta opción permite crear una lista de propiedades con el mismo nombre.

Si no necesita ninguna de las opciones descritas, utilice la opción siguiente:

MQSMPO_NONE
No se ha especificado ninguna opción.

Siempre es un campo de entrada. El valor inicial de este campo es MQSMPO_SET_FIRST.

StrucId (MQCHAR4)

Este es el identificador de estructura; el valor debe ser:

MQSMPO_STRUC_ID

Identificador para establecer la estructura de opciones de propiedad de mensaje.

Para el lenguaje de programación C, la constante **MQSMPO_STRUC_ID_ARRAY** también está definida; tiene el mismo valor que **MQSMPO_STRUC_ID**, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es **MQSMPO_STRUC_ID**.

ValueCCSID (MQLONG)

El juego de caracteres del valor de propiedad que se va a establecer si el valor es una serie de caracteres.

Siempre es un campo de entrada. El valor inicial de este campo es **MQCCSI_APPL**.

ValueEncoding (MQLONG)

La codificación del valor de propiedad que se va a establecer si el valor es numérico.

Siempre es un campo de entrada. El valor inicial de este campo es **MQENC_NATIVE**.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQSMPO_VERSION_1
Version-1 establece la estructura de opciones de propiedad de mensaje.

La constante siguiente especifica el número de versión de la versión actual:

MQSMPO_VERSIÓN_ACTUAL
Versión actual de la estructura de opciones de establecer propiedad de mensaje.

Siempre es un campo de entrada. El valor inicial de este campo es **MQSMPO_VERSION_1**.

Valores iniciales y declaraciones de lenguaje para MQSMPO

Tabla 550. Valores iniciales de campos en MQSMPO

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQSMPO_STRUC_ID	'SMPO'
Version	MQSMPO_VERSION_1	1
Options	MQSMPO_NONE	0
ValueEncoding	MQENC_NATIVE	Depende del entorno
ValueCCSID	MQCCSI_APPL	-3

Notas:

1. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
2. En el lenguaje de programación C, la variable de macroMQSMPO_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQSMPO MySMPO = {MQSMPO_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

MQSRO-Opciones de solicitud de suscripción

En esta sección se describen las opciones de solicitud de suscripción, los campos que contiene y los valores iniciales de dichos campos.

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
Options	Opciones	Opciones
NumPubs	Número de publicaciones	NumPubs

Visión general de MQSRO

Disponibilidad: AIX, HP-UX, IBM i, Solaris, Linux, Windows, z/OS más WebSphere MQ clientes MQI conectados a estos sistemas.

Finalidad: la estructura MQSRO permite a la aplicación especificar opciones que controlan cómo se realiza una solicitud de suscripción. La estructura es un parámetro de entrada/salida en la llamada MQSUBRQ.

Versión: La versión actual de MQSRO es MQSRO_VERSION_1.

Conjunto de caracteres y codificación: los datos de MQSRO deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE. Sin embargo, si la aplicación se ejecuta como un cliente MQI de MQ , la estructura debe estar en el juego de caracteres y la codificación del cliente.

Campos para MQSRO

La estructura MQSRO contiene los campos siguientes; los campos se describen en orden alfabético:

NumPubs (MQLONG)

Es un campo de salida, devuelto a la aplicación para indicar el número de publicaciones enviadas a la cola de suscripción como resultado de esta llamada. Aunque este número de publicaciones se han enviado como resultado de esta llamada, no hay garantía de que este número de mensajes estén disponibles para que la aplicación los obtenga, especialmente si son mensajes no persistentes.

Puede haber más de una publicación si el tema al que se ha suscrito contenía un comodín. Si no había comodines en la serie de tema cuando se creó la suscripción representada por *Hsub* , como máximo se envía una publicación como resultado de esta llamada.

Opciones (MQLONG)

Debe especificarse una de las opciones siguientes. Sólo se puede especificar una opción.

MQSRO_FAIL_IF_QUIESCING

La llamada MQSUBRQ falla si el gestor de colas está en estado de desactivación temporal. En z/OS, para una aplicación CICS o IMS , esta opción también fuerza que la llamada MQSUBRQ falle si la conexión está en un estado de desactivación temporal.

Opción predeterminada: Si la opción descrita anteriormente no es necesaria, se debe utilizar la opción siguiente:

MQSRO_NONE

Utilice este valor para indicar que no se han especificado otras opciones; todas las opciones toman sus valores predeterminados.

MQSRO_NONE ayuda a la documentación del programa. Aunque no se pretende que esta opción se utilice con ninguna otra, debido a que su valor es cero, no se puede detectar este uso.

StrucId (MQCHAR4)

Este es el identificador de estructura; el valor debe ser:

MQSRO_STRUC_ID
Identificador de la estructura de opciones de solicitud de suscripción.

Para el lenguaje de programación C, también se define la constante MQSRO_STRUC_ID_ARRAY; tiene el mismo valor que MQSRO_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Siempre es un campo de entrada. El valor inicial de este campo es MQSRO_STRUC_ID.

Versión (MQLONG)

Este es el número de versión de la estructura; el valor debe ser:

MQSRO_VERSION_1
Version-1 Estructura de opciones de solicitud de suscripción.

La constante siguiente especifica el número de versión de la versión actual:

MQSRO_CURRENT_VERSION
Versión actual de la estructura de opciones de solicitud de suscripción.

Siempre es un campo de entrada. El valor inicial de este campo es MQSRO_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQSRO

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQSRO_STRUC_ID	'SRO¬'
Version	MQSRO_VERSION_1	1
Options	MQSRO_NONE	0
NumPubs	Ninguna	0

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQSRO_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQSRO MySRO = {MQSRO_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
** 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.
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

MQSTS-Estructura de informes de estado

La tabla siguiente resume los campos de la estructura.

Tabla 551. Campos en MQSTS

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
CompCode	Código de terminación del primer error	CompCode
Reason	Código de razón del primer error	Razón
PutSuccessCount	Número de llamadas de colocación asíncronas satisfactorias	SuccessCount
PutWarningCount	Número de llamadas de colocación asíncronas que tenían avisos	WarningCount
PutFailureCount	Número de llamadas de colocación asíncronas fallidas	FailureCount

Tabla 551. Campos en MQSTS (continuación)		
Campo	**Descripción**	**Tema**
ObjectType	Tipo de objeto anómalo	ObjectType
ObjectName	Nombre del objeto anómalo	ObjectName
ObjectQMgrName	Nombre del gestor de colas propietario del objeto anómalo	ObjectQMgrName
ResolvedObjectName	Nombre resuelto de la cola de destino	ResolvedObjectNombre
ResolvedQMgrName	Nombre resuelto del gestor de colas de destino	ResolvedQMgrName
Nota: Los campos restantes se ignoran si la versión es menor que MQSTS_VERSION_2.		
ObjectString	Nombre de objeto largo de objeto anómalo	ObjectString
SubName	Nombre de suscripción de suscripción anómala	SubName
OpenOptions	Opciones de apertura asociadas a la anomalía	OpenOptions
SubOptions	Opciones de suscripción asociadas con el error	SubOptions

Visión general de MQSTS

Finalidad: la estructura MQSTS es un parámetro de salida del mandato MQSTAT.

Conjunto de caracteres y codificación: los datos de caracteres de MQSTS están en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas *CodedCharSetId* . Los datos numéricos en MQSTS están en la codificación de máquina nativa; esto se proporciona mediante *Codificación*.

Uso: El mandato MQSTAT se utiliza para recuperar información de estado. Esta información se devuelve en una estructura MQSTS. Para obtener información sobre MQSTAT, consulte "MQSTAT-Recuperar información de estado" en la página 766.

Campos para MQSTS

La estructura MQSTS contiene los campos siguientes; los campos se describen en **orden alfabético**:

CompCode (MQLONG)
El código de terminación de la operación sobre la que se informa.

La interpretación de `CompCode` depende del valor del parámetro `MQSTAT Type` .

MQSTAT_TYPE_ASYNC_ERROR

Es el código de terminación resultante de una operación de colocación asíncrona anterior en el objeto especificado en `ObjectName`.

MQSTAT_TYPE_RECONNECTION

Si la conexión se está reconectando o no se ha podido volver a conectar, este es el código de finalización que ha hecho que la conexión empezara a volver a conectarse.

Si la conexión está conectada actualmente, el valor es `MQCC_OK`.

MQSTAT_TYPE_RECONNECTION_ERROR

Si la conexión no se ha podido volver a conectar, este es el código de terminación que ha hecho que la reconexión fallara.

Si la conexión está conectada actualmente, o se vuelve a conectar, el valor es MQCC_OK.

CompCode es siempre un campo de salida. Su valor inicial es MQCC_OK.

ObjectName (MQCHAR48)
El nombre del objeto sobre el que se informa.

La interpretación de ObjectName depende del valor del parámetro MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR

Es el nombre de la cola o tema utilizado en la operación de colocación, cuya anomalía se notifica en los campos *CompCode* y *Reason* de la estructura MQSTS .

MQSTAT_TYPE_RECONNECTION

Si la conexión se está reconectando, este es el nombre del gestor de colas asociado a la conexión.

MQSTAT_TYPE_RECONNECTION_ERROR

Si la conexión no ha podido volver a conectarse, este es el nombre del objeto que ha hecho que fallara la reconexión. La razón de la anomalía se notifica en los campos *CompCode* y *Reason* de la estructura MQSTS .

ObjectName es un campo de salida. Su valor inicial es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ObjectQMgrNombre (MQCHAR48)
El nombre del gestor de colas sobre el que se informa.

La interpretación de ObjectQMgrName depende del valor del parámetro MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR

Es el nombre del gestor de colas en el que se define el objeto *ObjectName* . Un nombre que está completamente en blanco hasta el primer carácter nulo o el final del campo indica el gestor de colas al que está conectada la aplicación (el gestor de colas local).

MQSTAT_TYPE_RECONNECTION

En blanco.

MQSTAT_TYPE_RECONNECTION_ERROR

Si la conexión no ha podido volver a conectarse, este es el nombre del objeto que ha hecho que fallara la reconexión. La razón de la anomalía se notifica en los campos *CompCode* y *Reason* de la estructura MQSTS .

ObjectQMgrName es un campo de salida. Su valor es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ObjectString (MQCHARV)
Nombre de objeto largo del objeto anómalo sobre el que se informa. Sólo está presente en la versión 2 de MQSTS o superior.

La interpretación de ObjectString depende del valor del parámetro MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR

Es el nombre de objeto largo de la cola o tema utilizado en la operación MQPUT , que ha fallado.

MQSTAT_TYPE_RECONNECTION

Serie de caracteres de longitud cero

MQSTAT_TYPE_RECONNECTION_ERROR

Es el nombre de objeto largo del objeto que ha hecho que fallara la reconexión.

ObjectString es un campo de salida. Su valor inicial es una serie de longitud cero.

ObjectType (MQLONG)
El tipo del objeto especificado en *ObjectName* sobre el que se informa.

Los valores posibles de ObjectType se listan en "MQOT_ * (Tipos de objeto y tipos de objeto ampliados)" en la página 147.

ObjectType es un campo de salida. Su valor inicial es MQOT_Q.

OpenOptions (MQLONG)
El OpenOptions utilizado para abrir el objeto sobre el que se informa. Sólo está presente en la versión 2 de MQSTS o superior.

El valor de OpenOptions depende del valor del parámetro MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR

Cero.

MQSTAT_TYPE_RECONNECTION

Cero.

MQSTAT_TYPE_RECONNECTION_ERROR

El OpenOptions utilizado cuando se produjo la anomalía. La razón de la anomalía se notifica en los campos *CompCode* y *Reason* de la estructura MQSTS .

OpenOptions es un campo de salida. Su valor inicial es cero.

Recuento de PutFailure(MQLONG)
El número de operaciones de colocación asíncrona que han fallado.

El valor de PutFailureCount depende del valor del parámetro MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR

Número de operaciones de colocación asíncrona en el objeto especificado en la estructura MQSTS que se ha completado con MQCC_FAILED.

MQSTAT_TYPE_RECONNECTION

Cero.

MQSTAT_TYPE_RECONNECTION_ERROR

Cero.

PutFailureCount es un campo de salida. Su valor inicial es cero.

Recuento de PutSuccess(MQLONG)
El número de operaciones de colocación asíncronas que se han realizado correctamente.

El valor de PutSuccessCount depende del valor del parámetro MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR

Número de operaciones de colocación asíncrona en el objeto especificado en la estructura MQSTS que se ha completado con MQCC_OK.

MQSTAT_TYPE_RECONNECTION

Cero.

MQSTAT_TYPE_RECONNECTION_ERROR

Cero.

PutSuccessCount es un campo de salida. Su valor inicial es cero.

Recuento de PutWarning(MQLONG)
Número de operaciones de colocación asíncronas que han finalizado con un aviso.

El valor de PutWarningCount depende del valor del parámetro MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR

Número de operaciones de colocación asíncrona en el objeto especificado en la estructura MQSTS que se ha completado con MQCC_WARNING.

MQSTAT_TYPE_RECONNECTION

Cero.

MQSTAT_TYPE_RECONNECTION_ERROR

Cero.

PutWarningCount es un campo de salida. Su valor inicial es cero.

SubName (MQCHARV)
El nombre de la suscripción anómala. Sólo está presente en la versión 2 de MQSTS o superior.

La interpretación de SubName depende del valor del parámetro MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR

Serie de longitud cero.

MQSTAT_TYPE_RECONNECTION

Serie de longitud cero.

MQSTAT_TYPE_RECONNECTION_ERROR

El nombre de la suscripción que ha hecho que fallara la reconexión. Si no hay ningún nombre de suscripción disponible, o la anomalía no está relacionada con una suscripción, se trata de una serie de longitud cero.

SubName es un campo de salida. Su valor inicial es una serie de longitud cero.

SubOptions (MQLONG)
El SubOptions utilizado para abrir la suscripción anómala. Sólo está presente en la versión 2 de MQSTS o superior.

La interpretación de SubOptions depende del valor del parámetro MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR

Cero.

MQSTAT_TYPE_RECONNECTION

Cero.

MQSTAT_TYPE_RECONNECTION_ERROR

El SubOptions utilizado cuando se produjo la anomalía. Si la anomalía no está relacionada con la suscripción a un tema, el valor devuelto es cero.

SubOptions es un campo de salida. Su valor inicial es cero.

Razón (MQLONG)
El código de razón de la operación sobre la que se informa.

La interpretación de Reason depende del valor del parámetro MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR

Este es el código de razón resultante de una operación de colocación asíncrona anterior en el objeto especificado en ObjectName.

MQSTAT_TYPE_RECONNECTION

Si la conexión se está reconectando o no se ha podido reconectar, este es el código de razón que ha hecho que la reconexión empezara a reconectarse.

Si la conexión está conectada actualmente, el valor es MQRC_NONE.

MQSTAT_TYPE_RECONNECTION_ERROR

Si la conexión no ha podido volver a conectarse, este es el código de razón que ha hecho que la reconexión fallara.

Si la conexión está conectada actualmente, o se vuelve a conectar, el valor es MQRC_NONE.

Reason es un campo de salida. Su valor inicial es MQRC_NONE.

ResolvedObjectNombre (MQCHAR48)
El nombre del objeto nombrado en *ObjectName* después de que el gestor de colas local resuelva el nombre.

La interpretación de ResolvedObjectName depende del valor del parámetro MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR

ResolvedObjectName es el nombre del objeto nombrado en *ObjectName* después de que el gestor de colas local resuelva el nombre. El nombre devuelto es el nombre de un objeto que existe en el gestor de colas identificado por *ResolvedQMgrName*.

MQSTAT_TYPE_RECONNECTION

En blanco.

MQSTAT_TYPE_RECONNECTION_ERROR

En blanco.

ResolvedObjectName es un campo de salida. Su valor inicial es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

ResolvedQMgrNombre (MQCHAR48)
El nombre del gestor de colas de destino después de que el gestor de colas local resuelva el nombre.

La interpretación de ResolvedQMgrName depende del valor del parámetro MQSTAT Type .

MQSTAT_TYPE_ASYNC_ERROR

ResolvedQMgrName es el nombre del gestor de colas de destino después de que el gestor de colas local resuelva el nombre. El nombre devuelto es el nombre del gestor de colas que es propietario del objeto identificado por *ResolvedObjectName*. *ResolvedQMgrName* puede ser el nombre del gestor de colas local.

MQSTAT_TYPE_RECONNECTION

En blanco.

MQSTAT_TYPE_RECONNECTION_ERROR

En blanco.

ResolvedQMgrName es siempre un campo de salida. Su valor inicial es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

StrucId (MQCHAR4)
Identificador de la estructura de informes de estado, MQSTS.

StrucId es el identificador de estructura. El valor debe ser:

MQSTS_ID_STRUCT
Identificador de la estructura de informes de estado.

Para el lenguaje de programación C, la constante MQSTS_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQSTS_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

StrucId es siempre un campo de entrada. Su valor inicial es MQSTS_STRUC_ID.

Versión (MQLONG)
El número de versión de la estructura.

El valor debe ser:

MQSTS_VERSION_1
Estructura de informes de estado de la versión 1.

MQSTS_VERSION_2
Estructura de informes de estado de la versión 2.

La constante siguiente especifica el número de versión de la versión actual:

MQSTS_CURRENT_VERSION
Versión actual de la estructura de informes de estado. La versión actual es MQSTS_VERSION_2.

Version es siempre un campo de entrada. Su valor inicial es MQSTS_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQSTS

Tabla 552. Valores iniciales de campos en MQSTS

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQSTS_ID_STRUCT	'STAT¬'
Version	MQSTS_VERSION_1	1
CompCode	MQCC_OK	0
Reason	MQRC_NONE	0
PutSuccessCount	Ninguna	0
PutWarningCount	Ninguna	0
PutFailureCount	Ninguna	0
ObjectType	MQOT_Q	1
ObjectName	Ninguna	Serie nula o espacios en blanco
ObjectQMgrName	Ninguna	Serie nula o espacios en blanco
ResolvedObjectName	Ninguna	Serie nula o espacios en blanco
ResolvedQMgrName	Ninguna	Serie nula o espacios en blanco
ObjectString	MQCHARV_PREDETERMINADO	{NULL,0,0,0,-3}
SubName	MQCHARV_PREDETERMINADO	{NULL,0,0,0,-3}
OpenOptions	Ninguna	0
SubOptions	Ninguna	0

Tabla 552. Valores iniciales de campos en MQSTS (continuación)

Nombre de campo	Nombre de constante	Valor de constante

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macro MQSTS_DEFAULT contiene los valores listados anteriormente. Se puede utilizar de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQSTS MySTS = {MQSTS_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
      ** 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 **
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
     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)
```

MQTM-Mensaje de desencadenante

La tabla siguiente resume los campos de la estructura.

Tabla 553. Campos en MQTM

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
QName	Nombre de la cola desencadenada	QName
ProcessName	Nombre del objeto de proceso	ProcessName
TriggerData	Datos desencadenantes	TriggerData
ApplType	Tipo de aplicación	ApplType
ApplId	Identificador de aplicación	ApplId
EnvData	Datos de entorno	EnvData
UserData	Datos de usuario	UserData

Visión general de MQTM

Finalidad: La estructura MQTM describe los datos del mensaje desencadenante que envía el gestor de colas a una aplicación de supervisor desencadenante cuando se produce un suceso desencadenante para una cola.

Esta estructura forma parte de la interfaz de supervisor desencadenante (TMI) de WebSphere MQ , que es una de las interfaces de infraestructura de WebSphere MQ .

Nombre de formato: MQFMT_TRIGGER.

Conjunto de caracteres y codificación: los datos de caracteres de MQTM se encuentran en el juego de caracteres del gestor de colas que genera MQTM. Los datos numéricos en MQTM están en la codificación de máquina del gestor de colas que genera MQTM.

El juego de caracteres y la codificación de MQTM se proporcionan mediante los campos *CodedCharSetId* y *Encoding* en:

- El MQMD (si la estructura MQTM está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQTM (todos los demás casos).

Uso: Es posible que una aplicación de supervisor desencadenante tenga que pasar parte o toda la información del mensaje desencadenante a la aplicación que inicia la aplicación de supervisor desencadenante. La información que puede necesitar la aplicación iniciada incluye *QName*, *TriggerData* y *UserData*. La aplicación de supervisor desencadenante puede pasar la estructura MQTM directamente a la aplicación iniciada, o pasar una estructura MQTMC2 en su lugar, en función de lo que permita el entorno y lo que sea conveniente para la aplicación iniciada. Para obtener información sobre MQTMC2, consulte "MQTMC2 -Mensaje de desencadenante 2 (formato de caracteres)" en la página 584.

- En z/OS, para una aplicación MQAT_CICS que se inicia utilizando la transacción CKTI, toda la estructura de mensajes desencadenantes MQTM se pone a disposición de la transacción iniciada; la información se puede recuperar utilizando el mandato EXEC CICS RETRIEVE.
- En IBM i, la aplicación de supervisor desencadenante proporcionada con WebSphere MQ pasa una estructura MQTMC2 a la aplicación iniciada.

Para obtener información sobre cómo utilizar desencadenantes, consulte Inicio de aplicaciones WebSphere MQ utilizando desencadenantes .

MQMD para un mensaje desencadenante: los campos del MQMD de un mensaje desencadenante generado por el gestor de colas se establecen de la forma siguiente:

Campo de MQMD	Valor utilizado
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_1
Report	MQRO_NONE
MsgType	MQMT_DATAGRAM
Expiry	MQEI_UNLIMITED
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	Atributo *CodedCharSetId* del gestor de colas
Format	MQFMT_TRIGGER
Priority	Atributo *DefPriority* de la cola de inicio
Persistence	MQPER_NOT_PERSISTENT
MsgId	Un valor exclusivo
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	Espacios en blanco
ReplyToQMgr	Nombre de gestor de colas
UserIdentifier	Espacios en blanco
AccountingToken	MQACT_NONE
ApplIdentityData	Espacios en blanco
PutApplType	MQAT_QMGR, o según corresponda para el agente de canal de mensajes
PutApplName	Primeros 28 bytes del nombre del gestor de colas
PutDate	Fecha en la que se envía el mensaje desencadenante
PutTime	Hora a la que se envía el mensaje desencadenante
ApplOriginData	Espacios en blanco

Se recomienda una aplicación que genere un mensaje desencadenante para establecer valores similares, excepto para lo siguiente:

- El campo *Priority* se puede establecer en MQPRI_PRIORITY_AS_Q_DEF (el gestor de colas lo cambiará por la prioridad predeterminada para la cola de inicio cuando se coloque el mensaje).
- El campo *ReplyToQMgr* se puede establecer en blancos (el gestor de colas lo cambiará por el nombre del gestor de colas local cuando coloque el mensaje).
- Establezca los campos de contexto según corresponda para la aplicación.

Campos para MQTM

La estructura MQTM contiene los campos siguientes; los campos se describen en **orden alfabético**:

ApplId (MQCHAR256)

Es una serie de caracteres que identifica la aplicación que se va a iniciar y la utiliza la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo *ApplId* del objeto de proceso identificado por el campo *ProcessName* ; consulte "Atributos de las definiciones de proceso" en la página 851 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

El significado de *ApplId* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por WebSphere MQ requiere que *ApplId* sea el nombre de un programa ejecutable. Las notas siguientes se aplican a los entornos indicados:

- En z/OS, *ApplId* es:
  - Un identificador de transacción CICS , para aplicaciones iniciadas utilizando la CKTI de transacción de supervisor desencadenante CICS
  - Un identificador de transacción de IMS , para aplicaciones iniciadas utilizando el supervisor desencadenante de IMS CSQQTRMN
- En sistemas Windows , el nombre de programa puede tener como prefijo una unidad y una vía de acceso de directorio.
- En IBM i, el nombre de programa puede tener como prefijo un nombre de biblioteca y/carácter.
- En sistemas UNIX , el nombre de programa puede tener como prefijo una vía de acceso de directorio.

La longitud de este campo la proporciona MQ_PROCESS_APPL_ID_LENGTH. El valor inicial de este campo es la serie nula en C y 256 caracteres en blanco en otros lenguajes de programación.

ApplType (MQLONG)

Esto identifica la naturaleza del programa que se va a iniciar y la utiliza la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo *ApplType* del objeto de proceso identificado por el campo *ProcessName* ; consulte "Atributos de las definiciones de proceso" en la página 851 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

ApplType puede tener uno de los siguientes valores estándar. También se pueden utilizar tipos definidos por el usuario, pero deben restringirse a los valores del rango MQAT_USER_FIRST a MQAT_USER_LAST:

MQAT_AIX
Aplicación AIX (el mismo valor que MQAT_UNIX).

MQAT_LOTE
aplicación por lotes

INTERMEDIARIO
Aplicación de intermediario

MQAT_CICS
Transacción CICS .

MQAT_CICS_BRIDGE
Aplicación puente CICS .

MQAT_CICS_VSE
Transacción CICS/VSE .

MQAT_DOS
WebSphere MQ Aplicación cliente MQI en PC DOS.

MQAT_IMS
AplicaciónIMS .

MQAT_IMS_BRIDGE
IMS .

MQAT_JAVA
Aplicación Java.

MQAT_MVS
MVS o aplicación TSO (el mismo valor que MQAT_ZOS).

MQAT_NOTES_AGENT
Lotus Notes .

MQAT_NSK
Aplicación HP Integrity NonStop Server .

MQAT_OS390
OS/390 (mismo valor que MQAT_ZOS).

MQAT_OS400
Aplicación IBM i .

MQAT_RRS_BATCH
Aplicación por lotes RRS.

MQAT_UNIX
Aplicación UNIX .

MQAT_DESCONOCIDO
Aplicación de tipo desconocido.

USUARIO_MQ
Tipo de aplicación definido por el usuario.

MQAT_VOS
Aplicación Stratus VOS.

MQAT_WINDOWS
Aplicación Windows de 16 bits.

MQAT_WINDOWS_NT
Aplicación Windows de 32 bits.

MQAT_WLM
Aplicación del gestor de carga de trabajo de z/OS .

MQAT_XCF
XCF.

MQAT_ZOS
Aplicaciónz/OS .

MQAT_USER_FIRST
Valor más bajo para el tipo de aplicación definido por el usuario.

MQAT_USER_LAST
Valor más alto para el tipo de aplicación definido por el usuario.

El valor inicial de este campo es 0.

EnvData (MQCHAR128)

Es una serie de caracteres que contiene información relacionada con el entorno perteneciente a la aplicación que se va a iniciar y la utiliza la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo *EnvData* del objeto de proceso identificado por el campo *ProcessName* ; consulte "Atributos de las definiciones de proceso" en la página 851 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

En z/OS, para una aplicación CICS iniciada utilizando la transacción CKTI, o una aplicación IMS que se va a iniciar utilizando la transacción CSQQTRMN, esta información no se utiliza.

La longitud de este campo la proporciona MQ_PROCESS_ENV_DATA_LENGTH. El valor inicial de este campo es la serie nula en C y 128 caracteres en blanco en otros lenguajes de programación.

ProcessName (MQCHAR48)

Es el nombre del objeto de proceso del gestor de colas especificado para la cola desencadenada y puede ser utilizado por la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo *ProcessName* de la cola identificada por el campo *QName* ; consulte "Atributos para colas" en la página 816 para obtener detalles de este atributo.

Los nombres que son más cortos que la longitud definida del campo siempre se rellenan a la derecha con espacios en blanco; no finalizan prematuramente con un carácter nulo.

La longitud de este campo la proporciona MQ_PROCESS_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

QName (MQCHAR48)

Es el nombre de la cola para la que se ha producido un suceso desencadenante y lo utiliza la aplicación iniciada por la aplicación de supervisor desencadenante. El gestor de colas inicializa este campo con el valor del atributo *QName* de la cola desencadenada; consulte "Atributos para colas" en la página 816 para obtener detalles de este atributo.

Los nombres que son más cortos que la longitud definida del campo se rellenan a la derecha con espacios en blanco; no finalizan prematuramente con un carácter nulo.

La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

StrucId (MQCHAR4)

Es el identificador de estructura. El valor debe ser:

MQTM_STRUC_ID
Identificador de la estructura del mensaje desencadenante.

Para el lenguaje de programación C, también se define la constante MQTM_STRUC_ID_ARRAY; tiene el mismo valor que MQTM_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQTM_STRUC_ID.

TriggerData (MQCHAR64)

Se trata de datos de formato libre para que los utilice la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo *TriggerData* de la cola identificada por el campo *QName* ; consulte "Atributos para colas" en la página 816 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

En z/OS, para una aplicación CICS iniciada utilizando la transacción CKTI, esta información no se utiliza.

La longitud de este campo la proporciona MQ_TRIGGER_DATA_LENGTH. El valor inicial de este campo es la serie nula en C y 64 caracteres en blanco en otros lenguajes de programación.

UserData (MQCHAR128)

Es una serie de caracteres que contiene información de usuario relevante para la aplicación que se va a iniciar y la utiliza la aplicación de supervisor desencadenante que recibe el mensaje desencadenante. El gestor de colas inicializa este campo con el valor del atributo *UserData* del objeto de proceso identificado por el campo *ProcessName* ; consulte “Atributos de las definiciones de proceso” en la página 851 para obtener detalles de este atributo. El contenido de estos datos no es significativo para el gestor de colas.

Para Microsoft Windows, la serie de caracteres no debe contener comillas dobles si la definición de proceso se va a pasar a **runmqtrm**.

La longitud de este campo la proporciona MQ_PROCESS_USER_DATA_LENGTH. El valor inicial de este campo es la serie nula en C y 128 caracteres en blanco en otros lenguajes de programación.

Versión (MQLONG)

Es el número de versión de la estructura. El valor debe ser:

MQTM_VERSION_1
Número de versión para la estructura del mensaje desencadenante.

La constante siguiente especifica el número de versión de la versión actual:

MQTM_CURRENT_VERSION
Versión actual de la estructura del mensaje desencadenante.

El valor inicial de este campo es MQTM_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQTM

Tabla 554. Valores iniciales de los campos en MQTM para MQTM

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQTM_STRUC_ID	'TM¬¬'
Version	MQTM_VERSION_1	1
QName	Ninguna	Serie nula o espacios en blanco
ProcessName	Ninguna	Serie nula o espacios en blanco
TriggerData	Ninguna	Serie nula o espacios en blanco
ApplType	Ninguna	0
ApplId	Ninguna	Serie nula o espacios en blanco
EnvData	Ninguna	Serie nula o espacios en blanco
UserData	Ninguna	Serie nula o espacios en blanco

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQTM_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQTM MyTM = {MQTM_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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).
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

MQTMC2 -Mensaje de desencadenante 2 (formato de caracteres)

La tabla siguiente resume los campos de la estructura.

Tabla 555. Campos en MQTMC2

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
QName	Nombre de la cola desencadenada	QName
ProcessName	Nombre del objeto de proceso	ProcessName
TriggerData	Datos desencadenantes	TriggerData
ApplType	Tipo de aplicación	ApplType
ApplId	Identificador de aplicación	ApplId
EnvData	Datos de entorno	EnvData
UserData	Datos de usuario	UserData
QMgrName	Nombre del gestor de colas	QMgrName

Visión general de MQTMC2

Finalidad: Cuando una aplicación de supervisor desencadenante recupera un mensaje desencadenante (MQTM) de una cola de inicio, es posible que el supervisor desencadenante tenga que pasar parte o toda la información del mensaje desencadenante a la aplicación que inicia el supervisor desencadenante.

La información que puede necesitar la aplicación iniciada incluye *QName*, *TriggerData* y *UserData*. La aplicación de supervisor desencadenante puede pasar la estructura MQTM directamente a la aplicación iniciada, o pasar una estructura MQTMC2 en su lugar, en función de lo que permita el entorno y lo que sea conveniente para la aplicación iniciada.

Esta estructura forma parte de la interfaz de supervisor desencadenante (TMI) de WebSphere MQ , que es una de las interfaces de infraestructura de WebSphere MQ .

Conjunto de caracteres y codificación: los datos de caracteres de MQTMC2 están en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas *CodedCharSetId* .

Uso: la estructura MQTMC2 es muy similar al formato de la estructura MQTM. La diferencia es que los campos que no son de tipo carácter en MQTM se cambian en MQTMC2 por campos de tipo carácter de la misma longitud, y el nombre del gestor de colas se añade al final de la estructura.

- En z/OS, para una aplicación MQAT_IMS que se inicia utilizando la aplicación CSQQTRMN, se pone a disposición de la aplicación iniciada una estructura MQTMC2 .
- En IBM i, la aplicación de supervisor desencadenante proporcionada con WebSphere MQ pasa una estructura MQTMC2 a la aplicación iniciada.

Campos para MQTMC2

La estructura MQTMC2 contiene los campos siguientes; los campos se describen en **orden alfabético**:

ApplId (MQCHAR256)

Identificador de aplicación.

Consulte el campo *ApplId* en la estructura MQTM.

ApplType (MQCHAR4)
Tipo de aplicación.

Este campo siempre contiene espacios en blanco, cualquiera que sea el valor del campo *ApplType* en la estructura MQTM del mensaje desencadenante original.

EnvData (MQCHAR128)
Datos de entorno.

Consulte el campo *EnvData* en la estructura MQTM.

ProcessName (MQCHAR48)
Nombre del objeto de proceso.

Consulte el campo *ProcessName* en la estructura MQTM.

QMgrName (MQCHAR48)
Nombre de gestor de colas.

Es el nombre del gestor de colas en el que se ha producido el suceso desencadenante.

QName (MQCHAR48)
Nombre de la cola desencadenada.

Consulte el campo *QName* en la estructura MQTM.

StrucId (MQCHAR4)
Identificador de estructura.

El valor debe ser:

MQTMC_STRUC_ID
Identificador de la estructura del mensaje desencadenante (formato de caracteres).

Para el lenguaje de programación C, también se define la constante MQTMC_STRUC_ID_ARRAY; tiene el mismo valor que MQTMC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

TriggerData (MQCHAR64)
Datos del desencadenante.

Consulte el campo *TriggerData* en la estructura MQTM.

UserData (MQCHAR128)
Datos de usuario.

Consulte el campo *UserData* en la estructura MQTM.

Versión (MQCHAR4)
Número de versión de la estructura.

El valor debe ser:

MQTMC_VERSION_2
Estructura del mensaje desencadenante de la versión 2 (formato de caracteres).

Para el lenguaje de programación C, también se define la constante MQTMC_VERSION_2_ARRAY ; tiene el mismo valor que MQTMC_VERSION_2, pero es una matriz de caracteres en lugar de una serie.

La constante siguiente especifica el número de versión de la versión actual:

MQTMC_CURRENT_VERSION
Versión actual de la estructura del mensaje desencadenante (formato de caracteres).

Valores iniciales y declaraciones de lenguaje para MQTMC2

Tabla 556. Valores iniciales de los campos en MQTMC2 para MQTMC2

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQTMC_STRUC_ID	'TMC¬'
Version	MQTMC_VERSION_2	'¬¬¬2'
QName	Ninguna	Serie nula o espacios en blanco
ProcessName	Ninguna	Serie nula o espacios en blanco
TriggerData	Ninguna	Serie nula o espacios en blanco
ApplType	Ninguna	Espacios en blanco
ApplId	Ninguna	Serie nula o espacios en blanco
EnvData	Ninguna	Serie nula o espacios en blanco
UserData	Ninguna	Serie nula o espacios en blanco
QMgrName	Ninguna	Serie nula o espacios en blanco

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQTMC2_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQTMC2 MyTMC = {MQTMC2_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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).
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
MQTMC              DSECT
MQTMC_STRUCID      DS   CL4    Structure identifier
MQTMC_VERSION      DS   CL4    Structure version number
MQTMC_QNAME        DS   CL48   Name of triggered queue
MQTMC_PROCESSNAME  DS   CL48   Name of process object
MQTMC_TRIGGERDATA  DS   CL64   Trigger data
MQTMC_APPLTYPE     DS   CL4    Application type
MQTMC_APPLID       DS   CL256  Application identifier
MQTMC_ENVDATA      DS   CL128  Environment data
MQTMC_USERDATA     DS   CL128  User data
MQTMC_QMGRNAME     DS   CL48   Queue manager name
*
MQTMC_LENGTH       EQU  *-MQTMC
                   ORG  MQTMC
MQTMC_AREA         DS   CL(MQTMC_LENGTH)
```

Declaración de Visual Basic

```
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
```

MQWIH - Cabecera de información de trabajo

La tabla siguiente resume los campos de la estructura.

Tabla 557. Campos en MQWIH		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
StrucLength	Longitud de la estructura MQWIH	StrucLength
Encoding	Codificación numérica de los datos que siguen a MQWIH	Codificación
CodedCharSetId	Identificador de juego de caracteres de los datos que siguen a MQWIH	CodedCharSetId
Format	Nombre de formato de los datos que siguen a MQWIH	Formato
Flags	Distintivos	Distintivos
ServiceName	Nombre del servicio	ServiceName
ServiceStep	Nombre de paso de servicio	ServiceStep
MsgToken	Señal de mensaje	MsgToken
Reserved	Reserved	Reserved

Visión general de MQWIH

Disponibilidad: todos los sistemas WebSphere MQ , además de los clientes WebSphere MQ conectados a estos sistemas.

Finalidad: la estructura MQWIH describe la información que debe estar presente al principio de un mensaje que debe ser manejado por el gestor de carga de trabajo de z/OS .

Nombre de formato: MQFMT_WORK_INFO_HEADER.

Juego de caracteres y codificación: los campos de la estructura MQWIH están en el juego de caracteres y la codificación proporcionados por los campos *CodedCharSetId* y *Encoding* de la estructura de cabecera que precede a MQWIH, o por los campos de la estructura MQMD si MQWIH está al principio de los datos del mensaje de aplicación.

El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de cola.

Uso: si el gestor de carga de trabajo z/OS va a procesar un mensaje, el mensaje debe empezar con una estructura MQWIH.

Campos para MQWIH

La estructura MQWIH contiene los campos siguientes; los campos se describen en **orden alfabético**:

CodedCharSetId (MQLONG)

Especifica el identificador de juego de caracteres de los datos que siguen a la estructura MQWIH; no se aplica a los datos de tipo carácter de la propia estructura MQWIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Puede utilizar el siguiente valor especial:

MQCCSI_INHERIT
Los datos de caracteres en los datos *siguientes* a esta estructura están en el mismo juego de caracteres que esta estructura.

El gestor de colas cambia este valor en la estructura enviada en el mensaje por el identificador de juego de caracteres real de la estructura. Si no se produce ningún error, la llamada MQGET no devuelve el valor MQCCSI_INHERIT.

MQCCSI_INHERIT no se puede utilizar si el valor del campo *PutApplType* en MQMD es MQAT_BROKER.

El valor inicial de este campo es MQCCSI_UNDEFINED.

Encoding (MQLONG)

Especifica la codificación numérica de los datos que siguen a la estructura MQWIH; no se aplica a los datos numéricos de la propia estructura MQWIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos.

El valor inicial de este campo es 0.

Flags (MQLONG)

El valor debe ser:

MQWIH_NONE
Sin distintivos.

El valor inicial de este campo es MQWIH_NONE.

Format (MQCHAR8)

Especifica el nombre de formato de los datos que siguen a la estructura MQWIH.

En la llamada MQPUT o MQPUT1 , la aplicación debe establecer este campo en el valor adecuado para los datos. Las reglas para codificar este campo son las mismas que para el campo *Format* en MQMD.

La longitud de este campo la proporciona MQ_FORMAT_LENGTH. El valor inicial de este campo es MQFMT_NONE.

MsgToken (MQBYTE16)

Se trata de una señal de mensaje que identifica de forma exclusiva el mensaje.

Para las llamadas MQPUT y MQPUT1 , este campo se ignora. La longitud de este campo la proporciona MQ_MSG_TOKEN_LENGTH. El valor inicial de este campo es MQMTOK_NONE.

Reservado (MQCHAR32)

Es un campo reservado; debe estar en blanco.

ServiceName (MQCHAR32)

Es el nombre del servicio que va a procesar el mensaje.

La longitud de este campo la proporciona MQ_SERVICE_NAME_LENGTH. El valor inicial de este campo es de 32 caracteres en blanco.

ServiceStep (MQCHAR8)

Es el nombre del paso de *ServiceName* con el que se relaciona el mensaje.

La longitud de este campo la proporciona MQ_SERVICE_STEP_LENGTH. El valor inicial de este campo es de 8 caracteres en blanco.

StrucId (MQCHAR4)

Es el identificador de estructura. El valor debe ser:

MQWIH_STRUC_ID
Identificador de la estructura de cabecera de información de trabajo.

Para el lenguaje de programación C, también se define la constante MQWIH_STRUC_ID_ARRAY; tiene el mismo valor que MQWIH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQWIH_STRUC_ID.

StrucLength (MQLONG)

Es la longitud de la estructura MQWIH. El valor debe ser:

MQWIH_LENGTH_1
Longitud de la estructura de cabecera de información de trabajo version-1 .

La constante siguiente especifica la longitud de la versión actual:

MQWIH_CURRENT_LENGTH
Longitud de la versión actual de la estructura de cabecera de información de trabajo.

El valor inicial de este campo es MQWIH_LENGTH_1.

Versión (MQLONG)

Es el número de versión de la estructura. El valor debe ser:

MQWIH_VERSION_1
Estructura de cabecera de información de trabajo Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQWIH_CURRENT_VERSION
Versión actual de la estructura de cabecera de información de trabajo.

El valor inicial de este campo es MQWIH_VERSION_1.

Valores iniciales y declaraciones de idioma para MQWIH

Tabla 558. Valores iniciales de los campos en MQWIH para MQWIH

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQWIH_STRUC_ID	'WIH¬'
Version	MQWIH_VERSION_1	1
StrucLength	MQWIH_LENGTH_1	120
Encoding	Ninguna	0
CodedCharSetId	MQCCSI_UNDEFINED	0
Format	MQFMT_NONE	Espacios en blanco
Flags	MQWIH_NONE	0
ServiceName	Ninguna	Espacios en blanco
ServiceStep	Ninguna	Espacios en blanco
MsgToken	MQMTOK_NONE	Nulos
Reserved	Ninguna	Espacios en blanco

Tabla 558. Valores iniciales de los campos en MQWIH para MQWIH (continuación)

Nombre de campo	Nombre de constante	Valor de constante

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. En el lenguaje de programación C, la variable de macroMQWIH_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQWIH MyWIH = {MQWIH_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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).
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

MQXP-Bloque de parámetros de salida

La tabla siguiente resume los campos de la estructura.

Tabla 559. Campos en MQXP

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
ExitId	Identificador de salida	ExitId
ExitReason	Razón para la invocación de la salida	ExitReason
ExitResponse	Respuesta de la salida	ExitResponse
ExitCommand	Código de llamada a la API	ExitCommand
ExitParmCount	Recuento de parámetros	Recuento deExitParm

Tabla 559. Campos en MQXP (continuación)		
Campo	**Descripción**	**Tema**
ExitUserArea	Área de usuario	ExitUserArea

Visión general de MQXP

Disponibilidad: z/OS.

Finalidad: la estructura MQXP se utiliza como parámetro de entrada/salida para la salida cruzada de API. Para obtener más información sobre esta salida, consulte La salida cruzada de API.

Conjunto de caracteres y codificación: los datos de caracteres de MQXP están en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas *CodedCharSetId* . Los datos numéricos en MQXP están en la codificación de máquina nativa; esto lo proporciona MQENC_NATIVE.

Campos para MQXP

La estructura MQXP contiene los campos siguientes; los campos se describen en **orden alfabético**:

ExitCommand (MQLONG)

Este campo se establece en la entrada de la rutina de salida. Identifica la llamada de API que ha hecho que se invocara la salida:

MQXC_CALLBACK
La llamada CALLBACK.

MQXC_MQBACK
Llamada MQBACK.

MQXC_MQCB
La llamada MQCB.

MQXC_MQCLOSE
Llamada MQCLOSE.

MQXC_MQCMIT
Llamada MQCMIT.

MQXC_MQCTL
La llamada MQCTL.

MQXC_MQGET
La llamada MQGET.

MQXC_MQINQ
Llamada MQINQ.

MQXC_MQOPEN
La llamada MQOPEN.

MQXC_MQPUT
La llamada MQPUT.

MQXC_MQPUT1
La llamada MQPUT1 .

MQXC_MQSET
La llamada MQSET.

MQXC_MQSTAT
Llamada MQSTAT.

MQXC_MQSUB
Llamada MQSUB.

MQXC_MQSUBRQ
Llamada MQSUBRQ.

Es un campo de entrada para la salida.

ExitId (MQLONG)

Se establece en la entrada de la rutina de salida e indica el tipo de salida:

MQXT_API_CROSSING_EXIT
Salida cruzada de API para CICS.

Es un campo de entrada para la salida.

Recuento de ExitParm(MQLONG)

Este campo se establece en la entrada de la rutina de salida. Contiene el número de parámetros que toma la llamada MQ . Son las siguientes:

Nombre de llamada	Número de parámetros
MQBACK	3
MQCLOSE	5
MQCMIT	3
MQGET	9
MQINQ	10
MQOPEN	6
MQPUT	8
MQPUT1	8
MQSET	10

Es un campo de entrada para la salida.

ExitReason (MQLONG)

Se establece en la entrada de la rutina de salida. Para la salida cruzada de API, indica si se llama a la rutina antes o después de la ejecución de la llamada de API:

MQXR_ANTES
Antes de la ejecución de la API.

MQXR_AFTER
Después de la ejecución de la API.

Es un campo de entrada para la salida.

ExitResponse (MQLONG)

El valor lo establece la salida para comunicarse con el llamante. Los valores siguientes están definidos:

MQXCC_Correcto
La salida se ha completado correctamente.

MQXCC_SUPPRESS_FUNCTION
Suprimir función.

Cuando este valor se establece mediante una salida cruzada de API llamada *antes* de la llamada de API, la llamada de API no se realiza. El *CompCode* para la llamada se establece en MQCC_FAILED, el *Reason* se establece en MQRC_SUPPRESSED_BY_EXIT y todos los demás parámetros permanecen como la salida que los deja.

Cuando este valor se establece mediante una salida cruzada de API denominada *después* de la llamada de API, el gestor de colas lo ignora.

MQXCC_SKIP_FUNCTION
Omitir función.

Cuando este valor se establece mediante una salida cruzada de API llamada *antes* de la llamada de API, la llamada de API no se realiza; *CompCode* y *Reason* y todos los demás parámetros permanecen como la salida que los deja.

Cuando este valor se establece mediante una salida cruzada de API denominada *después* de la llamada de API, el gestor de colas lo ignora.

Es un campo de salida de la salida.

Área ExitUser(MQBYTE16)

Este es un campo que está disponible para que lo utilice la salida. Se inicializa en cero binario para la longitud del campo antes de la primera invocación de la salida para la tarea, y posteriormente los cambios realizados en este campo por la salida se conservan entre las invocaciones de la salida. Se define el valor siguiente:

MQXUA_NONE
No hay información de usuario.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQXUA_NONE_ARRAY; tiene el mismo valor que MQXUA_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_EXIT_USER_AREA_LENGTH. Es un campo de entrada/salida para la salida.

Reservado (MQLONG)

Este es un campo reservado. Su valor no es significativo para la salida.

StrucId (MQCHAR4)

Es el identificador de estructura. El valor debe ser:

MQXP_STRUC_ID
Identificador de la estructura de parámetros de salida.

Para el lenguaje de programación C, también se define la constante MQXP_STRUC_ID_ARRAY; tiene el mismo valor que MQXP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para la salida.

Versión (MQLONG)

Es el número de versión de la estructura. El valor debe ser:

MQXP_VERSION_1
Número de versión para la estructura de bloque de parámetros de salida.

Nota: Cuando se introduce una nueva versión de esta estructura, el diseño de la parte existente no cambia. Por lo tanto, la salida debe comprobar que el número de versión es igual o mayor que la versión más baja que contiene los campos que la salida necesita utilizar.

Es un campo de entrada para la salida.

Declaraciones lingüísticas

Esta estructura está soportada en los siguientes lenguajes de programación.

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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).
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

MQXQH-Cabecera de cola de transmisión

La tabla siguiente resume los campos de la estructura.

Tabla 560. Campos en MQXQH		
Campo	**Descripción**	**Tema**
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
RemoteQName	Nombre de la cola de destino	RemoteQName
RemoteQMgrName	Nombre del gestor de colas de destino	RemoteQMgrName
MsgDesc	Descriptor de mensaje original	MsgDesc

Visión general de MQXQH

Disponibilidad: Todos los sistemas WebSphere MQ y clientes WebSphere MQ .

Finalidad: la estructura MQXQH describe la información que tiene como prefijo los datos de mensaje de aplicación de los mensajes cuando están en colas de transmisión. Una cola de transmisión es un tipo especial de cola local que contiene temporalmente mensajes destinados a colas remotas (es decir, destinados a colas que no pertenecen al gestor de colas local). Una cola de transmisión se indica mediante el atributo de cola *Usage* que tiene el valor MQUS_TRANSMISSION.

Nombre de formato: MQFMT_XMIT_Q_HEADER.

Conjunto de caracteres y codificación: los datos de MQXOH deben estar en el juego de caracteres proporcionado por el atributo de gestor de colas *CodedCharSetId* y la codificación del gestor de colas local proporcionada por MQENC_NATIVE.

Establezca el juego de caracteres y la codificación de MQXQH en los campos *CodedCharSetId* y *Encoding* en:

- El MQMD separado (si la estructura MQXQH está al principio de los datos del mensaje), o
- La estructura de cabecera que precede a la estructura MQXQH (todos los demás casos).

Uso: un mensaje que está en una cola de transmisión tiene *dos* descriptores de mensaje:

- Un descriptor de mensaje se almacena por separado de los datos de mensaje; esto se denomina *descriptor de mensaje separado*y lo genera el gestor de colas cuando el mensaje se coloca en la cola de transmisión. Algunos de los campos del descriptor de mensaje separado se copian del descriptor de mensaje proporcionado por la aplicación en la llamada MQPUT o MQPUT1 .

  El descriptor de mensaje separado es el que se devuelve a la aplicación en el parámetro *MsgDesc* de la llamada MQGET cuando se elimina el mensaje de la cola de transmisión.
- Un segundo descriptor de mensaje se almacena dentro de la estructura MQXQH como parte de los datos del mensaje; esto se denomina *descriptor de mensaje incorporado*y es una copia del descriptor de mensaje proporcionado por la aplicación en la llamada MQPUT o MQPUT1 (con variaciones menores).

  El descriptor de mensaje incorporado es siempre un MQMD version-1 . Si el mensaje colocado por la aplicación tiene valores no predeterminados para uno o más de los campos version-2 en MQMD, una estructura MQMDE sigue a la MQXQH y, a su vez, va seguida de los datos del mensaje de la aplicación (si los hay). El MQMDE es:

  – Generado por el gestor de colas (si la aplicación utiliza un MQMD version-2 para transferir el mensaje), o
  – Ya está presente al principio de los datos del mensaje de aplicación (si la aplicación utiliza un MQMD version-1 para transferir el mensaje).

  El descriptor de mensaje incorporado es el que se devuelve a la aplicación en el parámetro *MsgDesc* de la llamada MQGET cuando se elimina el mensaje de la cola de destino final.

Campos en el descriptor de mensaje separado: el gestor de colas establece los campos en el descriptor de mensaje separado tal como se muestra. Si el gestor de colas no da soporte al MQMD version-2 , se utiliza un MQMD version-1 sin pérdida de función.

Campo en MQMD separado	Valor utilizado
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_2
Report	Se copia desde el descriptor de mensaje incorporado, pero con los bits identificados por MQRO_ACCEPT_UNSUP_IF_XMIT_MASK establecidos en cero. (Esto impide que se genere un mensaje de informe COA o COD cuando se coloca un mensaje en o se elimina de una cola de transmisión.)
MsgType	Copiado del descriptor de mensaje incorporado.
Expiry	Copiado del descriptor de mensaje incorporado.
Feedback	Copiado del descriptor de mensaje incorporado.
Encoding	MQENC_NATIVE (ver nota)
CodedCharSetId	Atributo *CodedCharSetId* del gestor de colas.
Format	MQFMT_XMIT_Q_HEADER
Priority	Copiado del descriptor de mensaje incorporado.
Persistence	Copiado del descriptor de mensaje incorporado.
MsgId	El gestor de colas genera un nuevo valor. Este identificador de mensaje es diferente del *MsgId* que el gestor de colas puede haber generado para el descriptor de mensaje incorporado (consulte más arriba).
CorrelId	*MsgId* del descriptor de mensaje incorporado. Para los mensajes que se transfieren a SYSTEM.CLUSTER.TRANSMIT.QUEUE, *CorrelId* está reservado para uso interno.
BackoutCount	0
ReplyToQ	Copiado del descriptor de mensaje incorporado.
ReplyToQMgr	Copiado del descriptor de mensaje incorporado.
UserIdentifier	Copiado del descriptor de mensaje incorporado.
AccountingToken	Copiado del descriptor de mensaje incorporado. Para los mensajes que se transfieren a SYSTEM.CLUSTER.TRANSMIT.QUEUE, *AccountingToken* está reservado para uso interno.
ApplIdentityData	Copiado del descriptor de mensaje incorporado.
PutApplType	MQAT_QMGR
PutApplName	Primeros 28 bytes del nombre del gestor de colas.
PutDate	Fecha en la que se colocó el mensaje en la cola de transmisión.
PutTime	Hora en que se colocó el mensaje en la cola de transmisión.
ApplOriginData	Espacios en blanco
GroupId	MQGI_NONE
MsgSeqNumber	1
Offset	0
MsgFlags	MQMF_NONE
OriginalLength	MQOL_UNDEFINED

- En Windows, el valor de MQENC_NATIVE para Micro Focus COBOL difiere del valor de C. El valor del campo *Encoding* en el descriptor de mensaje separado es siempre el valor para C en estos entornos; este valor es 546 en decimal. Además, los campos enteros de la estructura MQXQH están en la codificación que corresponde a este valor (la codificación Intel nativa).

Campos del descriptor de mensaje incorporado: los campos del descriptor de mensaje incorporado tienen los mismos valores que los del parámetro *MsgDesc* de la llamada MQPUT o MQPUT1 , excepto para lo siguiente:

- El campo *Version* siempre tiene el valor MQMD_VERSION_1.
- Si el campo *Priority* tiene el valor MQPRI_PRIORITY_AS_Q_DEF, se sustituye por el valor del atributo *DefPriority* de la cola.
- Si el campo *Persistence* tiene el valor MQPER_PERSISTENCE_AS_Q_DEF, se sustituye por el valor del atributo *DefPersistence* de la cola.
- Si el campo *MsgId* tiene el valor MQMI_NONE, o se ha especificado la opción MQPMO_NEW_MSG_ID, o el mensaje es un mensaje de lista de distribución, *MsgId* se sustituye por un nuevo identificador de mensaje generado por el gestor de colas.

  Cuando un mensaje de lista de distribución se divide en mensajes de lista de distribución más pequeños colocados en diferentes colas de transmisión, el campo *MsgId* de cada uno de los nuevos descriptores de mensajes incorporados es el mismo que el del mensaje de lista de distribución original.
- Si se ha especificado la opción MQPMO_NEW_CORREL_ID, *CorrelId* se sustituye por un nuevo identificador de correlación generado por el gestor de colas.
- Los campos de contexto se establecen según lo indicado por las opciones MQPMO_ * _CONTEXT especificadas en el parámetro *PutMsgOpts* ; los campos de contexto son:
  - *AccountingToken*
  - *ApplIdentityData*
  - *ApplOriginData*
  - *PutApplName*
  - *PutApplType*
  - *PutDate*
  - *PutTime*
  - *UserIdentifier*
- Los campos version-2 (si estaban presentes) se eliminan del MQMD y se mueven a una estructura MQMDE, si uno o varios de los campos version-2 tienen un valor no predeterminado.

Colocación de mensajes en colas remotas: Cuando una aplicación coloca un mensaje en una cola remota (especificando directamente el nombre de la cola remota o utilizando una definición local de la cola remota), el gestor de colas local:

- Crea una estructura MQXQH que contiene el descriptor de mensaje incorporado
- Añade un MQMDE si se necesita uno y no está ya presente
- Añade los datos de mensaje de aplicación
- Coloca el mensaje en una cola de transmisión adecuada

Colocación de mensajes directamente en colas de transmisión: una aplicación también puede colocar un mensaje directamente en una cola de transmisión. En este caso, la aplicación debe prefijar los datos del mensaje de aplicación con una estructura MQXQH e inicializar los campos con los valores adecuados. Además, el campo *Format* del parámetro *MsgDesc* de la llamada MQPUT o MQPUT1 debe tener el valor MQFMT_XMIT_Q_HEADER.

Los datos de tipo carácter de la estructura MQXQH creada por la aplicación deben estar en el juego de caracteres del gestor de colas local (definido por el atributo de gestor de colas *CodedCharSetId*) y los datos enteros deben estar en la codificación de máquina nativa. Además, los datos de tipo carácter de la estructura MQXQH deben rellenarse con espacios en blanco hasta la longitud definida del campo;

los datos no deben finalizarse prematuramente utilizando un carácter nulo, porque el gestor de colas no convierte los caracteres nulos y posteriores en blancos en la estructura MQXQH.

Sin embargo, el gestor de colas no comprueba que haya una estructura MQXQH o que se hayan especificado valores válidos para los campos.

Las aplicaciones no deben colocar sus mensajes directamente en SYSTEM.CLUSTER.TRANSMIT.QUEUE.

Obtener mensajes de colas de transmisión: las aplicaciones que obtienen mensajes de una cola de transmisión deben procesar la información en la estructura MQXQH de una forma adecuada. La presencia de la estructura MQXQH al principio de los datos del mensaje de aplicación se indica mediante el valor MQFMT_XMIT_Q_HEADER que se devuelve en el campo *Format* del parámetro *MsgDesc* de la llamada MQGET. Los valores devueltos en los campos *CodedCharSetId* y *Encoding* en el parámetro *MsgDesc* indican el juego de caracteres y la codificación de los datos de caracteres y enteros en la estructura MQXQH. El juego de caracteres y la codificación de los datos del mensaje de aplicación se definen mediante los campos *CodedCharSetId* y *Encoding* en el descriptor de mensaje incorporado.

Campos para MQXQH

La estructura MQXQH contiene los campos siguientes; los campos se describen en **orden alfabético**:

MsgDesc (MQMD1)

Este es el descriptor de mensaje incorporado y es una copia cercana del MQMD del descriptor de mensaje que se ha especificado como parámetro *MsgDesc* en la llamada MQPUT o MQPUT1 cuando el mensaje se colocó originalmente en la cola remota.

Nota: Se trata de un MQMD de version-1 .

Los valores iniciales de los campos de esta estructura son los mismos que los de la estructura MQMD.

RemoteQMgrNombre (MQCHAR48)

Es el nombre del gestor de colas o del grupo de compartición de colas que es propietario de la cola que es el destino aparente final del mensaje.

Si el mensaje es un mensaje de lista de distribución, *RemoteQMgrName* está en blanco.

La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

RemoteQName (MQCHAR48)

Este es el nombre de la cola de mensajes que es el destino final aparente del mensaje (esto podría no ser el destino final si, por ejemplo, esta cola se define en *RemoteQMgrName* para que sea una definición local de otra cola remota).

Si el mensaje es un mensaje de lista de distribución (es decir, el campo *Format* del descriptor de mensaje incorporado es MQFMT_DIST_HEADER), *RemoteQName* está en blanco.

La longitud de este campo la proporciona MQ_Q_NAME_LENGTH. El valor inicial de este campo es la serie nula en C y 48 caracteres en blanco en otros lenguajes de programación.

StrucId (MQCHAR4)

Es el identificador de estructura. El valor debe ser:

MQXQH_STRUC_ID
Identificador de la estructura de cabecera de cola de transmisión.

Para el lenguaje de programación C, también se define la constante MQXQH_STRUC_ID_ARRAY; tiene el mismo valor que MQXQH_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQXQH_STRUC_ID.

Versión (MQLONG)

Es el número de versión de la estructura. El valor debe ser:

MQXQH_VERSION_1
Número de versión para la estructura de cabecera de cola de transmisión.

La constante siguiente especifica el número de versión de la versión actual:

MQXQH_CURRENT_VERSION
Versión actual de la estructura de cabecera de cola de transmisión.

El valor inicial de este campo es MQXQH_VERSION_1.

Valores iniciales y declaraciones de lenguaje para MQXQH

Tabla 561. Valores iniciales de los campos en MQXQH para MQXQH

Nombre de campo	Nombre de constante	Valor de constante
StrucId	MQXQH_STRUC_ID	'XQH¬'
Version	MQXQH_VERSION_1	1
RemoteQName	Ninguna	Serie nula o espacios en blanco
RemoteQMgrName	Ninguna	Serie nula o espacios en blanco
MsgDesc	Los mismos nombres y valores que MQMD; consulte Tabla 515 en la página 440	-

Notas:

1. El símbolo ¬ representa un único carácter en blanco.
2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.
3. En el lenguaje de programación C, la variable de macroMQXQH_DEFAULT contiene los valores listados anteriormente. Utilícelo de la forma siguiente para proporcionar valores iniciales para los campos de la estructura:

```
MQXQH MyXQH = {MQXQH_DEFAULT};
```

Declaración C

```
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 */
};
```

declaración COBOL

```
**   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).
```

Declaración PL/I

```
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 */
```

Declaración High Level Assembler

```
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)
```

Declaración de Visual Basic

```
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
```

Llamadas de funciones

Esta sección proporciona información sobre todas las llamadas MQI que son posibles. Se proporcionan descripciones, sintaxis, información de parámetros, notas de uso e invocaciones de idioma para cada idioma posible para cada una de las diferentes llamadas.

Descripciones de llamadas

En esta sección se describen las llamadas MQI.

La ayuda en línea en las plataformas UNIX , en forma de páginas *man* , está disponible para estas llamadas.

Nota: Las llamadas asociadas con la conversión de datos, MQXCNVC y MQ_DATA_CONV_EXIT, se encuentran en "salida de conversión de datos" en la página 888.

Convenciones utilizadas en las descripciones de llamada

Para cada llamada, esta colección de temas proporciona una descripción de los parámetros y el uso de la llamada en un formato que es independiente del lenguaje de programación. Esto va seguido de invocaciones típicas de la llamada, y declaraciones típicas de sus parámetros, en cada uno de los lenguajes de programación soportados.

Importante: Al codificar llamadas de API de WebSphere MQ , debe asegurarse de que se proporcionan todos los parámetros relevantes (tal como se describe en las secciones siguientes). No hacerlo puede producir resultados imprevisibles.

La descripción de cada llamada contiene las secciones siguientes:

Nombre de llamada
El nombre de la llamada, seguido de una breve descripción de la finalidad de la llamada.

Parámetros
Para cada parámetro, el nombre va seguido de su tipo de datos entre paréntesis () y uno de los siguientes:

entrada
Proporcione información en el parámetro cuando realice la llamada.

salida
El gestor de colas devuelve información en el parámetro cuando la llamada se completa o falla.

entrada/salida
Proporcione información en el parámetro cuando realice la llamada y el gestor de colas cambie la información cuando la llamada se complete o falle.

Por ejemplo:

Compcode (MQLONG)-salida

En algunos casos, el tipo de datos es una estructura. En todos los casos, hay más información sobre el tipo de datos o la estructura en "Tipos de datos elementales" en la página 218.

Los dos últimos parámetros de cada llamada son un código de terminación y un código de razón. El código de terminación indica si la llamada se ha completado correctamente, parcialmente o no. En el código de razón se proporciona más información sobre el éxito parcial o el fracaso de la llamada. Para obtener más información sobre cada código de terminación y razón, consulte "Códigos de retorno" en la página 855.

Notas de uso
Información adicional sobre la llamada, describiendo cómo utilizarla y cualquier restricción sobre su uso.

Invocación de lenguaje ensamblador
Invocación típica de la llamada, y declaración de sus parámetros, en lenguaje ensamblador.

Invocación en C
Invocación típica de la llamada, y declaración de sus parámetros, en C.

Invocación en COBOL
Invocación típica de la llamada, y declaración de sus parámetros, en COBOL.

Invocación en PL/I
Invocación típica de la llamada, y declaración de sus parámetros, en PL/I.

Todos los parámetros se pasan por referencia.

Invocación en Visual Basic
Invocación típica de la llamada, y declaración de sus parámetros, en Visual Basic.

Otras convenciones de notación son:

Constantes
Los nombres de constantes se muestran en mayúsculas; por ejemplo, MQOO_OUTPUT. A continuación se muestra un conjunto de constantes que tienen el mismo prefijo: MQIA_ *. Consulte "Constantes" en la página 50 para ver el valor de una constante.

Matrices
En algunas llamadas, los parámetros son matrices de series de caracteres que no tienen tamaños fijos. En las descripciones de estos parámetros, una n minúscula representa una constante numérica. Cuando codifique la declaración para ese parámetro, sustituya el n por el valor numérico que necesite.

Utilización de las llamadas en el lenguaje C

Los parámetros que son *sólo de entrada* y de tipo MQHCONN, MQHOBJ, MQHMSG o MQLONG se pasan por valor. Para todos los demás parámetros, la *dirección* del parámetro se pasa por valor.

No es necesario que especifique todos los parámetros que se pasan por dirección cada vez que invoca una función. Cuando no necesite un parámetro determinado, especifique un puntero nulo como parámetro en la invocación de función, en lugar de la dirección de los datos de parámetro. Los parámetros para los cuales esto es posible se identifican en las descripciones de llamada.

No se devuelve ningún parámetro como valor de la llamada; en terminología C, esto significa que todas las llamadas devuelven `void`.

Declaración del parámetro Buffer

Las llamadas MQGET, MQPUTy MQPUT1 tienen cada una un parámetro que tiene un tipo de datos no definido: el parámetro *Buffer* . Utilice este parámetro para enviar y recibir los datos de mensaje de la aplicación.

Los parámetros de este tipo se muestran en los ejemplos de C como matrices de MQBYTE. Puede declarar los parámetros de esta forma, pero normalmente es más conveniente declararlos como la estructura particular que describe el diseño de los datos en el mensaje. El prototipo de función declara el parámetro como un puntero a void, de modo que puede especificar la dirección de cualquier tipo de datos como parámetro en la invocación de la llamada.

Puntero a vacío es un puntero a datos de formato no definido. Se define como:

```
typedef void *PMQVOID;
```

MQBACK-Retrotraer cambios

La llamada MQBACK indica al gestor de colas que se deben restituir todas las obtenciones y colocaciones de mensajes que se han producido desde el último punto de sincronización.

Los mensajes colocados como parte de una unidad de trabajo se suprimen; los mensajes recuperados como parte de una unidad de trabajo se restablecen en la cola.

- En z/OS, esta llamada sólo la utilizan los programas por lotes (incluidos los programas IMS por lotes DL/I).
- En IBM i, esta llamada no está soportada para las aplicaciones que se ejecutan en modalidad de compatibilidad.

Sintaxis

MQBACK (*Hconn*, *CódigoComp*, *Razón*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* lo ha devuelto una llamada MQCONN o MQCONNX anterior.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

ERROR_ENTORNO_MQRC
(2012, X'7DC') La llamada no es válida en el entorno.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OUTCOME_MIXED
(2123, X'84B') El resultado de la operación de confirmación o de devolución se mezcla.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón

Notas de uso

1. Sólo puede utilizar esta llamada cuando el propio gestor de colas coordina la unidad de trabajo. Este puede ser:
   - Una unidad de trabajo local, donde los cambios sólo afectan a los recursos de MQ .
   - Unidad de trabajo global, donde los cambios pueden afectar a los recursos que pertenecen a otros gestores de recursos, así como a los recursos de MQ .

   Para obtener más detalles sobre las unidades de trabajo locales y globales, consulte “MQBEGIN- Iniciar unidad de trabajo” en la página 610.
2. En entornos en los que el gestor de colas no coordina la unidad de trabajo, utilice la llamada de restitución adecuada en lugar de MQBACK. El entorno también puede dar soporte a una devolución implícita causada por la terminación anómala de la aplicación.
   - En z/OS, utilice las llamadas siguientes:

- Los programas por lotes (incluidos los programas DL/I por lotes IMS) pueden utilizar la llamada MQBACK si la unidad de trabajo sólo afecta a los recursos de MQ . Sin embargo, si la unidad de trabajo afecta tanto a los recursos de MQ como a los recursos que pertenecen a otros gestores de recursos (por ejemplo, DB2), utilice la llamada SRRBACK proporcionada por el servicio de recursos recuperables (RRS) de z/OS . La llamada SRRBACK restituye los cambios en los recursos que pertenecen a los gestores de recursos que se han habilitado para la coordinación RRS.
- Las aplicaciones CICS deben utilizar el mandato `EXEC CICS SYNCPOINT ROLLBACK` para restituir la unidad de trabajo. No utilice la llamada MQBACK para aplicaciones CICS.
- Las aplicaciones IMS (que no sean programas DL/I por lotes) deben utilizar llamadas IMS como `ROLB` para restituir la unidad de trabajo. No utilice la llamada MQBACK para aplicaciones IMS (que no sean programas DL/I por lotes).

- En IBM i, utilice esta llamada para las unidades de trabajo locales coordinadas por el gestor de colas. Esto significa que no debe existir una definición de compromiso a nivel de trabajo, es decir, el mandato STRCMTCTL con el parámetro `CMTSCOPE(*JOB)` no debe haberse emitido para el trabajo.

3. Si una aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en "MQDISC-Desconectar gestor de colas" en la página 664 para obtener más detalles.
4. Cuando una aplicación coloca u obtiene mensajes en grupos o segmentos de mensajes lógicos, el gestor de colas conserva información relacionada con el grupo de mensajes y el mensaje lógico para las últimas llamadas MQPUT y MQGET satisfactorias. Esta información está asociada con el descriptor de contexto de cola e incluye cosas como:
   - Los valores de los campos *GroupId*, *MsgSeqNumber*, *Offset*y *MsgFlags* en MQMD.
   - Indica si el mensaje forma parte de una unidad de trabajo.
   - Para la llamada MQPUT: si el mensaje es persistente o no persistente.

   El gestor de colas mantiene *tres* conjuntos de información de grupo y segmento, un conjunto para cada uno de los siguientes:

   - La última llamada MQPUT satisfactoria (puede formar parte de una unidad de trabajo).
   - La última llamada MQGET satisfactoria que ha eliminado un mensaje de la cola (esto puede formar parte de una unidad de trabajo).
   - La última llamada MQGET satisfactoria que ha examinado un mensaje en la cola (esto *no puede* formar parte de una unidad de trabajo).

5. La información asociada con la llamada MQGET se restaura al valor que tenía antes de la primera llamada MQGET satisfactoria para ese manejador de cola en la unidad de trabajo actual.

   Las colas actualizadas por la aplicación después de que se iniciara la unidad de trabajo, pero fuera del ámbito de la unidad de trabajo, no tienen la información de grupo y segmento restaurada si se restituye la unidad de trabajo.

   La restauración de la información de grupo y segmento a su valor anterior cuando se restituye una unidad de trabajo permite a la aplicación distribuir un grupo de mensajes grande o un mensaje lógico grande que consta de muchos segmentos entre varias unidades de trabajo, y reiniciar en el punto correcto del grupo de mensajes o mensaje lógico si falla una de las unidades de trabajo.

   El uso de varias unidades de trabajo puede ser ventajoso si el gestor de colas local sólo tiene un almacenamiento de cola limitado. Sin embargo, la aplicación debe mantener suficiente información para poder reiniciar la colocación u obtención de mensajes en el punto correcto si se produce una anomalía del sistema.

   Para obtener detalles sobre cómo reiniciar en el punto correcto después de una anomalía del sistema, consulte la opción MQPMO_LOGICAL_ORDER descrita en "MQPMO-Opciones de transferencia de mensajes" en la página 475y la opción MQGMO_LOGICAL_ORDER descrita en "MQGMO-Opciones de obtención de mensajes" en la página 344.

   Las notas de uso restantes sólo se aplican cuando el gestor de colas coordina las unidades de trabajo.

6. Una unidad de trabajo tiene el mismo ámbito que un descriptor de conexión. Todas las llamadas de MQ que afectan a una unidad de trabajo determinada se deben realizar utilizando el mismo descriptor de conexión. Las llamadas emitidas utilizando un descriptor de conexión diferente (por ejemplo, las llamadas emitidas por otra aplicación) afectan a una unidad de trabajo diferente. Consulte el parámetro *Hconn* descrito en "MQCONN-Conectar gestor de colas" en la página 641 para obtener información sobre el ámbito de los manejadores de conexión.
7. Esta llamada sólo afecta a los mensajes que se han colocado o recuperado como parte de la unidad de trabajo actual.
8. Una aplicación de larga ejecución que emite llamadas MQGET, MQPUT o MQPUT1 dentro de una unidad de trabajo, pero que nunca emite una llamada de confirmación o restitución, puede llenar las colas con mensajes que no están disponibles para otras aplicaciones. Para protegerse contra esta posibilidad, el administrador debe establecer el atributo de gestor de colas *MaxUncommittedMsgs* en un valor lo suficientemente bajo como para evitar que las aplicaciones fuera de control llenen las colas, pero lo suficientemente alto como para permitir que las aplicaciones de mensajería esperadas funcionen correctamente.

Invocación en C

```
MQBACK (Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQBACK' USING HCONN, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQBACK (Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQBACK,(HCONN,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQBACK Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQBEGIN-Iniciar unidad de trabajo

La llamada MQBEGIN inicia una unidad de trabajo coordinada por el gestor de colas y que puede implicar a gestores de recursos externos.

Sintaxis

MQBEGIN (*Hconn*, *BeginOptions*, *Compcode*, *Reason*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* lo ha devuelto una llamada MQCONN o MQCONNX anterior.

Hconn debe ser un descriptor de conexión no compartido. Si se especifica un descriptor de conexión compartido, la llamada falla con el código de razón MQRC_HCONN_ERROR. Consulte la descripción de las opciones MQCNO_HANDLE_SHARE_ * en "MQCNO - Opciones de conexión" en la página 297 para obtener más información sobre los descriptores de contexto compartidos y no compartidos.

BeginOptions
Tipo: MQBO-entrada/salida

Estas son opciones que controlan la acción de MQBEGIN, tal como se describe en "MQBO-Opciones de inicio" en la página 259.

Si no se necesitan opciones, los programas escritos en C o S/390 assembler pueden especificar una dirección de parámetro nula, en lugar de especificar la dirección de una estructura MQBO.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_WARNING:

MQRC_NO_EXTERNAL_PARTICIPANTES
(2121, X'849 ') No se ha registrado ningún gestor de recursos participante.

MQRC_PARTICIPANT_NOT_AVAILABLE
(2122, X'84A') El gestor de recursos participante no está disponible.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_BO_ERROR
(2134, X'856 ') Estructura de opciones de inicio no válida.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

ERROR_ENTORNO_MQRC
(2012, X'7DC') La llamada no es válida en el entorno.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_IN_PROGRESS
(2128, X'850 ') Unidad de trabajo ya iniciada.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón.

Notas de uso

1. Utilice la llamada MQBEGIN para iniciar una unidad de trabajo coordinada por el gestor de colas y que puede implicar cambios en los recursos propiedad de otros gestores de recursos. El gestor de colas da soporte a tres tipos de unidad de trabajo:
   - **Unidad de trabajo local coordinada por el gestor de colas**: unidad de trabajo en la que el gestor de colas es el único gestor de recursos que participa y, por lo tanto, el gestor de colas actúa como coordinador de unidad de trabajo.
     - Para iniciar este tipo de unidad de trabajo, especifique la opción MQPMO_SYNCPOINT o MQGMO_SYNCPOINT en la primera llamada MQPUT, MQPUT1o MQGET en la unidad de trabajo.
     - Para confirmar o restituir este tipo de unidad de trabajo, utilice la llamada MQCMIT o MQBACK.
   - **Unidad de trabajo global coordinada por el gestor de colas**: una unidad de trabajo en la que el gestor de colas actúa como coordinador de unidad de trabajo, ambos para MQ recursos *y* para recursos que pertenecen a otros gestores de recursos. Estos gestores de recursos cooperan con el

gestor de colas para asegurarse de que todos los cambios en los recursos de la unidad de trabajo se confirman o se restituyen juntos.

  - Para iniciar este tipo de unidad de trabajo, utilice la llamada MQBEGIN.
  - Para confirmar o restituir este tipo de unidad de trabajo, utilice las llamadas MQCMIT y MQBACK.
- **Unidad de trabajo global coordinada externamente**: una unidad de trabajo en la que el gestor de colas es un participante, pero el gestor de colas no actúa como coordinador de unidad de trabajo. En su lugar, hay un coordinador de unidad de trabajo externo con el que el gestor de colas coopera.
  - Para iniciar este tipo de unidad de trabajo, utilice la llamada pertinente proporcionada por el coordinador externo de la unidad de trabajo.

    Si se utiliza la llamada MQBEGIN para intentar iniciar la unidad de trabajo, la llamada falla con el código de razón MQRC_ENVIRONMENT_ERROR.
  - Para confirmar o restituir este tipo de unidad de trabajo, utilice las llamadas de confirmación y devolución proporcionadas por el coordinador de unidad de trabajo externo.

    Si utiliza la llamada MQCMIT o MQBACK para confirmar o restituir la unidad de trabajo, la llamada falla con el código de razón MQRC_ENVIRONMENT_ERROR.

2. Si la aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso en "MQDISC-Desconectar gestor de colas" en la página 664 para obtener más detalles.
3. Una aplicación sólo puede participar en una unidad de trabajo a la vez. La llamada MQBEGIN falla con el código de razón MQRC_UOW_IN_PROGRESS si ya existe una unidad de trabajo para la aplicación, independientemente del tipo de unidad de trabajo que sea.
4. La llamada MQBEGIN no es válida en un entorno de cliente MQI de MQ . Un intento de utilizar la llamada falla con el código de razón MQRC_ENVIRONMENT_ERROR.
5. Cuando el gestor de colas actúa como coordinador de unidad de trabajo para unidades de trabajo globales, los gestores de recursos que pueden participar en la unidad de trabajo se definen en el archivo de configuración del gestor de colas.
6. En IBM i, los tres tipos de unidad de trabajo están soportados de la siguiente manera:
   - **Unidad de trabajo local coordinada por el gestor de colas** sólo se puede utilizar cuando no existe una definición de compromiso a nivel de trabajo, es decir, el mandato STRCMTCTL con el parámetro CMTSCOPE(*JOB) no se debe haber emitido para el trabajo.
   - **Unidad de trabajo global coordinada por el gestor de colas** no está soportada.
   - **Unidad de trabajo global coordinada externamente** sólo se puede utilizar cuando existe una definición de compromiso a nivel de trabajo, es decir, el mandato STRCMTCTL con el parámetro CMTSCOPE(*JOB) debe haberse emitido para el trabajo. Si esto se ha hecho, las operaciones IBM i COMMIT y ROLLBACK se aplican a los recursos de MQ , así como a los recursos que pertenecen a otros gestores de recursos participantes.

Invocación en C

```
MQBEGIN (Hconn, &BeginOptions, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;         /* Connection handle */
MQBO     BeginOptions;  /* Options that control the action of MQBEGIN */
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
     CALL 'MQBEGIN' USING HCONN, BEGINOPTIONS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQBEGIN (Hconn, BeginOptions, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en Visual Basic

```
MQBEGIN Hconn, BeginOptions, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
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 - Convertir almacenamiento intermedio en descriptor de contexto de mensaje

La llamada de función MQBUFMH convierte un almacenamiento intermedio en un manejador de mensajes y es el inverso de la llamada MQMHBUF.

Esta llamada toma un descriptor de mensaje y las propiedades MQRFH2 del almacenamiento intermedio y las hace disponibles a través de un descriptor de mensaje. Las propiedades MQRFH2 de los datos del mensaje se eliminan, opcionalmente. Los campos *Encoding*, *CodedCharSetId*y *Format* del descriptor de mensaje se actualizan, si es necesario, para describir correctamente el contenido del almacenamiento intermedio después de que se hayan eliminado las propiedades.

Sintaxis

MQBUFMH (*Hconn*, *Hmsg*, *BufMsgHOpts* , *MsgDesc*, *Buffer*, *BufferLength*, *DataLength*, *Compcode* , *Reason*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro *Hmsg* .

Si el descriptor de mensaje se ha creado utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en la hebra convirtiendo un almacenamiento intermedio en un descriptor de mensaje. Si no se establece una conexión válida, la llamada falla con MQRC_CONNECTION_BROKEN.

Msj
Tipo: MQHMQSG-entrada

Este es el descriptor de mensaje para el que se necesita un almacenamiento intermedio. El valor ha sido devuelto por una llamada MQCRTMH anterior.

BufMsgHOpts
Tipo: MQBMHO-entrada

La estructura MQBMHO permite a las aplicaciones especificar opciones que controlan cómo se generan los manejadores de mensajes a partir de los almacenamientos intermedios.

Para obtener más detalles, consulte "MQBMHO-Opciones de almacenamiento intermedio a manejador de mensajes" en la página 257.

MsgDesc
Tipo: MQMD - entrada/salida

La estructura *MsgDesc* contiene las propiedades del descriptor de mensaje y describe el contenido del área de almacenamiento intermedio.

En la salida de la llamada, las propiedades se eliminan opcionalmente del área de almacenamiento intermedio y, en este caso, el descriptor de mensaje se actualiza para describir correctamente el área de almacenamiento intermedio.

Los datos de esta estructura deben estar en el juego de caracteres y la codificación de la aplicación.

BufferLength
Tipo: MQLONG - entrada

BufferLength es la longitud del área de almacenamiento intermedio, en bytes.

Un *BufferLength* de cero bytes es válido e indica que el área de almacenamiento intermedio no contiene datos.

Buffer
Tipo: MQBYTExBufferLongitud-entrada/salida

Estas son opciones que controlan la acción de MQBEGIN, tal como se describe en "MQBEGIN-Iniciar unidad de trabajo" en la página 610.

Buffer define el área que contiene el almacenamiento intermedio de mensajes. Para la mayoría de los datos, debe alinear el almacenamiento intermedio en un límite de 4 bytes.

Si *Buffer* contiene datos numéricos o de caracteres, establezca los campos *CodedCharSetId* y *Encoding* del parámetro *MsgDesc* en los valores adecuados para los datos; esto permite convertir los datos, si es necesario.

Si se encuentran propiedades en el almacenamiento intermedio de mensajes, se eliminan de forma opcional; posteriormente pasan a estar disponibles desde el descriptor de contexto de mensaje al devolver la llamada.

En el lenguaje de programación C, el parámetro se declara como un puntero a void, lo que significa que la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si el parámetro *BufferLength* es cero, no se hace referencia a *Buffer* ; en este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler puede ser nula.

DataLength
Tipo: MQLONG - salida

Longitud, en bytes, del almacenamiento intermedio que puede tener las propiedades eliminadas.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BMHO_ERROR
(2489, X'09B9') La estructura de opciones de almacenamiento intermedio a manejador de mensajes no es válida.

MQRC_BUFFER_ERROR
(2004, X'07D4') El parámetro de almacenamiento intermedio no es válido.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_HMSG_ERROR
(2460, X'099C') Descriptor de mensaje no válido.

MQRC_MD_ERROR
(2026, X'07EA') Descriptor de mensaje no válido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Opciones no válidas o no coherentes.

MQRC_RFH_ERROR
(2334, X'091E') La estructura MQRFH2 no es válida.

MQRC_RFH_FORMAT_ERROR
(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso

Las llamadas MQBUFMH no se pueden interceptar mediante salidas de API: un almacenamiento intermedio se convierte en un manejador de mensajes en el espacio de aplicación; la llamada no llega al gestor de colas.

Invocación en C

```
MQBUFMH (Hconn, Hmsg, &BufMsgHOpts, &MsgDesc, BufferLength, Buffer,
         &DataLength, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
CALL 'MQBUFMH' USING HCONN, HMSG, BUFMSGHOPTS, MSGDESC, BUFFERLENGTH,
                           BUFFER, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQBUFMH (Hconn, Hmsg, BufMsgHOpts, MsgDesc, BufferLength, Buffer,
              DataLength, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
CALL MQBUFMH,(HCONN,HMSG,BUFMSGHOPTS,MSGDESC,BUFFERLENGTH,BUFFER,
               DATALENGTH,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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-Gestionar devolución de llamada

La llamada MQCB registra una devolución de llamada para el descriptor de objeto especificado y controla la activación y los cambios en la devolución de llamada.

Una devolución de llamada es un fragmento de código (especificado como el nombre de una función que se puede enlazar dinámicamente o como puntero de función) al que llama IBM WebSphere MQ cuando se producen determinados sucesos.

Para utilizar MQCB y MQCTL en un cliente V7 debe estar conectado a un servidor V7 y el parámetro **SHARECNV** del canal debe tener un valor distinto de cero.

Los tipos de devolución de llamada que se pueden definir son:

Consumidor de mensajes
Se llama a una función de devolución de llamada de consumidor de mensajes cuando un mensaje, que cumple los criterios de selección especificados, está disponible en un descriptor de contexto de objeto.

Sólo se puede registrar una función de devolución de llamada para cada descriptor de contexto de objeto. Si se va a leer una sola cola con varios criterios de selección, la cola se debe abrir varias veces y se debe registrar una función de consumidor en cada descriptor de contexto.

Manejador de sucesos
Se llama al manejador de sucesos para condiciones que afectan a todo el entorno de devolución de llamada.

Se llama a la función cuando se produce una condición de suceso, por ejemplo, un gestor de colas o una conexión que se detiene o se desactiva temporalmente.

La función no se invoca para condiciones específicas de un único consumidor de mensajes, por ejemplo, MQRC_GET_INHIBITED; sin embargo, se llama si una función de devolución de llamada no finaliza normalmente.

Sintaxis

MQCB (*Hconn*, *Operación*, *CallbackDesc*, *Hobj*, *MsgDesc* , *GetMsgOpts*, *CompCode*, *Razón*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, puede especificar el siguiente valor especial para que *MQHC_DEF_HCONN* utilice el descriptor de conexión asociado a esta unidad de ejecución.

Operación
Tipo: MQLONG - entrada

La operación que se está procesando en la devolución de llamada definida para el descriptor de objeto especificado. Debe especificar una de las opciones siguientes; si se necesita más de una opción, los valores pueden ser:

- Añadido (no añadir la misma constante más de una vez), o
- Combinado utilizando la operación OR bit a bit (si el lenguaje de programación da soporte a operaciones de bits).

MQOP_REGISTRO
Defina la función de devolución de llamada para el descriptor de objeto especificado. Esta operación define la función que se va a llamar y los criterios de selección que se van a utilizar.

Si ya se ha definido una función de devolución de llamada para el descriptor de contexto de objeto, se sustituye la definición. Si se detecta un error al sustituir la devolución de llamada, se anula el registro de la función.

Si una devolución de llamada se registra en la misma función de devolución de llamada en la que se ha anulado el registro anteriormente, se trata como una operación de sustitución; no se invoca ninguna llamada inicial o final.

Puede utilizar MQOP_REGISTER con MQOP_SUSPEND o MQOP_RESUME.

MQOP_DEREGISTER
Detiene el consumo de mensajes para el descriptor de contexto de objeto y elimina el descriptor de contexto de los elegibles para una devolución de llamada.

Una devolución de llamada se anula automáticamente si se cierra el descriptor de contexto asociado.

Si se llama a MQOP_DEREGISTER desde dentro de un consumidor, y la devolución de llamada tiene definida una llamada de detención, se invoca tras la devolución del consumidor.

Si esta operación se emite en un *Hobj* sin ningún consumidor registrado, la llamada se devuelve con MQRC_CALLBACK_NOT_REGISTERED.

MQOP_SUSPENDER
Suspende el consumo de mensajes para el descriptor de objeto.

Si esta operación se aplica a un manejador de sucesos, el manejador de sucesos no obtiene sucesos mientras está suspendido y los sucesos que faltan mientras está en estado suspendido no se proporcionan a la operación cuando se reanuda.

Mientras está suspendida, la función de consumidor continúa obteniendo las devoluciones de llamada de tipo de control.

MQOP_RESUME
Reanude el consumo de mensajes para el descriptor de objeto.

Si esta operación se aplica a un manejador de sucesos, el manejador de sucesos no obtiene sucesos mientras está suspendido y los sucesos que faltan mientras está en estado suspendido no se proporcionan a la operación cuando se reanuda.

CallbackDesc
Tipo: MQCBD-entrada

Esta es una estructura que identifica la función de devolución de llamada que está registrando la aplicación y las opciones utilizadas al registrarla.

Consulte MQCBD para obtener detalles de la estructura.

El descriptor de devolución de llamada sólo es necesario para la opción MQOP_REGISTER; si el descriptor no es necesario, la dirección de parámetro pasada puede ser nula.

Hobj
Tipo: MQHOBJ - entrada

Este descriptor de contexto representa el acceso que se ha establecido al objeto desde el que se va a consumir un mensaje. Es un descriptor de contexto que se ha devuelto desde una llamada MQOPEN o MQSUB anterior (en el parámetro *Hobj*).

Hobj no es necesario al definir una rutina de manejador de sucesos (MQCBT_EVENT_HANDLER) y debe especificarse como MQHO_NONE.

Si se ha devuelto *Hobj* desde una llamada MQOPEN, la cola debe haberse abierto con una o varias de las opciones siguientes:

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE

MsgDesc
Tipo: MQMD-entrada

Esta estructura describe los atributos del mensaje necesario y los del mensaje recuperado.

El parámetro *MsgDesc* define los atributos de los mensajes que necesita el consumidor y la versión del MQMD que se va a pasar al consumidor de mensajes.

MsgId, *CorrelId*, *GroupId* , *MsgSeqNumber* y *Offset* en MQMD se utilizan para la selección de mensajes, en función de las opciones especificadas en el parámetro *GetMsgOpts* .

Encoding y *CodedCharSetId* se utilizan para la conversión de mensajes si especifica la opción MQGMO_CONVERT.

Consulte MQMD para obtener más detalles.

MsgDesc se utiliza para MQOP_REGISTER y si necesita valores distintos del valor predeterminado para cualquier campo. *MsgDesc* no se utiliza para un manejador de sucesos.

Si el descriptor no es necesario, la dirección de parámetro pasada puede ser nula.

Tenga en cuenta que si se registran varios consumidores en la misma cola con selectores solapados, el consumidor elegido para cada mensaje no está definido.

GetMsgOpts
Tipo: MQGMO-entrada

El parámetro *GetMsgOpts* controla cómo obtiene los mensajes el consumidor de mensajes. Todas las opciones de este parámetro tienen significados tal como se describe en "MQGMO-Opciones de obtención de mensajes" en la página 344, cuando se utilizan en una llamada MQGET, excepto:

MQGMO_SET_SIGNAL
Esta opción no está permitida.

MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT, MQGMO_MARK_ *
El orden de los mensajes entregados a un consumidor de navegación viene determinado por las combinaciones de estas opciones. Las combinaciones significativas son:

MQGMO_BROWSE_FIRST
El primer mensaje de la cola se entrega repetidamente al consumidor. Esto es útil cuando el consumidor consume de forma destructiva el mensaje en la devolución de llamada. Utilice esta opción con cuidado.

MQGMO_BROWSE_NEXT
Al consumidor se le asigna cada mensaje de la cola, desde la posición actual del cursor hasta que se alcanza el final de la cola.

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT
El cursor se restablece al inicio de la cola. A continuación, se proporciona al consumidor cada mensaje hasta que el cursor llega al final de la cola.

MQGMO_BROWSE_FIRST + MQGMO_MARK_ *
A partir del principio de la cola, al consumidor se le proporciona el primer mensaje no marcado en la cola, que a continuación se marca para este consumidor. Esta combinación garantiza que el consumidor pueda recibir nuevos mensajes añadidos detrás del punto de cursor actual.

MQGMO_BROWSE_NEXT + MQGMO_MARK_ *
A partir de la posición del cursor, al consumidor se le proporciona el siguiente mensaje no marcado en la cola, que a continuación se marca para este consumidor. Utilice esta combinación con cuidado porque los mensajes se pueden añadir a la cola detrás de la posición actual del cursor.

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT + MQGMO_MARK_ *
Esta combinación no está permitida. Si se utiliza, la llamada devuelve MQRC_OPTIONS_ERROR.

MQGMO_NO_WAIT, MQGMO_WAIT y WaitInterval
Estas opciones controlan cómo se invoca al consumidor.

MQGMO_NO_WAIT
Nunca se llama al consumidor con MQRC_NO_MSG_AVAILABLE. Sólo se llama al consumidor para mensajes y sucesos.

MQGMO_WAIT con un WaitInterval cero
El código MQRC_NO_MSG_AVAILABLE se pasa al consumidor cuando no hay mensajes disponibles y el consumidor se ha iniciado o se ha entregado al menos un mensaje desde el último código de razón "sin mensajes".

Esto impide que el consumidor sondee en un bucle ocupado cuando se especifica un intervalo de espera cero.

MQGMO_WAIT y un WaitInterval positivo
Se llama al consumidor después del intervalo de espera especificado con el código de razón MQRC_NO_MSG_AVAILABLE. Esta llamada se realiza independientemente de si se ha entregado algún mensaje al consumidor. Esto permite al usuario realizar un proceso de latido o de tipo de proceso por lotes.

MQGMO_WAIT y WaitInterval de MQWI_UNLIMITED
Especifica una espera infinita antes de devolver MQRC_NO_MSG_AVAILABLE. Nunca se llama al consumidor con MQRC_NO_MSG_AVAILABLE.

GetMsgOpts sólo se utiliza para MQOP_REGISTER y si necesita valores distintos del valor predeterminado para cualquier campo. *GetMsgOpts* no se utiliza para un manejador de sucesos.

Si los *GetMsgOpts* no son necesarios, la dirección de parámetro pasada puede ser nula. El uso de este parámetro es el mismo que especificar MQGMO_DEFAULT junto con MQGMO_FAIL_IF_QUIESCING.

Si se proporciona un descriptor de contexto de propiedades de mensaje en la estructura MQGMO, se proporciona una copia en la estructura MQGMO que se pasa a la devolución de llamada del

consumidor. Al volver de la llamada MQCB, la aplicación puede suprimir el descriptor de contexto de propiedades de mensaje.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Los códigos de razón de la lista siguiente son los que el gestor de colas puede devolver para el parámetro *Reason* .

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

nMQRC_ADAPTER_CONV_LOAD_ERROR
(2133, X'855') No se han podido cargar módulos de servicio de conversión de datos.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CALLBACK_LINK_ERROR
(2487, X'9B7') Campo de tipo de devolución de llamada incorrecto.

MQRC_CALLBACK_NOT_REGISTERED
(2448, X' 990 ') No se puede anular el registro, suspender o reanudar porque no hay ninguna devolución de llamada registrada.

MQRC_CALLBACK_ROUTINE_ERROR
(2486, X'9B6') Se debe especificar *CallbackFunction* o *CallbackName* , pero no ambos.

MQRC_CALLBACK_TYPE_ERROR
(2483, X'9B3') Campo de tipo de devolución de llamada incorrecto.

MQRC_CBD_OPTIONS_ERROR
(2484, X'9B4') Campo de opciones MQCBD incorrecto.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') CICS ha rechazado la solicitud de espera.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_CORREL_ID_ERROR
(2207, X'89F') Error de identificador de correlación.

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_FUNCTION_NOT_SUPPORTED
(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

MQRC_GET_INHIBITED
(2016, X'7E0') Se han inhibido las obtenciones para la cola.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GMO_ERROR
(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR
(2019, X'7E3') El manejador de objeto no es válido.

MQRC_INCONSISTENT_BROWSE
(2259, X'8D3') La especificación de examinar es incoherente.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_INVALID_MSG_UNDER_CURSOR
(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') Las opciones de coincidencia no son válidas.

MQRC_MAX_MSG_LENGTH_ERROR
(2485, X'9B4') Campo *MaxMsgLength* incorrecto.

MQRC_MD_ERROR
(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MODULE_ENTRY_NOT_FOUND
(2497, X'9C1') No se ha podido encontrar el punto de entrada de función especificado en el módulo.

MQRC_MODULE_INVALID
(2496, X'9C0') Se ha encontrado el módulo, pero es del tipo incorrecto; no de 32 bits, 64 bits o una biblioteca de enlace dinámico válida.

MQRC_MODULE_NOT_FOUND
(2495, X'9BF') El módulo no se ha encontrado en la vía de acceso de búsqueda o no tiene autorización para cargarse.

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') El uso del símbolo de mensaje no es válido.

MQRC_NO_MSG_AVAILABLE
(2033, X'7F1') No hay ningún mensaje disponible.

MQRC_NO_MSG_UNDER_CURSOR
(2034, X'7F2') El cursor para examinar no está situado en el mensaje.

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') La cola no se ha abierto para examen.

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') La cola no se ha abierto para entrada.

MQRC_OBJECT_CHANGED
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OPERATION_ERROR
(2206, X'89E') Código de operación incorrecto en llamada de API.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_DELETED
(2052, X'804') La cola se ha suprimido.

MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') La cola tiene un tipo de índice incorrecto.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SIGNAL_OUTSTANDING
(2069, X'815') Símbolo pendiente para este descriptor de contexto.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818') El punto de sincronización de soporte no está disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WAIT_INTERVAL_ERROR
(2090, X'82A') El intervalo de espera de MQGMO no es válido.

MQRC_WRONG_GMO_VERSION
(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso

1. MQCB se utiliza para definir la acción que se va a invocar para cada mensaje, que coincide con los criterios especificados, disponibles en la cola. Cuando se procesa la acción, el mensaje se elimina de la cola y se pasa al consumidor de mensajes definido, o se proporciona una señal de mensaje, que se utiliza para recuperar el mensaje.
2. MQCB se puede utilizar para definir rutinas de devolución de llamada antes de iniciar el consumo con MQCTL o se puede utilizar desde dentro de una rutina de devolución de llamada.
3. Para utilizar MQCB desde fuera de una rutina de devolución de llamada, primero debe suspender el consumo de mensajes utilizando MQCTL y reanudar el consumo posteriormente.
4. MQCB no está soportado en el adaptador IMS .

Secuencia de devolución de llamada de consumidor de mensajes

Puede configurar un consumidor para invocar la devolución de llamada en puntos clave durante el ciclo de vida del consumidor. Por ejemplo:

- cuando el consumidor esté registrado por primera vez,
- cuando se inicia la conexión,
- cuando se detiene la conexión y
- cuando se anula el registro del consumidor, ya sea explícita o implícitamente mediante un MQCLOSE.

Tabla 562. Definiciones de verbo MQCTL

Verbo	**Significado**
MQCTL (START)	Llamada MQCTL utilizando la operación MQOP_START
MQCTL (STOP)	Llamada MQCTL utilizando la operación MQOP_STOP
MQCTL (ESPERAR)	Llamada MQCTL utilizando la operación MQOP_START_WAIT

Esto permite al consumidor mantener el estado asociado con el consumidor. Cuando una aplicación solicita una devolución de llamada, las reglas para la invocación de consumidor son las siguientes:

REGISTRAR
Es siempre el primer tipo de invocación de la devolución de llamada.

Siempre se llama en la misma hebra, como la llamada MQCB (REGISTER).

START
Siempre se llama de forma síncrona con el verbo MQCTL (START).

- Todas las devoluciones de llamada START se completan antes de que se devuelva el verbo MQCTL (START).

Está en la misma hebra que la entrega de mensajes si se solicita THREAD_AFFINITY.

La llamada con inicio no se garantiza si, por ejemplo, una devolución de llamada anterior emite MQCTL (STOP) durante MQCTL (START).

STOP
No se entregan más mensajes o sucesos después de esta llamada hasta que se reinicia la conexión.

Se garantiza un STOP si la aplicación se ha llamado anteriormente para START, o un mensaje, o un suceso.

DEREGISTER
Es siempre el último tipo de invocación de la devolución de llamada.

Asegúrese de que la aplicación realiza la inicialización y limpieza basadas en hebras en las devoluciones de llamada START y STOP. Puede realizar una inicialización y limpieza no basadas en hebras con devoluciones de llamada REGISTER y DEREGISTER.

No haga ninguna suposición sobre la vida y la disponibilidad del hilo aparte de lo que se indica. Por ejemplo, no confíe en que una hebra permanezca activa más allá de la última llamada a DEREGISTER. De forma similar, cuando haya elegido no utilizar THREAD_AFFINITY, no presuponga que la hebra existe siempre que se inicie la conexión.

Si la aplicación tiene requisitos específicos para las características de hebra, siempre puede crear una hebra en consecuencia y, a continuación, utilizar MQCTL (WAIT). Esto tiene el efecto de 'donar' la hebra a IBM WebSphere MQ para la entrega de mensajes asíncronos.

Uso de conexión de consumidor de mensajes

Puede configurar un consumidor para invocar la devolución de llamada en puntos clave durante el ciclo de vida del consumidor. Por ejemplo:

- cuando el consumidor esté registrado por primera vez,
- cuando se inicia la conexión,
- cuando se detiene la conexión y
- cuando se anula el registro del consumidor, ya sea explícita o implícitamente mediante un MQCLOSE.

Tabla 563. Definiciones de verbo MQCTL

Verbo	**Significado**
MQCTL (START)	Llamada MQCTL utilizando la operación MQOP_START
MQCTL (STOP)	Llamada MQCTL utilizando la operación MQOP_STOP
MQCTL (ESPERAR)	Llamada MQCTL utilizando la operación MQOP_START_WAIT

Esto permite al consumidor mantener el estado asociado con el consumidor. Cuando una aplicación solicita una devolución de llamada, las reglas para la invocación de consumidor son las siguientes:

REGISTRAR
Es siempre el primer tipo de invocación de la devolución de llamada.

Siempre se llama en la misma hebra, como la llamada MQCB (REGISTER).

START
Siempre se llama de forma síncrona con el verbo MQCTL (START).

- Todas las devoluciones de llamada START se completan antes de que se devuelva el verbo MQCTL (START).

Está en la misma hebra que la entrega de mensajes si se solicita THREAD_AFFINITY.

La llamada con inicio no se garantiza si, por ejemplo, una devolución de llamada anterior emite MQCTL (STOP) durante MQCTL (START).

STOP
No se entregan más mensajes o sucesos después de esta llamada hasta que se reinicia la conexión.

Se garantiza un STOP si la aplicación se ha llamado anteriormente para START, o un mensaje, o un suceso.

DEREGISTER
Es siempre el último tipo de invocación de la devolución de llamada.

Asegúrese de que la aplicación realiza la inicialización y limpieza basadas en hebras en las devoluciones de llamada START y STOP. Puede realizar una inicialización y limpieza no basadas en hebras con devoluciones de llamada REGISTER y DEREGISTER.

No haga ninguna suposición sobre la vida y la disponibilidad del hilo aparte de lo que se indica. Por ejemplo, no confíe en que una hebra permanezca activa más allá de la última llamada a DEREGISTER. De forma similar, cuando haya elegido no utilizar THREAD_AFFINITY, no presuponga que la hebra existe siempre que se inicie la conexión.

Si la aplicación tiene requisitos específicos para las características de hebra, siempre puede crear una hebra en consecuencia y, a continuación, utilizar MQCTL (WAIT). Esto tiene el efecto de 'donar' la hebra a IBM WebSphere MQ para la entrega de mensajes asíncronos.

Invocación en C

```
MQCB (Hconn, Operation, CallbackDesc, Hobj, MsgDesc,
GetMsgOpts, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
CALL 'MQCB' USING HCONN, OPERATION, CBDESC, HOBJ, MSGDESC,
            GETMSGOPTS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQCB(Hconn, Operation, CallbackDesc, Hobj, MsgDesc, GetMsgOpts,
          CompCode, Reason)
```

Declare los parámetros como se indica a continuación:

```
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-Función de devolución de llamada

La llamada a la función MQCB_FUNCTION es la función de devolución de llamada para el manejo de sucesos y el consumo de mensajes asíncronos.

La definición de llamada MQCB_FUNCTION se proporciona únicamente para describir los parámetros que se pasan a la función de devolución de llamada. El gestor de colas no proporciona ningún punto de entrada denominado MQCB_FUNCTION.

La especificación de la función real que se va a llamar es una entrada a la llamada MQCB y se pasa a través de la estructura MQCBD .

Sintaxis

MQCB_FUNCTION (*Hconn*, *MsgDesc*, *GetMsgOpts*, *Buffer*, *Contexto*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* lo ha devuelto una llamada MQCONN o MQCONNX anterior. En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para Hconn:

MQHC_DEF_CONN

Manejador de conexión predeterminado.

MsgDesc
Tipo: MQMD-entrada

Esta estructura describe los atributos del mensaje recuperado.

Para obtener detalles, consulte “MQMD - Descriptor de mensaje” en la página 392.

La versión de MQMD pasada es la misma versión que la pasada en la llamada MQCB que ha definido la función de consumidor.

La dirección del MQMD se pasa como caracteres nulos si se ha utilizado un MQGMO de la versión 4 para solicitar que se devuelva un manejador de mensajes en lugar de un MQMD.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de sucesos.

GetMsgOpts
Tipo: MQGMO-entrada

Opciones utilizadas para controlar las acciones del consumidor de mensajes. Este parámetro también contiene información adicional sobre el mensaje devuelto.

Consulte MQGMO para obtener más detalles.

La versión de MQGMO pasada es la última versión soportada.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de sucesos.

Buffer
Tipo: MQBYTExBufferLongitud-entrada

Es el área que contiene los datos del mensaje.

Si no hay ningún mensaje disponible para esta llamada, o si el mensaje no contiene datos de mensaje, la dirección del *Buffer* se pasa como nulos.

Es un campo de entrada para la función de consumidor de mensajes; no es relevante para una función de manejador de sucesos.

Contexto
Tipo: MQCBC-entrada/salida

Esta estructura proporciona información de contexto a las funciones de devolución de llamada. Para obtener detalles, consulte "MQCBC-Contexto de devolución de llamada" en la página 261.

Notas de uso

1. Tenga en cuenta que si las rutinas de devolución de llamada utilizan servicios que podrían retrasar o bloquear la hebra, por ejemplo, MQGET con espera, podría retrasar la asignación de otras devoluciones de llamada.
2. No se establece automáticamente una unidad de trabajo separada para cada invocación de una rutina de devolución de llamada, por lo que las rutinas pueden emitir una llamada de confirmación, o aplazar la confirmación, hasta que se haya procesado un lote lógico de trabajo. Cuando se confirma el lote de trabajo, confirma los mensajes para todas las funciones de devolución de llamada que se han invocado desde el último punto de sincronización.
3. Los programas invocados por CICS LINK o CICS START recuperan parámetros utilizando servicios CICS a través de objetos con nombre conocidos como contenedores de canal. Los nombres de contenedor son los mismos que los nombres de parámetro. Para obtener más información, consulte la documentación de CICS .
4. Las rutinas de devolución de llamada pueden emitir una llamada MQDISC, pero no para su propia conexión. Por ejemplo, si una rutina de devolución de llamada ha creado una conexión, también puede desconectar la conexión.
5. Una rutina de devolución de llamada no debe, en general, basarse en que se invoque desde la misma hebra cada vez. Si es necesario, utilice MQCTLO_THREAD_AFFINITY cuando se inicie la conexión.
6. Cuando una rutina de devolución de llamada recibe un código de razón distinto de cero, debe realizar la acción adecuada.
7. MQCB_FUNCTION no está soportado en el adaptador IMS .

MQCLOSE-Cerrar objeto

La llamada MQCLOSE renuncia al acceso a un objeto y es la inversa de las llamadas MQOPEN y MQSUB.

Sintaxis

MQCLOSE (*Hconn*, *Hobj*, *Options*, *CompCode*, *Razón*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* lo ha devuelto una llamada MQCONN o MQCONNX anterior.

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, puede omitir la llamada MQCONN y especificar el valor siguiente para *Hconn*:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Hobj
Tipo: MQHOBJ - entrada/salida

Este descriptor de contexto representa el objeto que se está cerrando. El objeto puede ser de cualquier tipo. El valor de *Hobj* lo ha devuelto una llamada MQOPEN anterior.

Al finalizar correctamente la llamada, el gestor de colas establece este parámetro en un valor que no es un descriptor de contexto válido para el entorno. Este valor es:

MQHO_UNUSABLE_HOBJ
Descriptor de objeto inutilizable.

En z/OS, *Hobj* se establece en un valor que no está definido.

opciones
Tipo: MQLONG - entrada

Este parámetro controla cómo se cierra el objeto.

Sólo las colas dinámicas permanentes y las suscripciones se pueden cerrar de más de una forma, porque se deben retener o suprimir; estas son colas con el atributo *DefinitionType* que tiene el valor MQQDT_PERMANENT_DYNAMIC (consulte el atributo *DefinitionType* descrito en "Atributos para colas" en la página 816). Las opciones de cierre se resumen en este tema.

Las suscripciones duraderas se pueden conservar o eliminar; estas se crean utilizando la llamada MQSUB con la opción MQSO_DURABLE.

Al cerrar el descriptor de contexto en un destino gestionado (es decir, el parámetro *Hobj* devuelto en una llamada MQSUB que ha utilizado la opción MQSO_MANAGED), el gestor de colas limpia las publicaciones que no se han recuperado cuando también se ha eliminado la suscripción asociada. La suscripción se elimina utilizando la opción MQCO_REMOVE_SUB en el parámetro *Hsub* devuelto en una llamada MQSUB. Tenga en cuenta que MQCO_REMOVE_SUB es el comportamiento predeterminado en MQCLOSE para una suscripción no duradera.

Al cerrar un descriptor de contexto en un destino no gestionado, es responsable de limpiar la cola donde se envían las publicaciones. Cierre primero la suscripción utilizando MQCO_REMOVE_SUB y, a continuación, procese los mensajes fuera de la cola hasta que no quede ninguno.

Sólo debe especificar una opción entre las siguientes:

Opciones de cola dinámica: estas opciones controlan cómo se cierran las colas dinámicas permanentes.

MQCO_DELETE
La cola se suprime si se cumple alguna de las condiciones siguientes:

- Es una cola dinámica permanente, creada por una llamada MQOPEN anterior, y no hay mensajes en la cola y no hay solicitudes de obtención o colocación no confirmadas pendientes para la cola (para la tarea actual o para cualquier otra tarea).

- Es la cola dinámica temporal que ha creado la llamada MQOPEN que ha devuelto *Hobj*. En este caso, se depuran todos los mensajes de la cola.

En todos los demás casos, incluido el caso en el que se ha devuelto *Hobj* en una llamada MQSUB, la llamada falla con el código de razón MQRC_OPTION_NOT_VALID_FOR_TYPE, y el objeto no se suprime.

En z/OS, si la cola es una cola dinámica que se ha suprimido lógicamente, y este es el último descriptor de contexto para ella, la cola se suprime físicamente. Para conocer detalles, consulte "Notas de uso" en la página 634.

MQCO_DELETE_PURGE
La cola se suprime y los mensajes que contiene se depuran, si se cumple alguna de las condiciones siguientes:

- Es una cola dinámica permanente, creada por una llamada MQOPEN anterior, y no hay solicitudes get o put no confirmadas pendientes para la cola (ni para la tarea actual ni para ninguna otra tarea).
- Es la cola dinámica temporal que ha creado la llamada MQOPEN que ha devuelto *Hobj*.

En todos los demás casos, incluido el caso en el que se ha devuelto *Hobj* en una llamada MQSUB, la llamada falla con el código de razón MQRC_OPTION_NOT_VALID_FOR_TYPE, y el objeto no se suprime.

La tabla muestra qué opciones de cierre son válidas y si el objeto se conserva o se suprime.

Tipo de objeto o cola	**MQCO_NONE**	**MQCO_DELETE**	**MQCO_DELETE_PURGE**
Objeto que no es una cola	Retenido	No válido	No válido
Cola predefinida	Retenido	No válido	No válido
cola dinámica permanente	Retenido	Suprimido si está vacío y no hay actualizaciones pendientes	Mensajes suprimidos; cola suprimida si no hay actualizaciones pendientes
Cola dinámica temporal (llamada emitida por el creador de la cola)	Suprimida	Suprimida	Suprimida
Cola dinámica temporal (llamada no emitida por el creador de la cola)	Retenido	No válido	No válido
Lista de distribución	Retenido	No válido	No válido
Destino de suscripción gestionada	Retenido	No válido	No válido
Lista de distribución (se ha eliminado la suscripción)	Mensajes suprimidos; cola suprimida	No válido	No válido

Opciones de cierre de suscripción: Estas opciones controlan si se eliminan las suscripciones duraderas cuando se cierra el descriptor de contexto y si se limpian las publicaciones que todavía están a la espera de ser leídas por la aplicación. Estas opciones sólo son válidas para su uso con un descriptor de objeto devuelto en el parámetro *Hsub* de una llamada MQSUB.

MQCO_KEEP_SUB
El descriptor de contexto de la suscripción se cierra, pero la suscripción realizada se mantiene. Las publicaciones se siguen enviando al destino especificado en la suscripción. Esta opción sólo es válida si la suscripción se ha realizado con la opción MQSO_DURABLE.

MQCO_KEEP_SUB es el valor predeterminado si la suscripción es duradera

MQCO_REMOVE_SUB
La suscripción se elimina y se cierra el descriptor de contexto de la suscripción.

El parámetro *Hobj* de la llamada MQSUB no se invalida mediante el cierre del parámetro *Hsub* y puede seguir utilizándose para MQGET o MQCB para recibir las publicaciones restantes. Cuando el parámetro *Hobj* de la llamada MQSUB también se cierra, si era un destino gestionado se eliminan las publicaciones no recuperadas.

MQCO_REMOVE_SUB es el valor predeterminado si la suscripción no es duradera.

Estas opciones de cierre de suscripción se resumen en las tablas siguientes.

Para cerrar un descriptor de contexto de suscripción duradera pero conservar la suscripción, utilice las siguientes opciones de cierre de suscripción:

Tarea	**Opción de cierre de suscripción**
Mantener publicaciones en un descriptor de contexto MQOPENed	MQCO_KEEP_SUB
Eliminar publicaciones en un descriptor de contexto MQOPENed	Acción no permitida
Mantener publicaciones en un descriptor de contexto MQSO_MANAGED	MQCO_KEEP_SUB
Eliminar publicaciones en un descriptor de contexto MQSO_MANAGED	Acción no permitida

Para anular la suscripción, ya sea cerrando un descriptor de contexto de suscripción duradera y cancelándola o cerrando un descriptor de contexto de suscripción no duradera, utilice las siguientes opciones de cierre de suscripción:

Tarea	**Opción de cierre de suscripción**
Mantener publicaciones en un descriptor de contexto MQOPENed	MQCO_REMOVE_SUB
Eliminar publicaciones en un descriptor de contexto MQOPENed	Acción no permitida
Mantener publicaciones en un descriptor de contexto MQSO_MANAGED	MQCO_REMOVE_SUB

Opciones de lectura anticipada: Las opciones siguientes controlan lo que sucede con los mensajes no persistentes que se han enviado al cliente antes de que una aplicación los solicitara y que todavía no han sido consumidos por la aplicación. Estos mensajes se almacenan en el almacenamiento intermedio de lectura anticipada del cliente a la espera de ser solicitados por la aplicación y se pueden descartar o consumir de la cola antes de que se complete MQCLOSE.

MQCO_INMEDIATO
El objeto se cierra inmediatamente y los mensajes que se han enviado al cliente antes de que una aplicación los solicitara se descartan y no están disponibles para que los consuma ninguna aplicación. Este es el valor predeterminado.

MQCO_QUIESCE
Se realiza una solicitud para cerrar el objeto, pero si alguno de los mensajes que se han enviado al cliente antes de que una aplicación los solicitara, sigue residiendo en el almacenamiento intermedio de lectura anticipada del cliente, la llamada MQCLOSE devuelve un aviso de MQRC_READ_AHEAD_MSGS y el descriptor de contexto del objeto sigue siendo válido.

A continuación, la aplicación puede seguir utilizando el descriptor de objeto para recuperar mensajes hasta que no haya más disponibles y, a continuación, vuelva a cerrar el objeto. No se envían más mensajes al cliente antes de que una aplicación los solicite, la lectura anticipada ahora está desactivada.

Se recomienda a las aplicaciones que utilicen MQCO_QUIESCE en lugar de intentar alcanzar un punto en el que no haya más mensajes en el almacenamiento intermedio de lectura anticipada del cliente, porque podría llegar un mensaje entre la última llamada MQGET y el siguiente MQCLOSE que se descartaría si se utilizara MQCO_IMMEDIATE.

Si se emite un MQCLOSE con MQCO_QUIESCE desde una función de devolución de llamada asíncrona, se aplica el mismo comportamiento de lectura anticipada de mensajes. Si se devuelve el aviso MQRC_READ_AHEAD_MSGS, se llama a la función de devolución de llamada al menos una vez más. Cuando el último mensaje restante que se ha leído con anticipación se ha pasado a la función de devolución de llamada, el campo ConsumerFlags de MQCBC se establece en MQCBCF_READA_BUFFER_EMPTY.

Opción predeterminada: Si no necesita ninguna de las opciones descritas anteriormente, puede utilizar la opción siguiente:

MQCO_NONE
No es necesario ningún proceso de cierre opcional.

Este *debe* especificarse para:

- Objetos que no son colas
- Colas predefinidas
- Colas dinámicas temporales (pero sólo en aquellos casos en los que *Hobj no* es el descriptor de contexto devuelto por la llamada MQOPEN que ha creado la cola).
- Listas de distribución

En todos los casos anteriores, el objeto se conserva y no se suprime.

Si se especifica esta opción para una cola dinámica temporal:

- La cola se suprime, si la ha creado la llamada MQOPEN que ha devuelto *Hobj*; se depura cualquier mensaje que esté en la cola.
- En todos los demás casos, la cola (y los mensajes que contiene) se retienen.

Si se especifica esta opción para una cola dinámica permanente, la cola se conserva y no se suprime.

En z/OS, si la cola es una cola dinámica que se ha suprimido lógicamente, y este es el último descriptor de contexto para ella, la cola se suprime físicamente. Para conocer detalles, consulte “Notas de uso” en la página 634.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Los códigos de razón listados son los que puede devolver el gestor de colas para el parámetro *Reason*.

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_WARNING:

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') El mensaje lógico no está completo.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_STRUC_FAILED
(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') CICS ha rechazado la solicitud de espera.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC__DB2_NOT_AVAILABLE
(2342, X' 926 ') El subsistema Db2 no está disponible.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR
(2019, X'7E3') El manejador de objeto no es válido.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OPTION_NOT_VALID_FOR_TYPE
(2045, X'7FD') En una llamada MQOPEN o MQCLOSE: opción no válida para el tipo de objeto.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_Q_NOT_EMPTY
(2055, X'807 ') La cola contiene uno o más mensajes o solicitudes put u get no confirmadas.

MQRC_READ_AHEAD_MSGS
(nnnn, X'xxx ') El cliente tiene mensajes de lectura anticipada que la aplicación aún no ha consumido.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SECURITY_ERROR
(2063, X'80F') Se ha producido un error de seguridad.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso

1. Cuando una aplicación emite la llamada MQDISC, o finaliza de forma normal o anómala, los objetos abiertos por la aplicación y que siguen abiertos se cierran automáticamente con la opción MQCO_NONE.
2. Los puntos siguientes se aplican si el objeto que se está cerrando es una *cola*:
   - Si las operaciones en la cola se realizan como parte de una unidad de trabajo, la cola se puede cerrar antes o después de que se produzca el punto de sincronización sin afectar al resultado del punto de sincronización. Si se desencadena la cola, la realización de una retrotracción antes de cerrar la cola puede hacer que se emita un mensaje desencadenante. Para obtener más información sobre los mensajes desencadenantes, consulte Propiedades de los mensajes desencadenantes.
   - Si la cola se ha abierto con la opción MQOO_BROWSE, el cursor para examinar se destruye. Si la cola se vuelve a abrir con la opción MQOO_BROWSE, se crea un nuevo cursor para examinar (consulte MQOO_BROWSE).
   - Si un mensaje está bloqueado actualmente para este descriptor de contexto en el momento de la llamada MQCLOSE, el bloqueo se libera (consulte MQGMO_LOCK).
   - En z/OS, si hay una solicitud MQGET con la opción MQGMO_SET_SIGNAL pendiente para el descriptor de contexto de cola que se está cerrando, la solicitud se cancela (consulte MQGMO_SET_SIGNAL). Las solicitudes de señal para la misma cola pero alojadas en distintos manejadores (*Hobj*) no se ven afectadas (a menos que se suprima una cola dinámica, en cuyo caso también se cancelan).
3. Los puntos siguientes se aplican si el objeto que se está cerrando es una *cola dinámica* (permanente o temporal):
   - Para una cola dinámica, puede especificar las opciones MQCO_DELETE y MQCO_DELETE_PURGE independientemente de las opciones especificadas en la llamada MQOPEN correspondiente.
   - Cuando se suprime una cola dinámica, se cancelan todas las llamadas MQGET con la opción MQGMO_WAIT que están pendientes en la cola y se devuelve el código de razón MQRC_Q_DELETED. Consulte MQGMO_WAIT.

Aunque las aplicaciones no pueden acceder a una cola suprimida, la cola no se elimina del sistema y los recursos asociados no se liberan, hasta que se hayan cerrado todos los descriptores de contexto que hacen referencia a la cola y todas las unidades de trabajo que afectan a la cola se hayan confirmado o restituido.

En z/OS, una cola que se ha suprimido lógicamente pero que todavía no se ha eliminado del sistema impide la creación de una nueva cola con el mismo nombre que la cola suprimida; la llamada MQOPEN falla con el código de razón MQRC_NAME_IN_USE en este caso. Además, una cola de este tipo se puede seguir visualizando utilizando mandatos MQSC, aunque las aplicaciones no puedan acceder a ella.

- Cuando se suprime una cola dinámica permanente, si el descriptor de contexto *Hobj* especificado en la llamada MQCLOSE *no* es el que ha devuelto la llamada MQOPEN que ha creado la cola, se comprueba que el identificador de usuario que se ha utilizado para validar la llamada MQOPEN está autorizado para suprimir la cola. Si se ha especificado la opción MQOO_ALTERNATE_USER_AUTHORITY en la llamada MQOPEN, el identificador de usuario seleccionado es *AlternateUserId*.

  Esta comprobación no se realiza si:

  – El descriptor de contexto especificado es el devuelto por la llamada MQOPEN que ha creado la cola.

  – La cola que se está suprimiendo es una cola dinámica temporal.
- Cuando se cierra una cola dinámica temporal, si el descriptor de contexto *Hobj* especificado en la llamada MQCLOSE es el que ha devuelto la llamada MQOPEN que ha creado la cola, se suprime la cola. Esto ocurre independientemente de las opciones de cierre especificadas en la llamada MQCLOSE. Si hay mensajes en la cola, se descartan; no se generan mensajes de informe.

  Si hay unidades de trabajo no confirmadas que afectan a la cola, la cola y sus mensajes se siguen suprimiendo, pero las unidades de trabajo no fallan. Sin embargo, como se ha descrito anteriormente, los recursos asociados con las unidades de trabajo no se liberan hasta que cada una de las unidades de trabajo se haya confirmado o restituido.

4. Los puntos siguientes se aplican si el objeto que se está cerrando es una *lista de distribución*:

   - La única opción de cierre válida para una lista de distribución es MQCO_NONE; la llamada falla con el código de razón MQRC_OPTIONS_ERROR o MQRC_OPTION_NOT_VALID_FOR_TYPE si se especifica alguna otra opción.
   - Cuando se cierra una lista de distribución, los códigos de terminación y los códigos de razón individuales no se devuelven para las colas de la lista; sólo los parámetros *CompCode* y *Reason* de la llamada están disponibles para fines de diagnóstico.

     Si se produce una anomalía al cerrar una de las colas, el gestor de colas continúa el proceso e intenta cerrar las colas restantes de la lista de distribución. Los parámetros *CompCode* y *Reason* de la llamada se establecen para devolver información que describe la anomalía. Es posible que el código de finalización sea MQCC_FAILED, aunque la mayoría de las colas se hayan cerrado correctamente. La cola que ha encontrado el error no está identificada.

     Si hay una anomalía en más de una cola, no se define qué anomalía se notifica en los parámetros *CompCode* y *Reason* .

5. En IBM i, si la aplicación se conectó implícitamente cuando se emitió la primera llamada MQOPEN, se produce un MQDISC implícito cuando se emite el último MQCLOSE.

   Sólo las aplicaciones que se ejecutan en modalidad de compatibilidad se pueden conectar implícitamente; otras aplicaciones deben emitir la llamada MQCONN o MQCONNX para conectarse al gestor de colas de forma explícita.

Invocación en C

```
MQCLOSE (Hconn, &Hobj, Options, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
     CALL 'MQCLOSE' USING HCONN, HOBJ, OPTIONS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQCLOSE (Hconn, Hobj, Options, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
CALL MQCLOSE,(HCONN,HOBJ,OPTIONS,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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
```

Invocación en Visual Basic

```
MQCLOSE Hconn, Hobj, Options, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
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-Confirmar cambios

La llamada MQCMIT indica al gestor de colas que la aplicación ha alcanzado un punto de sincronización y que todas las obtenciones y colocaciones de mensajes que se han producido desde el último punto de sincronización se deben hacer permanentes.

Los mensajes colocados como parte de una unidad de trabajo se ponen a disposición de otras aplicaciones; los mensajes recuperados como parte de una unidad de trabajo se suprimen.

- En z/OS, la llamada solo la utilizan los programas por lotes (incluidos los programas DL/I por lotes IMS).
- En IBM i, esta llamada no está soportada para las aplicaciones que se ejecutan en modalidad de compatibilidad.

Sintaxis

MQCMIT (*Hconn*, *CompCode*, *Razón*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* lo ha devuelto una llamada MQCONN o MQCONNX anterior.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Los códigos de razón listados son los que puede devolver el gestor de colas para el parámetro *Reason*.

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_WARNING:

MQRC_BACKED_OUT
(2003, X'7D3') Unidad de trabajo restituida.

MQRC_OUTCOME_PENDING
(2124, X'84C') El resultado de la operación de confirmación está pendiente.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CALL_INTERRUPTED
(2549, X'9F5') MQPUT o MQCMIT se ha interrumpido y el proceso de reconexión no puede restablecer un resultado definido.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

ERROR_ENTORNO_MQRC
(2012, X'7DC') La llamada no es válida en el entorno.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OUTCOME_MIXED
(2123, X'84B') El resultado de la operación de confirmación o de devolución se mezcla.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RECONNECT_FAILED
(2548, X'9F4') Después de la reconexión, se ha producido un error al restablecer los descriptores de contexto para una conexión reconectable.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso

1. Utilice esta llamada sólo cuando el propio gestor de colas coordine la unidad de trabajo. Este puede ser:
   - Unidad de trabajo local, donde los cambios sólo afectan a los recursos de WebSphere MQ .
   - Unidad de trabajo global, donde los cambios pueden afectar a los recursos que pertenecen a otros gestores de recursos, así como a los recursos de WebSphere MQ .

   Para obtener más detalles sobre las unidades de trabajo locales y globales, consulte "MQBEGIN- Iniciar unidad de trabajo" en la página 610.
2. En entornos en los que el gestor de colas no coordina la unidad de trabajo, se debe utilizar la llamada de confirmación adecuada en lugar de MQCMIT. El entorno también puede dar soporte a una confirmación implícita causada por la terminación normal de la aplicación.

- En z/OS, utilice las llamadas siguientes:
  - Los programas por lotes (incluidos los programas DL/I por lotes IMS) pueden utilizar la llamada MQCMIT si la unidad de trabajo sólo afecta a los recursos de WebSphere MQ . Sin embargo, si la unidad de trabajo afecta tanto a los recursos de WebSphere MQ como a los recursos que pertenecen a otros gestores de recursos (por ejemplo, DB2), utilice la llamada SRRCMIT proporcionada por z/OS Recoverable Resource Service (RRS). La llamada SRRCMIT confirma los cambios en los recursos que pertenecen a los gestores de recursos que se han habilitado para la coordinación RRS.
  - Las aplicaciones CICS deben utilizar el mandato `EXEC CICS SYNCPOINT` para confirmar la unidad de trabajo de forma explícita. De forma alternativa, la finalización de la transacción da como resultado una confirmación implícita de la unidad de trabajo. La llamada MQCMIT no se puede utilizar para aplicaciones CICS .
  - Las aplicaciones IMS (que no sean programas DL/I por lotes) deben utilizar llamadas IMS como `GU` y `CHKP` para confirmar la unidad de trabajo. La llamada MQCMIT no se puede utilizar para aplicaciones IMS (que no sean programas DL/I por lotes).
- En IBM i, utilice esta llamada para las unidades de trabajo locales coordinadas por el gestor de colas. Esto significa que no debe existir una definición de compromiso a nivel de trabajo, es decir, el mandato `STRCMTCTL` con el parámetro `CMTSCOPE(*JOB)` no debe haberse emitido para el trabajo.

3. Si una aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de si la aplicación finaliza de forma normal o anómala. Consulte las notas de uso de MQDISC para obtener más detalles.
4. Cuando una aplicación coloca u obtiene mensajes en grupos o segmentos de mensajes lógicos, el gestor de colas conserva información relacionada con el grupo de mensajes y el mensaje lógico para las últimas llamadas MQPUT y MQGET satisfactorias. Esta información está asociada con el descriptor de contexto de cola e incluye cosas como:
   - Los valores de los campos *GroupId*, *MsgSeqNumber*, *Offset*y *MsgFlags* en MQMD.
   - Indica si el mensaje forma parte de una unidad de trabajo.
   - Para la llamada MQPUT: si el mensaje es persistente o no persistente.

   Cuando se confirma una unidad de trabajo, el gestor de colas conserva la información de grupo y segmento, y la aplicación puede continuar colocando u obteniendo mensajes en el grupo de mensajes o mensaje lógico actual.

   La retención de la información de grupo y segmento cuando se confirma una unidad de trabajo permite a la aplicación distribuir un grupo de mensajes grande o un mensaje lógico grande que consta de muchos segmentos entre varias unidades de trabajo. El uso de varias unidades de trabajo es ventajoso si el gestor de colas local sólo tiene almacenamiento de cola limitado. Sin embargo, la aplicación debe mantener suficiente información para reiniciar la colocación u obtención de mensajes en el punto correcto si se produce una anomalía del sistema. Para obtener detalles sobre cómo reiniciar en el punto correcto después de una anomalía del sistema, consulte MQPMO_LOGICAL_ORDER y MQGMO_LOGICAL_ORDER.

   Las notas de uso restantes sólo se aplican cuando el gestor de colas coordina las unidades de trabajo:
5. Una unidad de trabajo tiene el mismo ámbito que un descriptor de conexión; todas las llamadas de WebSphere MQ que afectan a una unidad de trabajo determinada deben realizarse utilizando el mismo descriptor de conexión. Las llamadas emitidas utilizando un descriptor de conexión diferente (por ejemplo, las llamadas emitidas por otra aplicación) afectan a una unidad de trabajo diferente. Consulte el parámetro *Hconn* descrito en MQCONN para obtener información sobre el ámbito de los manejadores de conexión.
6. Esta llamada sólo afecta a los mensajes que se han colocado o recuperado como parte de la unidad de trabajo actual.
7. Una aplicación de larga ejecución que emite llamadas MQGET, MQPUT o MQPUT1 dentro de una unidad de trabajo, pero que nunca emite una llamada de confirmación o de devolución, puede llenar las colas con mensajes que no están disponibles para otras aplicaciones. Para evitar esto, el administrador debe establecer el atributo de gestor de colas *MaxUncommittedMsgs* en un valor lo

suficientemente bajo como para evitar que las aplicaciones fuera de control llenen las colas, pero lo suficientemente alto como para permitir que las aplicaciones de mensajería esperadas funcionen correctamente.

8. En sistemas UNIX y Windows , si el parámetro *Reason* es MQRC_CONNECTION_BROKEN (con un *CompCode* de MQCC_FAILED), o MQRC_UNEXPECTED_ERROR, es posible que la unidad de trabajo se haya confirmado correctamente.

Invocación en C

```
MQCMIT (Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQCMIT' USING HCONN, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQCMIT (Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQCMIT,(HCONN,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQCMIT Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQCONN-Conectar gestor de colas

La llamada MQCONN conecta un programa de aplicación a un gestor de colas.

Proporciona un descriptor de conexión del gestor de colas, que la aplicación utiliza en las llamadas de cola de mensajes posteriores.

- En z/OS, las aplicaciones CICS no tienen que emitir esta llamada. Estas aplicaciones se conectan automáticamente al gestor de colas al que está conectado el sistema CICS . Sin embargo, las llamadas MQCONN y MQDISC se siguen aceptando de las aplicaciones CICS .
- En IBM i, las aplicaciones que se ejecutan en modalidad de compatibilidad no tienen que emitir esta llamada. Estas aplicaciones se conectan automáticamente al gestor de colas cuando emiten la primera llamada MQOPEN. Sin embargo, las llamadas MQCONN y MQDISC se siguen aceptando de las aplicaciones IBM i.

  Otras aplicaciones (es decir, aplicaciones que no se ejecutan en modalidad de compatibilidad) deben utilizar la llamada MQCONN o MQCONNX para conectarse al gestor de colas y la llamada MQDISC para desconectarse del gestor de colas. Este es el estilo recomendado de programación.

No se puede realizar una conexión de cliente en una instalación sólo de servidor, y no se puede realizar una conexión local en una instalación sólo de cliente.

Sintaxis

MQCONN (*QMgrName*, *Hconn*, *CompCode*, *Razón*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Es el nombre del gestor de colas al que la aplicación desea conectarse. El nombre puede contener los siguientes caracteres:

- Caracteres alfabéticos en mayúsculas (de la A a la Z)
- Caracteres alfabéticos en minúsculas (de la a a la z)
- Dígitos numéricos (de 0 a 9)
- Punto (.), barra inclinada (/), subrayado (_), porcentaje (%)

El nombre no debe contener espacios en blanco iniciales o intercalados, pero puede contener espacios en blanco finales. Se puede utilizar un carácter nulo para indicar el final de datos significativos en el nombre; el nulo y cualquier carácter que lo siga se tratan como espacios en blanco. Las restricciones siguientes se aplican en los entornos indicados:

- En sistemas que utilizan EBCDIC Katakana, no se pueden utilizar caracteres en minúsculas.
- En z/OS, los nombres que empiezan o terminan con un subrayado no pueden ser procesados por los paneles de operaciones y los paneles de control. Por esta razón, evite este tipo de nombres.

- En IBM i, escriba los nombres que contengan caracteres en minúsculas, una barra inclinada o un porcentaje entre comillas cuando se especifiquen en mandatos. No especifique estas comillas en el parámetro *QMgrName* .

Si el nombre consta por completo de espacios en blanco, se utiliza el nombre del gestor de colas *por omisión* .

El nombre especificado para *QMgrName* debe ser el nombre de un gestor de colas *conectable* .

En z/OS, los gestores de colas a los que es posible conectarse están determinados por el entorno:

- Para CICS, solo puede utilizar el gestor de colas al que está conectado el sistema CICS . Todavía se debe especificar el parámetro *QMgrName* , pero su valor se ignora; se recomiendan espacios en blanco.
- Para IMS, sólo se pueden conectar los gestores de colas listados en la tabla de definición de subsistema (CSQQDEFV), *y* listados en la tabla SSM en IMS(consulte la nota de uso 6).
- Para el proceso por lotes z/OS y TSO, sólo se pueden conectar los gestores de colas que residen en el mismo sistema que la aplicación (consulte la nota de uso 6).

Grupos de compartición de colas:En sistemas donde existen varios gestores de colas y están configurados para formar un grupo de compartición de colas, el nombre del grupo de compartición de colas se puede especificar para *QMgrName* en lugar del nombre de un gestor de colas. Esto permite a la aplicación conectarse a *cualquier* gestor de colas que esté disponible en el grupo de compartición de colas y que esté en la misma imagen de z/OS que la aplicación. El sistema también se puede configurar para que el uso de un *QMgrName* en blanco se conecte al grupo de compartición de colas en lugar de al gestor de colas predeterminado.

Si *QMgrName* especifica el nombre del grupo de compartición de colas, pero también hay un gestor de colas con ese nombre en el sistema, se establece una conexión con el último en lugar del primero. Sólo si esa conexión falla, se intenta la conexión con uno de los gestores de colas del grupo de compartición de colas.

Si la conexión es satisfactoria, puede utilizar el descriptor de contexto devuelto por la llamada MQCONN o MQCONNX para acceder a *todos* los recursos (compartidos y no compartidos) que pertenecen al gestor de colas al que se ha realizado la conexión. El acceso a estos recursos está sujeto a los controles de autorización típicos.

Si la aplicación emite dos llamadas MQCONN o MQCONNX para establecer conexiones simultáneas, y una o ambas llamadas especifica el nombre del grupo de compartición de colas, la segunda llamada devuelve el código de terminación MQCC_WARNING y el código de razón MQRC_ALREADY_CONNECTED cuando se conecta al mismo gestor de colas que la primera llamada.

Los grupos de compartición de colas solo están soportados en z/OS. La conexión a un grupo de compartición de colas sólo está soportada en los entornos por lotes, por lotes RRS y TSO.

WebSphere MQ Aplicaciones cliente MQI: para aplicaciones cliente MQI de WebSphere MQ MQI, se intenta una conexión para cada definición de canal de conexión de cliente con el nombre de gestor de colas especificado, hasta que una sea satisfactoria. Sin embargo, el gestor de colas debe tener el mismo nombre que el nombre especificado. Si se especifica un nombre todo en blanco, se intenta cada canal de conexión de cliente con un nombre de gestor de colas todo en blanco hasta que uno sea satisfactorio; en este caso, no hay ninguna comprobación con respecto al nombre real del gestor de colas.

WebSphere MQ no están soportadas en z/OS, pero z/OS puede actuar como un servidor WebSphere MQ , al que se pueden conectar las aplicaciones cliente de WebSphere MQ .

WebSphere MQ Grupos de gestores de colas de cliente MQI: si el nombre especificado empieza con un asterisco (*), el gestor de colas con el que se realiza la conexión puede tener un nombre distinto del especificado por la aplicación. El nombre especificado (sin el asterisco) define un *grupo* de gestores de colas que son elegibles para la conexión. La implementación selecciona uno del grupo intentando cada uno a su vez hasta que se encuentra uno con el que se puede realizar una conexión. El orden en el que se intentan las conexiones está influenciado por el peso del canal de cliente y los valores de afinidad de conexión de los canales candidatos. Si ninguno de los gestores de colas

del grupo está disponible para la conexión, la llamada falla. Cada gestor de colas se intenta una sola vez. Si se especifica un asterisco solo para el nombre, se utiliza un grupo de gestores de colas predeterminado definido por la implementación.

Los grupos de gestores de colas sólo están soportados para las aplicaciones que se ejecutan en un entorno MQ-client; la llamada falla si una aplicación no cliente especifica un nombre de gestor de colas que empieza con un asterisco. Un grupo se define proporcionando varias definiciones de canal de conexión de cliente con el mismo nombre de gestor de colas (el nombre especificado sin el asterisco), para comunicarse con cada uno de los gestores de colas del grupo. El grupo predeterminado se define proporcionando una o más definiciones de canal de conexión de cliente, cada una con un nombre de gestor de colas en blanco (por lo tanto, especificar un nombre todo en blanco tiene el mismo efecto que especificar un solo asterisco para el nombre de una aplicación cliente).

Después de conectarse a un gestor de colas de un grupo, una aplicación puede especificar espacios en blanco de la forma típica en los campos de nombre de gestor de colas en los descriptores de mensaje y objeto para indicar el nombre del gestor de colas al que se ha conectado la aplicación (el *gestor de colas local*). Si la aplicación necesita conocer este nombre, utilice la llamada MQINQ para consultar el atributo de gestor de colas *QMgrName* .

El prefijo de un asterisco al nombre de conexión implica que la aplicación no depende de la conexión con un gestor de colas determinado del grupo. Las aplicaciones adecuadas son:

- Aplicaciones que colocan mensajes pero no obtienen mensajes.
- Aplicaciones que colocan mensajes de solicitud y, a continuación, obtienen los mensajes de respuesta de una cola *dinámica temporal* .

Las aplicaciones no adecuadas son las que necesitan obtener mensajes de una cola determinada en un gestor de colas determinado; estas aplicaciones no deben añadir un asterisco al nombre.

Si especifica un asterisco, la longitud máxima del resto del nombre es de 47 caracteres.

Los grupos de gestores de colas no están soportados en z/OS.

La longitud de este parámetro la proporciona MQ_Q_MGR_NAME_LENGTH.

Hconn
Tipo: MQHCONN-salida

Este manejador representa la conexión con el gestor de colas. Especifíquelo en todas las llamadas de cola de mensajes posteriores emitidas por la aplicación. Deja de ser válido cuando se emite la llamada MQDISC, o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto.

WebSphere MQ ahora proporciona la biblioteca mqm con paquetes de cliente, así como paquetes de servidor. Esto significa que cuando se realiza una llamada MQI que se encuentra en la biblioteca mqm, se comprueba el tipo de conexión para ver si es una conexión de cliente o servidor y, a continuación, se realiza la llamada subyacente correcta. Por lo tanto, una salida que se pasa a un *Hconn* ahora se puede enlazar con la biblioteca mqm, pero se utiliza en una instalación de cliente.

Ámbito de descriptor de contexto:El ámbito del descriptor de contexto devuelto depende de la llamada utilizada para conectarse al gestor de colas (MQCONN o MQCONNX). Si la llamada utilizada es MQCONNX, el ámbito del descriptor de contexto también depende de la opción MQCNO_HANDLE_SHARE_ * especificada en el campo *Options* de la estructura MQCNO.

- Si la llamada es MQCONN, o se especifica la opción MQCNO_HANDLE_SHARE_NONE, el descriptor de contexto devuelto es un descriptor de contexto *no compartido* .

  El ámbito de un descriptor de contexto no compartido es la unidad más pequeña de proceso paralelo soportada por la plataforma en la que se ejecuta la aplicación (consulte Tabla 564 en la página 644 para obtener detalles); el descriptor de contexto no es válido fuera de la unidad de proceso paralelo desde la que se ha emitido la llamada.
- Si especifica la opción MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK, el descriptor de contexto devuelto es un descriptor de contexto *compartido* .

El ámbito de un descriptor de contexto compartido es el proceso que es propietario de la hebra desde la que se ha emitido la llamada; el descriptor de contexto se puede utilizar desde cualquier hebra que pertenezca a dicho proceso. No todas las plataformas soportan hebras.

- Si la llamada MQCONN o MQCONNX falla con un código de terminación igual a MQCC_FAILED, el valor Hconn no está definido.

Tabla 564. Ámbito de descriptores de contexto no compartidos en diversas plataformas

Plataforma	**Ámbito de descriptor de contexto no compartido**
z/OS	• CICS: la tarea CICS • IMS: la tarea, hasta el siguiente punto de sincronización (excluyendo las subtareas de la tarea) • z/OS por lotes y TSO: la tarea (excluidas las subtareas de la tarea)
IBM i	Trabajo
Sistemas UNIX	Hebra
Aplicaciones Windows de 16 bits	Proceso
Aplicaciones Windows de 32 bits	Hebra

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, el valor devuelto es:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_WARNING:

MQRC_ALREADY_CONNECTED
(2002, X'7D2') La aplicación ya está conectada.

MQRC_CLUSTER_EXIT_LOAD_ERROR
(2267, X'8DB') No se puede cargar la salida de carga de trabajo de clúster.

MQRC_SSL_ALREADY_INITIALIZED
(2391, X' 957 ') SSL ya se ha inicializado.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_CONN_LOAD_ERROR
(2129, X'851 ') No se ha podido cargar el módulo de conexión del adaptador.

MQRC_ADAPTER_DEFS_ERROR
(2131, X'853 ') Módulo de definición de subsistema de adaptador no válido.

MQRC_ADAPTER_DEFS_LOAD_ERROR
(2132, X'854 ') No se ha podido cargar el módulo de definición de subsistema de adaptador.

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ADAPTER_STORAGE_INSUFICIENTE
(2127, X'84F') Almacenamiento insuficiente para el adaptador.

MQRC_ANOTHER_Q_MGR_CONNECTED
(2103, X'837 ') Otro gestor de colas ya está conectado.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_INIT_ERROR
(2375, X' 947 ') La inicialización de salida de API ha fallado.

MQRC_API_EXIT_TERM_ERROR
(2376, X' 948 ') La terminación de salida de API ha fallado.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONN_ID_IN_USE
(2160, X'870 ') El identificador de conexión ya está en uso.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_ERROR
(2273, X'8E1') Error al procesar la llamada MQCONN.

MQRC_CONNECTION_NOT_AVAILABLE
(2568, X'A08') Se produce en una llamada MQCONN o MQCONNX cuando el gestor de colas no puede proporcionar una conexión del tipo de conexión solicitado en la instalación actual. Una conexión con el cliente no se puede realizar en una instalación solo de servidor. No se puede realizar una conexión local en una instalación solo de cliente.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_CRYPTO_HARDWARE_ERROR
(2382, X'94E') Error de configuración de hardware criptográfico.

MQRC_DUPLICATE_RECOV_COORD
(2163, X'873 ') El coordinador de recuperación existe.

ERROR_ENTORNO_MQRC
(2012, X'7DC') La llamada no es válida en el entorno.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOST_NOT_AVAILABLE
(2538, X'9EA') Se ha emitido una llamada MQCONN desde un cliente para conectarse a un gestor de colas, pero el intento de asignar una conversación al sistema remoto ha fallado.

MQRC_INSTALLATION_MISMATCH
(2583, X'A17') Discrepancia entre la instalación del gestor de colas y la biblioteca seleccionada.

MQRC_KEY_REPOSITORY_ERROR
(2381, X'94D') El repositorio de claves no es válido.

MQRC_MAX_CONNS_LIMIT_ALCANZADO
(2025, X'7E9') Se ha alcanzado el número máximo de conexiones.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_OPEN_FAILED
(2137, X'859 ') El objeto no se ha abierto correctamente.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SECURITY_ERROR
(2063, X'80F') Se ha producido un error de seguridad.

MQRC_SSL_INITIALIZATION_ERROR
(2393, X' 959 ') Error de inicialización de SSL.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso

1. El gestor de colas con el que se realiza la conexión utilizando la llamada MQCONN se denomina *gestor de colas local*.
2. Las colas que son propiedad del gestor de colas local aparecen en la aplicación como colas locales. Es posible colocar mensajes y obtener mensajes de estas colas.

   Las colas compartidas que son propiedad del grupo de compartición de colas al que pertenece el gestor de colas local aparecen en la aplicación como colas locales. Es posible colocar mensajes y obtener mensajes de estas colas.

   Las colas que son propiedad de gestores de colas remotos aparecen como colas remotas. Es posible colocar mensajes en estas colas, pero no obtener mensajes de estas colas.
3. Si el gestor de colas falla mientras una aplicación se está ejecutando, la aplicación debe volver a emitir la llamada MQCONN para obtener un nuevo descriptor de conexión que utilizar en las llamadas posteriores de WebSphere MQ . La aplicación puede emitir la llamada MQCONN periódicamente hasta que la llamada sea satisfactoria.

   Si una aplicación no está segura de si está conectada al gestor de colas, la aplicación puede emitir de forma segura una llamada MQCONN para obtener un descriptor de conexión. Si la aplicación

ya está conectada, el descriptor de contexto devuelto es el mismo que el devuelto por la llamada MQCONN anterior, pero con el código de terminación MQCC_WARNING y el código de razón MQRC_ALREADY_CONNECTED.

4. Cuando la aplicación ha terminado de utilizar las llamadas de WebSphere MQ , la aplicación debe utilizar la llamada MQDISC para desconectarse del gestor de colas.
5. Si la llamada MQCONN falla con un código de terminación igual a MQCC_FAILED, el valor Hconn no está definido.
6. En z/OS:
   - Las aplicaciones por lotes, TSO y IMS deben emitir la llamada MQCONN para utilizar las otras llamadas WebSphere MQ . Estas aplicaciones pueden conectarse a más de un gestor de colas simultáneamente.

     Si el gestor de colas falla, la aplicación debe volver a emitir la llamada después de que el gestor de colas se haya reiniciado para obtener un nuevo descriptor de conexión.

     Aunque las aplicaciones IMS pueden emitir la llamada MQCONN repetidamente, aunque ya estén conectadas, esto no se recomienda para los programas de proceso de mensajes en línea (MPP).
   - Las aplicaciones CICS no tienen que emitir la llamada MQCONN para utilizar las otras llamadas WebSphere MQ , pero pueden hacerlo si lo desean; se aceptan tanto la llamada MQCONN como la llamada MQDISC. Sin embargo, no es posible conectarse a más de un gestor de colas simultáneamente.

     Si el gestor de colas falla, estas aplicaciones se vuelven a conectar automáticamente cuando se reinicia el gestor de colas, por lo que no es necesario emitir la llamada MQCONN.
7. En z/OS, para definir los gestores de colas disponibles:
   - Para aplicaciones por lotes, los programadores del sistema pueden utilizar la macro CSQBDEF para crear un módulo (CSQBDEFV) que defina el nombre del gestor de colas predeterminado o el nombre del grupo de compartición de colas.
   - Para aplicaciones IMS , los programadores del sistema pueden utilizar la macro CSQQDEFX para crear un módulo (CSQQDEFV) que defina los nombres de los gestores de colas disponibles y especifique el gestor de colas predeterminado.

     Además, cada gestor de colas debe estar definido en la región de control IMS y en cada región dependiente que acceda a ese gestor de colas. Para ello, debe crear un miembro de subsistema en IMS.Biblioteca PROCLIB e identifique el miembro de subsistema en las regiones de IMS aplicables. Si una aplicación intenta conectarse a un gestor de colas que no está definido en el miembro del subsistema para su región IMS , la aplicación termina de forma anómala.
8. En IBM i, las aplicaciones escritas para releases anteriores del gestor de colas se pueden ejecutar sin volver a compilar. Esto se denomina *modalidad de compatibilidad*. Esta modalidad de operación proporciona un entorno de ejecución compatible para las aplicaciones. Se compone de lo siguiente:
   - El programa de servicio AMQZSTUB que reside en la biblioteca QMQM.

     AMQZSTUB proporciona la misma interfaz pública que los releases anteriores y tiene la misma firma. Utilice este programa de servicio para acceder a la MQI a través de llamadas de procedimiento enlazadas.
   - El programa QMQM que reside en la biblioteca QMQM.

     QMQM proporciona un medio de acceder a la MQI a través de llamadas de programa dinámicas.
   - Programas MQCLOSE, MQCONN, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQPUT1y MQSET que residen en la biblioteca QMQM.

     Estos programas también proporcionan un medio de acceder a la MQI a través de llamadas de programa dinámicas, pero con una lista de parámetros que corresponde a las descripciones estándar de las llamadas de WebSphere MQ .

Estas tres interfaces no incluyen prestaciones introducidas en WebSphere MQ Versión 5.1. Por ejemplo, las llamadas MQBACK, MQCMIT y MQCONNX no están soportadas. El soporte proporcionado por estas interfaces es sólo para aplicaciones de una sola hebra.

El soporte para las nuevas llamadas de WebSphere MQ en aplicaciones de una sola hebra, y para todas las llamadas de WebSphere MQ en aplicaciones de varias hebras, se proporciona a través de los programas de servicio LIBMQM y LIBMQM_R.

9. En IBM i, los programas que finalizan de forma anómala no se desconectan automáticamente del gestor de colas. Escriba aplicaciones para permitir la posibilidad de que la llamada MQCONN o MQCONNX devuelva el código de terminación MQCC_WARNING y el código de razón MQRC_ALREADY_CONNECTED. Utilice el descriptor de conexión devuelto en esta situación como normal.

Invocación en C

```
MQCONN (QMgrName, &Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQCHAR48  QMgrName;  /* Name of queue manager */
MQHCONN   Hconn;     /* Connection handle */
MQLONG    CompCode;  /* Completion code */
MQLONG    Reason;    /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQCONN' USING QMGRNAME, HCONN, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQCONN (QMgrName, Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
CALL MQCONN,(QMGRNAME,HCONN,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
QMGRNAME  DS  CL48  Name of queue manager
HCONN     DS  F     Connection handle
COMPCODE  DS  F     Completion code
REASON    DS  F     Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQCONN QMgrName, Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
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 - Conectar gestor de colas (ampliado)

La llamada MQCONNX conecta un programa de aplicación a un gestor de colas. Proporciona un descriptor de conexión del gestor de colas, que utiliza la aplicación en las llamadas posteriores de WebSphere MQ .

La llamada MQCONNX es como la llamada MQCONN, excepto que MQCONNX permite especificar opciones para controlar la forma en que funciona la llamada.

- Esta llamada está soportada en todos los sistemas WebSphere MQ y clientes WebSphere MQ conectados a estos sistemas.
- En IBM i, esta llamada no está soportada para las aplicaciones que se ejecutan en modalidad de compatibilidad.

No se puede realizar una conexión de cliente en una instalación sólo de servidor, y no se puede realizar una conexión local en una instalación sólo de cliente.

Sintaxis

MQCONNX (*QMgrName*, *ConnectOpts*, *Hconn*, *CompCode*, *Razón*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Consulte el parámetro *QMgrName* descrito en “MQCONN-Conectar gestor de colas” en la página 641 para obtener más detalles.

ConnectOpts
Tipo: MQCNO-entrada/salida

Para obtener más detalles, consulte “MQCNO - Opciones de conexión” en la página 297.

Hconn
Tipo: MQHCONN-salida

Este manejador representa la conexión con el gestor de colas. Especifíquelo en todas las llamadas de cola de mensajes posteriores emitidas por la aplicación. Deja de ser válido cuando se emite la llamada MQDISC, o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto.

WebSphere MQ ahora proporciona la biblioteca mqm con paquetes de cliente, así como paquetes de servidor. Esto significa que cuando se realiza una llamada MQI que se encuentra en la biblioteca mqm, se comprueba el tipo de conexión para ver si es una conexión de cliente o servidor y, a

continuación, se realiza la llamada subyacente correcta. Por lo tanto, una salida que se pasa a un *Hconn* ahora se puede enlazar con la biblioteca mqm, pero se utiliza en una instalación de cliente.

Ámbito de descriptor de contexto:El ámbito del descriptor de contexto devuelto depende de la llamada utilizada para conectarse al gestor de colas (MQCONN o MQCONNX). Si la llamada utilizada es MQCONNX, el ámbito del descriptor de contexto también depende de la opción MQCNO_HANDLE_SHARE_ * especificada en el campo *Options* de la estructura MQCNO.

- Si la llamada es MQCONN, o se especifica la opción MQCNO_HANDLE_SHARE_NONE, el descriptor de contexto devuelto es un descriptor de contexto *no compartido* .

  El ámbito de un descriptor de contexto no compartido es la unidad más pequeña de proceso paralelo soportada por la plataforma en la que se ejecuta la aplicación (consulte Tabla 565 en la página 650 para obtener detalles); el descriptor de contexto no es válido fuera de la unidad de proceso paralelo desde la que se ha emitido la llamada.
- Si especifica la opción MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK, el descriptor de contexto devuelto es un descriptor de contexto *compartido* .

  El ámbito de un descriptor de contexto compartido es el proceso que es propietario de la hebra desde la que se ha emitido la llamada; el descriptor de contexto se puede utilizar desde cualquier hebra que pertenezca a dicho proceso. No todas las plataformas soportan hebras.
- Si la llamada MQCONN o MQCONNX falla con un código de terminación igual a MQCC_FAILED, el valor Hconn no está definido.

Tabla 565. Ámbito de descriptores de contexto no compartidos en diversas plataformas

Plataforma	**Ámbito de descriptor de contexto no compartido**
z/OS	• CICS: la tarea CICS • IMS: la tarea, hasta el siguiente punto de sincronización (excluyendo las subtareas de la tarea) • z/OS por lotes y TSO: la tarea (excluidas las subtareas de la tarea)
IBM i	Trabajo
Sistemas UNIX	Hebra
Aplicaciones Windows de 16 bits	Proceso
Aplicaciones Windows de 32 bits	Hebra

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, el valor devuelto es:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

CompCode
Tipo: MQLONG - salida

Consulte el parámetro *CompCode* descrito en "MQCONN-Conectar gestor de colas" en la página 641 para obtener más detalles.

Razón
Tipo: MQLONG - salida

Las llamadas MQCONN y MQCONNX pueden devolver los códigos siguientes. Para obtener una lista de códigos adicionales que puede devolver la llamada MQCONNX, consulte los códigos siguientes.

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_WARNING:

MQRC_ALREADY_CONNECTED
(2002, X'7D2') La aplicación ya está conectada.

MQRC_CLUSTER_EXIT_LOAD_ERROR
(2267, X'8DB') No se puede cargar la salida de carga de trabajo de clúster.

MQRC_SSL_ALREADY_INITIALIZED
(2391, X' 957 ') SSL ya se ha inicializado.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_CONN_LOAD_ERROR
(2129, X'851 ') No se ha podido cargar el módulo de conexión del adaptador.

MQRC_ADAPTER_DEFS_ERROR
(2131, X'853 ') Módulo de definición de subsistema de adaptador no válido.

MQRC_ADAPTER_DEFS_LOAD_ERROR
(2132, X'854 ') No se ha podido cargar el módulo de definición de subsistema de adaptador.

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ADAPTER_STORAGE_INSUFICIENTE
(2127, X'84F') Almacenamiento insuficiente para el adaptador.

MQRC_ANOTHER_Q_MGR_CONNECTED
(2103, X'837 ') Otro gestor de colas ya está conectado.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_INIT_ERROR
(2375, X' 947 ') La inicialización de salida de API ha fallado.

MQRC_API_EXIT_TERM_ERROR
(2376, X' 948 ') La terminación de salida de API ha fallado.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONN_ID_IN_USE
(2160, X'870 ') El identificador de conexión ya está en uso.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_ERROR
(2273, X'8E1') Error al procesar la llamada MQCONN.

MQRC_CONNECTION_NOT_AVAILABLE
(2568, X'A08') Se produce en una llamada MQCONN o MQCONNX cuando el gestor de colas no puede proporcionar una conexión del tipo de conexión solicitado en la instalación actual. Una conexión con el cliente no se puede realizar en una instalación solo de servidor. No se puede realizar una conexión local en una instalación solo de cliente.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_CRYPTO_HARDWARE_ERROR
(2382, X'94E') Error de configuración de hardware criptográfico.

MQRC_DUPLICATE_RECOV_COORD
(2163, X'873 ') El coordinador de recuperación existe.

ERROR_ENTORNO_MQRC
(2012, X'7DC') La llamada no es válida en el entorno.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOST_NOT_AVAILABLE
(2538, X'9EA') Se ha emitido una llamada MQCONN desde un cliente para conectarse a un gestor de colas, pero el intento de asignar una conversación al sistema remoto ha fallado.

MQRC_INSTALLATION_MISMATCH
(2583, X'A17') Discrepancia entre la instalación del gestor de colas y la biblioteca seleccionada.

MQRC_KEY_REPOSITORY_ERROR
(2381, X'94D') El repositorio de claves no es válido.

MQRC_MAX_CONNS_LIMIT_ALCANZADO
(2025, X'7E9') Se ha alcanzado el número máximo de conexiones.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_OPEN_FAILED
(2137, X'859 ') El objeto no se ha abierto correctamente.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SECURITY_ERROR
(2063, X'80F') Se ha producido un error de seguridad.

MQRC_SSL_INITIALIZATION_ERROR
(2393, X' 959 ') Error de inicialización de SSL.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

La llamada MQCONNX puede devolver los siguientes códigos de razón adicionales:

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_AIR_ERROR
(2385, X' 951 ') El registro de información de autenticación no es válido.

MQRC_AUTH_INFO_CONN_NAME_ERROR
(2387, X' 953 ') Nombre de conexión de información de autenticación no válido.

MQRC_AUTH_INFO_REC_COUNT_ERROR
(2383, X'94F') El recuento de registros de información de autenticación no es válido.

MQRC_AUTH_INFO_REC_ERROR
(2384, X' 950 ') Los campos de registro de información de autenticación no son válidos.

MQRC_AUTH_INFO_TYPE_ERROR
(2386, X' 952 ') Tipo de información de autenticación no válido.

MQRC_CD_ERROR
(2277, X'8E5') Definición de canal no válida.

MQRC_CLIENT_CONN_ERROR
(2278, X'8E6') Los campos de conexión de cliente no son válidos.

MQRC_CNO_ERROR
(2139, X'85B') La estructura de opciones de conexión no es válida.

MQRC_CONN_TAG_IN_USE
(2271, X'8DF') Código de conexión en uso.

MQRC_CONN_TAG_NOT_USABLE
(2350, X'92E') Código de conexión no utilizable.

MQRC_LDAP_PASSWORD_ERROR
(2390, X' 956 ') La contraseña LDAP no es válida.

MQRC_LDAP_USER_NAME_ERROR
(2388, X' 954 ') Los campos de nombre de usuario LDAP no son válidos.

MQRC_LDAP_USER_NAME_LENGTH_ERR
(2389, X' 955 ') La longitud del nombre de usuario LDAP no es válida.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_SCO_ERROR
(2380, X'94C') La estructura de opciones de configuración SSL no es válida.

MQRC_SSL_CONFIG_ERROR
(2392, X' 958 ') Error de configuración SSL.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso

Para el lenguaje de programación Visual Basic, se aplica el punto siguiente:

- El parámetro *ConnectOpts* se declara como de tipo MQCNO. Si la aplicación se ejecuta como un cliente MQI de WebSphere MQ y desea especificar los parámetros del canal de conexión de cliente, declare el parámetro *ConnectOpts* como de tipo Any, para que la aplicación pueda especificar una estructura MQCNOCD en la llamada en lugar de una estructura MQCNO. Sin embargo, esto significa que no se puede comprobar el parámetro *ConnectOpts* para asegurarse de que es el tipo de datos correcto.

Invocación en C

```
MQCONNX (QMgrName, &ConnectOpts, &Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
CALL 'MQCONNX' USING QMGRNAME, CONNECTOPTS, HCONN, COMPCODE,
REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQCONNX (QMgrName, ConnectOpts, Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
CALL MQCONNX,(QMGRNAME,CONNECTOPTS,HCONN,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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
```

Invocación en Visual Basic

```
MQCONNX QMgrName, ConnectOpts, Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
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-Crear manejador de mensajes

La llamada MQCRTMH devuelve un descriptor de mensaje.

Una aplicación puede utilizar la llamada MQCRTMH en llamadas de cola de mensajes posteriores:

- Utilice la llamada MQSETMP para establecer una propiedad del manejador de mensajes.
- Utilice la llamada MQINQMP para consultar el valor de una propiedad del manejador de mensajes.
- Utilice la llamada MQDLTMP para suprimir una propiedad del manejador de mensajes.

El descriptor de mensaje se puede utilizar en las llamadas MQPUT y MQPUT1 para asociar las propiedades del descriptor de mensaje con las del mensaje que se está colocando. De forma similar, al especificar un descriptor de mensaje en la llamada MQGET, se puede acceder a las propiedades del mensaje que se está recuperando utilizando el descriptor de mensaje cuando se completa la llamada MQGET.

Utilice MQDLTMH para suprimir el descriptor de mensaje.

Sintaxis

MQCRTMH (*Hconn*, *CrtMsgHOpts*, *Hmsg*, *CompCode*, *Razón*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* lo ha devuelto una llamada MQCONN o MQCONNX anterior. Si la conexión con el gestor de colas deja de ser válida y no hay ninguna llamada WebSphere MQ operativa en el manejador de mensajes, se llama implícitamente a MQDLTMH para suprimir el mensaje.

De forma alternativa, puede especificar el valor siguiente:

MQHC_UNASSOCIATED_HCONN
El descriptor de conexión no representa una conexión con ningún gestor de colas en particular.

Cuando se utiliza este valor, el descriptor de mensaje debe suprimirse con una llamada explícita a MQDLTMH para liberar cualquier almacenamiento asignado a él; WebSphere MQ nunca suprime implícitamente el descriptor de mensaje.

Debe haber al menos una conexión válida con un gestor de colas establecido en la hebra que crea el manejador de mensajes; de lo contrario, la llamada falla con MQRC_HCONN_ERROR.

En un entorno con varias instalaciones en un único sistema, el valor MQHC_UNASSOCIATED_HCONN está limitado a utilizar con la primera instalación cargada en el proceso. Se devuelve el código de razón MQRC_HMSG_NOT_AVAILABLE si el manejador de mensajes se proporciona a una instalación diferente.

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, la llamada MQCONN se puede omitir y puede especificar el valor siguiente para *Hconn*:

MQHC_DEF_CONN
Descriptor de conexión predeterminado

CrtMsgHOpts
Tipo: MQCMHO-entrada

Las opciones que controlan la acción de MQCRTMH. Consulte MQCMHO para obtener detalles.

Msj
Tipo: MQHMSG-salida

En la salida, se devuelve un descriptor de mensaje que se puede utilizar para establecer, consultar y suprimir propiedades del descriptor de mensaje. Inicialmente, el manejador de mensajes no contiene propiedades.

Un descriptor de mensaje también tiene un descriptor de mensaje asociado. Inicialmente contiene los valores predeterminados. Los valores de los campos de descriptor de mensaje asociados se pueden establecer y consultar utilizando las llamadas MQSETMP y MQINQMP. La llamada MQDLTMP restablece un campo del descriptor de mensaje a su valor predeterminado.

Si el parámetro *Hconn* se especifica como el valor MQHC_UNASSOCIATED_HCONN, el descriptor de mensaje devuelto se puede utilizar en llamadas MQGET, MQPUT o MQPUT1 con cualquier conexión dentro de la unidad de proceso, pero sólo puede ser utilizado por una llamada WebSphere MQ cada vez. Si el descriptor de contexto está en uso cuando una segunda llamada WebSphere MQ intenta utilizar el mismo descriptor de contexto de mensaje, la segunda llamada WebSphere MQ falla con el código de razón MQRC_MSG_HANDLE_IN_USE.

Si el parámetro *Hconn* no es MQHC_UNASSOCIATED_HCONN, el manejador de mensajes devuelto sólo se puede utilizar en la conexión especificada.

Se debe utilizar el mismo valor de parámetro *Hconn* en las llamadas MQI posteriores donde se utiliza este manejador de mensajes:

- MQDLTMH
- MQSETMP
- MQINQMP
- MQDLTMP
- MQMHBUF
- MQBUFMH

El descriptor de mensaje devuelto deja de ser válido cuando se emite la llamada MQDLTMH para el descriptor de mensaje, o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto. MQDLTMH se llama implícitamente si se proporciona una conexión específica cuando se crea el descriptor de mensaje y la conexión con el gestor de colas deja de ser válida, por ejemplo, si se llama a MQDBC.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_CMHO_ERROR
(2461, X'099D') Crear estructura de opciones de manejador de mensajes no válida.

MQRC_CONNECTION_BROKEN
(2273, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_HANDLE_NOT_AVAILABLE
(2017, X'07E1') No hay más manejadores disponibles.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HMSG_ERROR
(2460, X'099C') El puntero de manejador de mensajes no es válido.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Opciones no válidas o no coherentes.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

C

```
MQCRTMH (Hconn, &CrtMsgHOpts, &Hmsg, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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, CRTMSGOPTS, HMSG, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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);
```

Declare los parámetros como se indica a continuación:

```
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)
```

Declare los parámetros como se indica a continuación:

```
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-devoluciones de llamada de control

La llamada MQCTL realiza acciones de control en devoluciones de llamada y los manejadores de objeto abiertos para una conexión.

Sintaxis

MQCTL (*Hconn*, *Operation*, *ControlOpts*, *CompCode*, *Reason*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, la llamada MQCONN se puede omitir y puede especificar el siguiente valor especial para *Hconn*:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

operation
Tipo: MQLONG - entrada

La operación que se está procesando en la devolución de llamada definida para el descriptor de objeto especificado. Debe especificar una, y sólo una, de las opciones siguientes:

MQOP_START

Iniciar el consumo de mensajes para todas las funciones de consumidor de mensajes definidas para el descriptor de conexión especificado.

Las devoluciones de llamada se ejecutan en una hebra iniciada por el sistema, que es diferente de cualquiera de las hebras de aplicación.

Esta operación proporciona el control del descriptor de conexión proporcionado al sistema. Las únicas llamadas MQI que puede emitir una hebra que no sea la hebra de consumidor son:

- MQCTL con operación MQOP_STOP
- MQCTL con la operación MQOP_SUSPEND

- MQDISC-Realiza MQCTL con la operación MQOP_STOP antes de desconectar el HConn.

MQRC_HCONN_ASYNC_ACTIVE se devuelve si se emite una llamada de API WebSphere MQ mientras se inicia el descriptor de conexión y la llamada no se origina en una función de consumidor de mensajes.

Si un consumidor de mensajes detiene la conexión durante MQCBCT_START_CALL, la llamada MQCTL devuelve un código de razón de anomalía de MQRC_CONNECTION_STOPPED.

Esto se puede emitir en una función de consumidor. Para la misma conexión que la rutina de devolución de llamada, su única finalidad es cancelar una operación MQOP_STOP emitida anteriormente.

Esta opción no está soportada en los entornos siguientes: CICS en z/OS o si la aplicación está enlazada con una biblioteca WebSphere MQ sin hebras.

MQOP_START_WAIT
Iniciar el consumo de mensajes para todas las funciones de consumidor de mensajes definidas para el descriptor de conexión especificado.

Los consumidores de mensajes se ejecutan en la misma hebra y el control no se devuelve al interlocutor de MQCTL hasta que:

- Liberado por el uso de las operaciones MQCTL MQOP_STOP o MQOP_SUSPEND, o
- Todas las rutinas de consumidor se han desregistrado o suspendido.

Si se anula el registro o se suspenden todos los consumidores, se emite una operación MQOP_STOP implícita.

Esta opción no se puede utilizar desde una rutina de devolución de llamada, ya sea para el descriptor de conexión actual o cualquier otro descriptor de conexión. Si se intenta la llamada, se devuelve con MQRC_ENVIRONMENT_ERROR.

Si, en algún momento durante una operación MQOP_START_WAIT no hay ningún consumidor registrado, no suspendido, la llamada falla con un código de razón de MQRC_NO_CALLBACKS_ACTIVE.

Si, durante una operación MQOP_START_WAIT, la conexión se suspende, la llamada MQCTL devuelve un código de razón de aviso de MQRC_CONNECTION_SUSPENDED; la conexión permanece 'iniciada'.

La aplicación puede elegir emitir MQOP_STOP o MQOP_RESUME. En esta instancia, la operación MQOP_RESUME se bloquea.

Esta opción no está soportada en un cliente de una sola hebra.

MQOP_STOP
Detenga el consumo de mensajes y espere a que todos los consumidores completen sus operaciones antes de que se complete esta opción. Esta operación libera el descriptor de conexión.

Si se emite desde dentro de una rutina de devolución de llamada, esta opción no entra en vigor hasta que se cierra la rutina. No se llama a más rutinas de consumidor de mensajes después de que se hayan completado las rutinas de consumidor para los mensajes que ya se han leído, y después de que se hayan realizado llamadas de detención (si se han solicitado) a rutinas de devolución de llamada.

Si se emite fuera de una rutina de devolución de llamada, el control no vuelve al llamante hasta que se hayan completado las rutinas del consumidor para los mensajes ya leídos y después de que se hayan realizado las llamadas de detención (si se han solicitado) a las devoluciones de llamada. Sin embargo, las devoluciones de llamada permanecen registradas.

Esta función no tiene ningún efecto en los mensajes de lectura anticipada. Debe asegurarse de que los consumidores ejecuten MQCLOSE (MQCO_QUIESCE), desde dentro de la función de devolución de llamada, para determinar si hay más mensajes disponibles para entregar.

MQOP_SUSPENDER
Pausar el consumo de mensajes. Esta operación libera el descriptor de conexión.

Esto no tiene ningún efecto en la lectura anticipada de los mensajes para la aplicación. Si tiene previsto dejar de consumir mensajes durante mucho tiempo, considere la posibilidad de cerrar la cola y volver a abrirla cuando continúe el consumo.

Si se emite desde dentro de una rutina de devolución de llamada, no entra en vigor hasta que se cierra la rutina. No se llamarán más rutinas de consumidor de mensajes después de las salidas de rutina actuales.

Si se emite fuera de una devolución de llamada, el control no vuelve al interlocutor hasta que se haya completado la rutina de consumidor actual y no se llame a más.

MQOP_RESUME
Reanude el consumo de mensajes.

Esta opción se emite normalmente desde la hebra de aplicación principal, pero también se puede utilizar desde una rutina de devolución de llamada para cancelar una solicitud de suspensión anterior emitida en la misma rutina.

Si se utiliza MQOP_RESUME para reanudar un MQOP_START_WAIT, la operación se bloquea.

ControlOpts
Tipo: MQCTLO-entrada

Opciones que controlan la acción de MQCTL

Consulte "MQCTLO-Estructura de opciones de devolución de llamada de control" en la página 317 para obtener detalles de la estructura.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

nMQRC_ADAPTER_CONV_LOAD_ERROR
(2133, X'855') No se han podido cargar módulos de servicio de conversión de datos.

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALLBACK_LINK_ERROR
(2487, X'9B7') No se puede llamar a la rutina de devolución de llamada

MQRC_CALLBACK_NOT_ REGISTRADO
(2448, X' 990 ') No se puede anular el registro, suspender o reanudar porque no hay ninguna devolución de llamada registrada

MQRC_CALLBACK_ROUTINE_ERROR
(2486, X'9B6') Se ha especificado CallbackFunction y CallbackName en una llamada MQOP_REGISTER.

O bien se ha especificado CallbackFunction o CallbackName pero no coincide con la función de devolución de llamada registrada actualmente.

MQRC_CALLBACK_TYPE_ERROR
(2483, X'9B3') Campo de tipo CallBackincorrecto.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CBD_ERROR
(2444, X'98C') El bloque de opciones es incorrecto.

MQRC_CBD_OPTIONS_ERROR
(2484, X'9B4') Campo de opciones MQCBD incorrecto.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') CICS ha rechazado la solicitud de espera.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_CORREL_ID_ERROR
(2207, X'89F') Error de identificador de correlación.

MQRC_FUNCTION_NOT_SUPPORTED
(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

MQRC_GET_INHIBITED
(2016, X'7E0') Se han inhibido las obtenciones para la cola.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GMO_ERROR
(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR
(2019, X'7E3') El manejador de objeto no es válido.

MQRC_INCONSISTENT_BROWSE
(2259, X'8D3') La especificación de examinar es incoherente.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_INVALID_MSG_UNDER_CURSOR
(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') Las opciones de coincidencia no son válidas.

MQRC_MAX_MSG_LENGTH_ERROR
(2485, X'9B5') Campo de longitud MaxMsgincorrecto

MQRC_MD_ERROR
(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MODULE_ENTRY_NOT_FOUND
(2497, X'9C1') No se ha podido encontrar el punto de entrada de función especificado en el módulo.

MQRC_MODULE_INVALID
(2496, X'9C0') Se ha encontrado el módulo pero es del tipo incorrecto (32 bit/64 bit) o no es una dll válida.

MQRC_MODULE_NOT_FOUND
(2495, X'9BF') El módulo no se ha encontrado en la vía de acceso de búsqueda o no tiene autorización para cargarse.

MQRC_MSG_ID_ERROR
(2206, X'89E') Error de identificador de mensaje.

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') El uso del símbolo de mensaje no es válido.

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') La cola no se ha abierto para examen.

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') La cola no se ha abierto para entrada.

MQRC_OBJECT_CHANGED
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OPERATION_ERROR
(2488, X'9B8') Código de operación incorrecto en llamada de API

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_DELETED
(2052, X'804') La cola se ha suprimido.

MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') La cola tiene un tipo de índice incorrecto.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SIGNAL_OUTSTANDING
(2069, X'815') Símbolo pendiente para este descriptor de contexto.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818 ') El soporte de punto de sincronización no está disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WAIT_INTERVAL_ERROR
(2090, X'82A') El intervalo de espera de MQGMO no es válido.

MQRC_WRONG_GMO_VERSION
(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso

1. Las rutinas de devolución de llamada deben comprobar las respuestas de todos los servicios que invocan y, si la rutina detecta una condición que no se puede resolver, debe emitir un mandato MQCB MQOP_DEREGISTER para evitar llamadas repetidas a la rutina de devolución de llamada.
2. En z/OS, cuando la operación es MQOP_START:
   - Los programas que utilizan rutinas de devolución de llamada asíncrona deben estar autorizados para utilizar z/OS UNIX System Services (USS).
   - Los programas LE (Language Environment) que utilizan rutinas de devolución de llamada asíncrona deben utilizar la opción de tiempo de ejecución LE POSIX(ON).
   - Los programas no LE que utilizan rutinas de devolución de llamada asíncronas no deben utilizar la interfaz pthread_create de USS (servicio invocable BPX1PTC).
3. MQCTL no está soportado en el adaptador IMS .

Nota: En CICS, MQOP_START no está soportado. En su lugar, utilice la llamada de función MQOP_START_WAIT.

Invocación en C

```
MQCTL (Hconn, Operation, &ControlOpts, &CompCode, &Reason)
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
CALL 'MQCTL' USING HCONN, OPERATION, CTLOPTS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQCTL(Hconn, Operation, CtlOpts, CompCode, Reason)
```

Declare los parámetros como se indica a continuación:

```
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-Desconectar gestor de colas

La llamada MQDISC interrumpe la conexión entre el gestor de colas y el programa de aplicación, y es la inversa de la llamada MQCONN o MQCONNX.

- En z/OS, todas las aplicaciones que utilizan el consumo de mensajes asíncronos, el manejo de sucesos o la devolución de llamada, la hebra de control principal debe emitir una llamada MQDISC antes de finalizar. Consulte Consumo asíncrono de mensajes de WebSphere MQ para obtener más detalles.
- En z/OS, las aplicaciones CICS no necesitan emitir esta llamada para desconectarse del gestor de colas, pero es posible que deban emitirla para finalizar el uso de una etiqueta de conexión.
- En IBM i, las aplicaciones que se ejecutan en modalidad de compatibilidad no necesitan emitir esta llamada. Consulte “MQCONN-Conectar gestor de colas” en la página 641 para obtener más información.

Sintaxis

MQDISC (*Hconn*, *CompCode*, *Razón*)

Parámetros

Hconn
Tipo: MQHCONN-entrada/salida

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* lo ha devuelto una llamada MQCONN o MQCONNX anterior.

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, puede omitir la llamada MQCONN y especificar el valor siguiente para *Hconn*:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Al finalizar correctamente la llamada, el gestor de colas establece *Hconn* en un valor que no es un descriptor de contexto válido para el entorno. Este valor es:

MQHC_UNUSABLE_HCONN
Descriptor de conexión inutilizable.

En z/OS, *Hconn* se establece en un valor que no está definido.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los códigos siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_WARNING:

MQRC_BACKED_OUT
(2003, X'7D3') Unidad de trabajo restituida.

MQRC_CONN_TAG_NOT_LIBERADO
(2344, X' 928 ') No se ha liberado la etiqueta de conexión.

MQRC_OUTCOME_PENDING
(2124, X'84C') El resultado de la operación de confirmación está pendiente.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_DISC_LOAD_ERROR
(2138, X'85A') No se ha podido cargar el módulo de desconexión del adaptador.

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_INIT_ERROR
(2375, X' 947 ') La inicialización de salida de API ha fallado.

MQRC_API_EXIT_TERM_ERROR
(2376, X' 948 ') La terminación de salida de API ha fallado.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_OUTCOME_MIXED
(2123, X'84B') El resultado de la operación de confirmación o de devolución se mezcla.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso

1. Si se emite una llamada MQDISC cuando la conexión todavía tiene objetos abiertos bajo esa conexión, el gestor de colas cierra esos objetos, con las opciones de cierre establecidas en MQCO_NONE.
2. Si la aplicación finaliza con cambios no confirmados en una unidad de trabajo, la disposición de dichos cambios depende de cómo finalice la aplicación:

   a. Si la aplicación emite la llamada MQDISC antes de finalizar:

   - Para una unidad de trabajo coordinada por el gestor de colas, el gestor de colas emite la llamada MQCMIT en nombre de la aplicación. La unidad de trabajo se confirma si es posible y se restituye si no es así.
   - Para una unidad de trabajo coordinada externamente, no hay ningún cambio en el estado de la unidad de trabajo; sin embargo, el gestor de colas normalmente indica que la unidad de trabajo debe confirmarse cuando se lo solicite el coordinador de la unidad de trabajo.

     En z/OS, CICS, IMS (que no sean programas DL/1 por lotes) y aplicaciones RRS son como estas.

   b. Si la aplicación finaliza normalmente pero sin emitir la llamada MQDISC, la acción realizada depende del entorno:

   - En z/OS, excepto para aplicaciones MQ Java o MQ JMS, se producen las acciones descritas en la nota 2a .
   - En todos los demás casos, se producen las acciones descritas en la nota 2c .

Debido a las diferencias entre entornos, asegúrese de que las aplicaciones que desea portar confirmen o restituya la unidad de trabajo antes de que finalicen.

c. Si la aplicación finaliza *anormalmente* sin emitir la llamada MQDISC, la unidad de trabajo se restituye.

3. En z/OS, se aplican los puntos siguientes:

   - Las aplicaciones CICS no tienen que emitir la llamada MQDISC para desconectarse del gestor de colas, porque el propio sistema CICS se conecta al gestor de colas y la llamada MQDISC no tiene ningún efecto en esta conexión.
   - CICS, IMS (que no sean programas DL/1 por lotes) y aplicaciones RRS utilizan unidades de trabajo coordinadas por un coordinador de unidad de trabajo externo. Como resultado, la llamada MQDISC no afecta al estado de la unidad de trabajo (si existe) que existe cuando se emite la llamada.

     Sin embargo, la llamada MQDISC *sí* indica el final de uso del código de conexión *ConnTag* que se ha asociado con la conexión mediante una llamada MQCONNX anterior emitida por la aplicación. Si hay una unidad de trabajo activa que hace referencia al código de conexión cuando se emite la llamada MQDISC, la llamada se completa con el código de terminación MQCC_WARNING y el código de razón MQRC_CONN_TAG_NOT_LIBERAR. El código de conexión no pasa a estar disponible para su reutilización hasta que el coordinador de unidad de trabajo externo haya resuelto la unidad de trabajo.

4. En IBM i, las aplicaciones que se ejecutan en modalidad de compatibilidad no tienen que emitir esta llamada; consulte la llamada MQCONN para obtener más detalles.

Nota: En CICS, MQOP_START no está soportado. En su lugar, utilice la llamada de función MQOP_START_WAIT.

Invocación en C

```
MQDISC (&Hconn, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQDISC' USING HCONN, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQDISC (Hconn, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación de ensamblador de System/390

```
CALL MQDISC,(HCONN,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Invocación en Visual Basic

```
MQDISC Hconn, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQDLTMH-Suprimir descriptor de mensaje

La llamada MQDLTMH suprime un manejador de mensajes y es el inverso de la llamada MQCRTMH.

Sintaxis

MQDLTMH (*Hconn*, *Hmsg*, *DltMsgHOpts*, *CompCode*, *Razón*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas.

El valor debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro *Hmsg* .

Si el manejador de mensajes se ha creado utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en la hebra suprimiendo el manejador de mensajes; de lo contrario, la llamada falla con MQRC_CONNECTION_BROKEN.

Msj
Tipo: MQHMSG-entrada/salida

Este es el descriptor de contexto de mensaje que se va a suprimir. El valor ha sido devuelto por una llamada MQCRTMH anterior.

Al finalizar correctamente la llamada, el descriptor de contexto se establece en un valor no válido para el entorno. Este valor es:

MQHM_UNUSABLE_HMSG
Manejador de mensajes inutilizable.

El descriptor de mensaje no se puede suprimir si hay otra llamada WebSphere MQ en curso a la que se ha pasado el mismo descriptor de mensaje.

DltMsgHOpts
Tipo: MQDMHO-entrada

Para obtener más detalles, consulte "MQDMHO-Suprimir opciones de manejador de mensajes" en la página 334.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_DMHO_ERROR
(2462, X'099E') La estructura de opciones del manejador de mensajes no es válida.

MQRC_HMSG_ERROR
(2460, X'099C') El puntero de manejador de mensajes no es válido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Opciones no válidas o no coherentes.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Invocación en C

```
MQDLTMH (Hconn, &Hmsg, &DltMsgHOpts, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
CALL 'MQDLTMH' USING HCONN, HMSG, DLTMSGOPTS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** Connection handle
01  HCONN  PIC S9(9) BINARY.

** Options that control the action of MQDLTMH
01  DLTMSGHOPTS.
   COPY CMQDLMHOV.

** Completion code
01 COMPCODE  PIC S9(9) BINARY.

** Reason code qualifying COMPCODE
 01  REASON   PIC S9(9) BINARY.
```

Invocación en PL/I

```
call MQDLTMH (Hconn, Hmsg, DltMsgHOpts, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
dcl Hconn       fixed bin(31);  /* Connection handle */
dcl Hmsg        fixed bin(63);  /* Message handle */
dcl DltMsgHOpts like MQDMHO;    /* Options that control the action of MQDLTMH */
dcl CompCode    fixed bin(31);  /* Completion code */
dcl Reason      fixed bin(31);  /* Reason code qualifying CompCode */
```

Invocación en ensamblador de alto nivel

```
CALL MQDLTMH,(HCONN,HMSG,DLTMSGHOPTS,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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-Suprimir propiedad de mensaje

La llamada MQDLTMP suprime una propiedad de un manejador de mensajes y es la inversa de la llamada MQSETMP.

Sintaxis

MQDLTMP (*Hconn*, *Hmsg*, *DltPropOpts*, *Nombre*, *CompCode*, *Razón*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro *Hmsg* .

Si el descriptor de mensaje se ha creado utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en la hebra suprimiendo el descriptor de mensaje, de lo contrario, la llamada fallará con MQRC_CONNECTION_BROKEN.

Msj
Tipo: MQHMSG-entrada

Este es el descriptor de contexto de mensaje que contiene la propiedad que se va a suprimir. El valor ha sido devuelto por una llamada MQCRTMH anterior.

DltPropOpts
Tipo: MQDMPO-entrada

Consulte el tipo de datos MQDMPO para obtener detalles.

Nombre
Tipo: MQCHARV-entrada

El nombre de la propiedad que se va a suprimir. Consulte Nombres de propiedad para obtener más información sobre los nombres de propiedad.

Los comodines no están permitidos en el nombre de propiedad.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_WARNING:

MQRC_PROPERTY_NOT_AVAILABLE
(2471, X'09A7') Propiedad no disponible.

MQRC_RFH_FORMAT_ERROR
(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'0852 ') No se puede cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH
(2157, X'086D') Los ASID primario y de inicio difieren.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_DMPO_ERROR
(2481, X'09B1') Suprimir estructura de opciones de propiedad de mensaje no válida.

MQRC_HMSG_ERROR
(2460, X'099C') Descriptor de mensaje no válido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Opciones no válidas o no coherentes.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Nombre de propiedad no válido.

MQRC_SOURCE_CCSID_ERROR
(2111, X'083F') Identificador de juego de caracteres codificado de nombre de propiedad no válido.

MQRC_UNEXPECTED_ERROR
(2195, X'0893 ') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte:

- Códigos de razón para WebSphere MQ for z/OS
- Códigos de razón de API para otras plataformas WebSphere MQ

.

Invocación en C

```
MQDLTMP (Hconn, Hmsg, &DltPropOpts, &Name, &CompCode, &Reason)
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
CALL 'MQDLTMP' USING HCONN, HMSG, DLTPROPOPTS, NAME, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQDLTMP (Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
CALL MQDLTMP,(HCONN,HMSG,DLTPROPOPTS,NAME,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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
```

Mensaje MQGET - get

La llamada MQGET recupera un mensaje de una cola local que se ha abierto utilizando la llamada MQOPEN.

Sintaxis

MQGET (*Hconn*, *Hobj*, *MsgDesc*, *GetMsgOpts*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Razón*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* lo ha devuelto una llamada MQCONN o MQCONNX anterior.

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn*:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Hobj

Tipo: MQHOBJ - entrada

Este manejador representa la cola de la que debe recuperarse el mensaje. El valor de *Hobj* lo ha devuelto una llamada MQOPEN anterior. La cola debe haberse abierto con una o varias de las opciones siguientes (consulte “MQOPEN-Abrir objeto” en la página 713 para obtener más detalles):

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE

MsgDesc
Tipo: MQMD - entrada/salida

Esta estructura describe los atributos del mensaje necesario y los del mensaje recuperado. Consulte "MQMD - Descriptor de mensaje" en la página 392 para obtener los detalles.

Si *BufferLength* es inferior a la longitud del mensaje, el gestor de colas llenará *MsgDesc*, si se ha especificado MQGMO_ACCEPT_TRUNCATED_MSG en el parámetro *GetMsgOpts* (consulte el campo MQGMO - Opciones).

Si la aplicación proporciona un MQMD de la versión 1, el mensaje devuelto tiene un MQMDE como prefijo de los datos del mensaje de aplicación, pero *solamente* si uno o varios de los campos del MQMDE tiene un valor no por omisión. Si todos los campos de MQMDE tienen valores predeterminados, se omitirá MQMDE. Un nombre de formato de MQFMT_MD_EXTENSION en el campo *Formato* en MQMD indica que hay un MQMDE.

La aplicación no tiene que proporcionar una estructura MQMD si se suministra un manejador de mensajes válido en el campo *MsgHandle*. Si no se proporciona ningún valor en este campo, el descriptor del mensaje se tomará del descriptor asociado a los manejadores de mensajes.

Si la aplicación proporciona un manejador de mensajes en lugar de una estructura MQMD, y especifica MQGMO_PROPERTIES_FORCE_MQRFH2, la llamada fallará con el código de razón MQRC_MD_ERROR. La llamada también falla, con el código de razón MQRC_MD_ERROR, si la aplicación no proporciona una estructura MQMD y especifica MQGMO_PROPERTIES_AS_Q_DEF, y el atributo de cola *PropertyControl* es MQPROP_FORCE_MQRFH2.

Si se especifican opciones de coincidencia y se utiliza el descriptor de mensajes asociado con el manejador de mensajes, los campos de entrada que se utilizan para la comparación proceden del manejador de mensajes.

GetMsgOpts
Tipo: MQGMO - entrada/salida

Para obtener más detalles, consulte "MQGMO-Opciones de obtención de mensajes" en la página 344.

BufferLength
Tipo: MQLONG - entrada

Esta longitud en bytes del área *Buffer*. Especifique cero para los mensajes que no tienen datos o si se va a eliminar el mensaje de la cola y a descartar los datos (en este caso debe especificar MQGMO_ACCEPT_TRUNCATED_MSG).

Nota: La longitud del mensaje más largo que puede leerse desde la cola se indica en el atributo de cola *MaxMsgLength*; vea el apartado "Atributos para colas" en la página 816.

Buffer
Tipo: MQBYTExBufferLength - salida

Se trata del área que contiene los datos del mensaje. Alinee el almacenamiento intermedio en un límite adecuado según la naturaleza de los datos del mensaje. La alineación de 4 bytes es adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen estructuras de cabecera IBM WebSphere MQ), pero algunos mensajes pueden requerir una alineación más estricta. Por ejemplo, un ejemplo que contiene un entero binario de 64 bits podría requerir una alineación de 8 bytes.

Si *BufferLength* es inferior a la longitud del mensaje, la cantidad máxima de mensaje se traspasará a *Buffer*; esto ocurre si se ha especificado MQGMO_ACCEPT_TRUNCATED_MSG en el parámetro *GetMsgOpts* (consulte el campo MQGMO - Opciones para obtener más información).

El juego de caracteres y la codificación de datos en *Buffer* lo indican los campos *CodedCharSetId* y *Encoding* que se devuelven en el parámetro *MsgDesc*. Si estos valores son distintos de los valores

que necesita el destinatario, éste deberá convertir los datos de mensajes de la aplicación en el juego de caracteres y en la codificación necesarios. Se puede utilizar la opción MQGMO_CONVERT (con una salida escrita por el usuario, si es necesario) para convertir los datos de mensaje; consulte "MQGMO- Opciones de obtención de mensajes" en la página 344 para obtener detalles de esta opción.

Nota: Todos los demás parámetros de la llamada MQGET están en el juego de caracteres y en la codificación del gestor de colas local (indicado por el atributo del gestor de colas *CodedCharSetId* y por MQENC_NATIVE).

Si falla la llamada, el contenido del almacenamiento también se podría haber modificado.

En el lenguaje de programación C, el parámetro se declara como pointer-to-void y se puede especificar cualquier tipo de datos como parámetro.

Si el parámetro *BufferLength* es cero, no se hará referencia a *Buffer*; en tal caso, la dirección del parámetro que han pasado los programas escritos en C o el ensamblador System/390 puede ser nula.

DataLength
Tipo: MQLONG - salida

Indica la longitud en bytes de los datos de la aplicación *en el mensaje*. Si este valor es mayor que *BufferLength*, solamente se devolverán los bytes de *BufferLength* en el parámetro (es decir, se truncará el mensaje). *Buffer*. Si el valor es cero, el mensaje no contendrá datos de aplicación.

Si *BufferLength* es inferior a la longitud del mensaje, el gestor de colas seguirá completando *DataLength*, si se ha especificado MQGMO_ACCEPT_TRUNCATED_MSG en el parámetro *GetMsgOpts* (consulte el campo MQGMO - Opciones para obtener más información). Esto permite a la aplicación determinar el tamaño del almacenamiento intermedio necesario para introducir los datos del mensaje y después reemitir la llamada con un almacenamiento intermedio que tenga el tamaño adecuado.

Sin embargo, si se especifica la opción MQGMO_CONVERT, y la longitud de los datos del mensaje convertido es demasiado larga para que quepa en *Buffer*, el valor devuelto por *DataLength* es:

- La longitud de los datos *no convertidos*, para los formatos definidos por el gestor de colas.

  En este caso, si la naturaleza de los datos hace que se expanda durante la conversión, la aplicación deberá localizar un almacenamiento intermedio mayor que el valor que ha devuelto el gestor de colas para *DataLength*.
- El valor devuelto por la salida de conversión de datos, para formatos definidos por la aplicación.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Los códigos de razón listados son los que puede devolver el gestor de colas para el parámetro *Reason*. Si la aplicación especifica la opción MQGMO_CONVERT y se invoca una salida escrita por el usuario para convertir algunos datos de mensajes o todos ellos, la salida decide qué valor se devuelve para el parámetro *Reason*. Como resultado, es posible tener valores distintos de los que se documentan.

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_WARNING:

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848') Los datos convertidos son demasiado grandes para el almacenamiento intermedio.

MQRC_CONVERTED_STRING_TOO_BIG
(2190, X'88E') La serie convertida es demasiado grande para el campo.

MQRC_DBCS_ERROR
(2150, X'866') La serie DBCS no es válida.

MQRC_FORMAT_ERROR
(2110, X'83E') El formato de mensaje no válido.

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') El mensaje lógico no está completo.

MQRC_INCONSISTENT_CCSIDS
(2243, X'8C3') Los segmentos del mensaje tienen distintos CCSID.

MQRC_INCONSISTENT_ENCODINGS
(2244, X'8C4') Los segmentos del mensaje tienen distintas codificaciones.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') Uso no válido del token de mensaje.

MQRC_NO_MSG_LOCKED
(2209, X'8A1') No hay ningún mensaje bloqueado.

MQRC_NOT_CONVERTED
(2119, X'847') Los datos del mensaje no se han convertido.

MQRC_OPTIONS_CHANGED
(nnnn, X'xxx') Se han cambiado las opciones que tenían que ser coherentes.

MQRC_PARTIALLY_CONVERTED
(2272, X'8E0') Los datos del mensaje se han convertido parcialmente.

MQRC_SIGNAL_REQUEST_ACCEPTED
(2070, X'816') No se ha devuelto ningún mensaje (pero se acepta la solicitud de señal).

MQRC_SOURCE_BUFFER_ERROR
(2145, X'861') El parámetro de almacenamiento intermedio de origen no es válido.

MQRC_SOURCE_CCSID_ERROR
(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

MQRC_SOURCE_DECIMAL_ENC_ERROR
(2113, X'841') Codificación de decimal empaquetado en el mensaje no reconocida.

MQRC_SOURCE_FLOAT_ENC_ERROR
(2114, X'842') Codificación de coma flotante en el mensaje no reconocida.

MQRC_SOURCE_INTEGER_ENC_ERROR
(2112, X'840') Codificación de entero de origen no reconocida.

MQRC_SOURCE_LENGTH_ERROR
(2143, X'85F') Parámetro de longitud de origen no válido.

MQRC_TARGET_BUFFER_ERROR
(2146, X'862') Parámetro de almacenamiento intermedio de destino no válido.

MQRC_TARGET_CCSID_ERROR
(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

MQRC_TARGET_DECIMAL_ENC_ERROR
(2117, X'845') Codificación de decimal empaquetado especificada por receptor no reconocida.

MQRC_TARGET_FLOAT_ENC_ERROR
(2118, X'846') Codificación de coma flotante especificada por receptor no reconocida.

MQRC_TARGET_INTEGER_ENC_ERROR
(2116, X'844') Codificación de entero de destino no reconocida.

MQRC_TRUNCATED_MSG_ACCEPTED
(2079, X'81F') Se ha devuelto un mensaje truncado (el proceso se ha completado).

MQRC_TRUNCATED_MSG_FAILED
(2080, X'820') Se ha devuelto un mensaje truncado (el proceso no se ha completado).

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

nMQRC_ADAPTER_CONV_LOAD_ERROR
(2133, X'855') No se han podido cargar módulos de servicio de conversión de datos.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BACKED_OUT
(2003, X'7D3') Unidad de trabajo restituida.

MQRC_BUFFER_ERROR
(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_STRUC_FAILED
(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') La cabecera de lista de la estructura de recurso de acoplamiento se está utilizando.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') CICS ha rechazado la solicitud de espera.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_CORREL_ID_ERROR
(2207, X'89F') Error de identificador de correlación.

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_DB2_NOT_AVAILABLE
(2342, X'926') El subsistema DB2 no está disponible.

MQRC_GET_INHIBITED
(2016, X'7E0') Se han inhibido las obtenciones para la cola.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GMO_ERROR
(2186, X'88A') No es válida la estructura de las opciones de obtención de mensaje.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR
(2019, X'7E3') El manejador de objeto no es válido.

MQRC_INCONSISTENT_BROWSE
(2259, X'8D3') La especificación de examinar es incoherente.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_INVALID_MSG_UNDER_CURSOR
(2246, X'8C6') El mensaje bajo cursor no es válido para recuperación.

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') Las opciones de coincidencia no son válidas.

MQRC_MD_ERROR
(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MSG_ID_ERROR
(2206, X'89E') Error de identificador de mensaje.

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_TOKEN_ERROR
(2331, X'91B') El uso del símbolo de mensaje no es válido.

MQRC_NO_MSG_AVAILABLE
(2033, X'7F1') No hay ningún mensaje disponible.

MQRC_NO_MSG_UNDER_CURSOR
(2034, X'7F2') El cursor para examinar no está situado en el mensaje.

MQRC_NOT_OPEN_FOR_BROWSE
(2036, X'7F4') La cola no se ha abierto para examen.

MQRC_NOT_OPEN_FOR_INPUT
(2037, X'7F5') La cola no se ha abierto para entrada.

MQRC_OBJECT_CHANGED
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_DELETED
(2052, X'804') La cola se ha suprimido.

MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') La cola tiene un tipo de índice incorrecto.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SECOND_MARK_NOT_ALLOWED
(2062, X'80E') Ya hay un mensaje marcado.

MQRC_SIGNAL_OUTSTANDING
(2069, X'815') Símbolo pendiente para este descriptor de contexto.

MQRC_SIGNAL1_ERROR
(2099, X'833') Campo de señal no válido.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818') El punto de sincronización de soporte no está disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WAIT_INTERVAL_ERROR
(2090, X'82A') El intervalo de espera de MQGMO no es válido.

MQRC_WRONG_GMO_VERSION
(2256, X'8D0') Se ha suministrado una versión equivocada de MQGMO.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso

1. El mensaje recuperado se suprime normalmente de la cola. Esta supresión puede producirse como parte de la propia llamada MQGET o como parte de un punto de sincronización.

Las opciones de examen son: MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT y MQGMO_BROWSE_MSG_UNDER_CURSOR.

2. Si se especifica la opción MQGMO_LOCK con una de las opciones de examen, el mensaje examinado se bloquea para que sea visible sólo para este manejador.

   Si se especifica la opción MQGMO_UNLOCK, se desbloquea un mensaje anteriormente bloqueado. En este caso no se recupera ningún mensaje y los parámetros *MsgDesc*, *BufferLength*, *Buffer* y *DataLength* ni se comprueban ni se modifican.

3. Para las aplicaciones que emiten una llamada MQGET, se puede perder el mensaje recuperado si la aplicación termina de norma anormal o si se perdiera la conexión mientras se procesa la llamada. Este problema surge porque el substituto que se ejecuta en la misma plataforma que el gestor de colas que emite la llamada MQGET en nombre de la aplicación no puede detectar la pérdida de la aplicación hasta que el substituto esté a punto de devolver el mensaje a la aplicación *tras* eliminar el mensaje de la cola. Este problema puede ocurrir tanto para mensajes persistentes como para mensajes no persistentes.

   Para eliminar el riesgo de perder mensajes de esta forma, recupere siempre los mensajes en unidades de trabajo. Es decir, especificando la opción MQGMO_SYNCPOINT en la llamada MQGET y utilizando las llamadas MQCMIT o MQBACK para confirmar o para restituir la unidad de trabajo cuando finaliza el proceso del mensaje. Si se especifica MQGMO_SYNCPOINT y el cliente finaliza de forma anómala o la conexión se pierde, el sustituto restituye la unidad de trabajo en el gestor de colas y el mensaje se restablece en la cola. Para obtener más información sobre los puntos de sincronización, consulte Consideraciones sobre los puntos de sincronización en aplicaciones WebSphere MQ.

   Esta situación puede producirse con clientes de IBM WebSphere MQ , así como con aplicaciones que se ejecutan en la misma plataforma que el gestor de colas.

4. Si una aplicación pone una secuencia de mensajes en unadeterminada cola dentro de una única unidad de trabajo y, a continuación, confirma esa unidad de trabajo de forma satisfactoria, los mensajes están disponibles para la recuperación tal como se indica a continuación:

   - Si la cola es una cola *no compartida* (es decir, una cola local), todos los mensajes dentro de la unidad de trabajo están disponibles al mismo tiempo.
   - Si la cola es una cola *compartida*, los mensajes dentro de la unidad de trabajo están disponibles en el orden en que se colocaron, pero no todos al mismo tiempo. Cuando el sistema está muy cargado, es posible que el primer mensaje de la unidad de trabajo se recupere correctamente, pero que la llamada MQGET para el segundo o posteriores mensajes en la unidad de trabajo fallen con MQRC_NO_MSG_AVAILABLE. Si esto ocurre, la aplicación debe esperar un poco y luego volverá a intentar la operación.

5. Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes,el orden de dichos mensajes se conserva si se cumplen determinadas condiciones. Consulte Notas de uso de MQPUT para obtener detalles. Si las condiciones se cumplen, los mensajes se presentan a la aplicación receptora en el orden en que fueron enviados, si:

   - Sólo un receptor está obteniendo mensajes de la cola.

     Si hay dos o más aplicaciones que obtienen mensajes de la cola, debe ponerse de acuerdo con el remitente sobre el mecanismo que se utilizará para identificar los mensajes que pertenecen a una secuencia. Por ejemplo, es posible que el remitente establezca todos los campos *CorrelId* en los mensajes de una secuencia a un valor que era exclusivo para dicha secuencia de mensajes.
   - El receptor no cambia deliberadamente el orden de recuperación, por ejemplo, especificando un determinado *MsgId* o *CorrelId*.

   Si la aplicación emisora pone los mensajes como un grupo de mensajes, los mensajes se presentan a la aplicación receptora en el orden correcto si la aplicación receptora especifica la opción MQGMO_LOGICAL_ORDER en la llamada MQGET. Para obtener más información sobre los grupos de mensajes, consulte:

   - MQMD - Campo MsgFlags

- MQPMO_LOGICAL_ORDER
- MQGMO_LOGICAL_ORDER

Si el usuario está obteniendo mensajes de un grupo bajo el punto de sincronización, deben garantizar que se haya procesado todo el grupo antes de intentar finalizar la transacción.

6. Las aplicaciones deben comprobar el código de retorno MQFB_QUIT en el campo *Feedback* del parámetro *MsgDesc* y finalizar si encuentran este valor. Consulte MQMD - Campo Feedback para obtener más información.
7. Si la cola identificada por *Hobj* se ha abierto con la opción MQOO_SAVE_ALL_CONTEXT y el código de terminación de la llamada MQGET es MQCC_OK o MQCC_WARNING, el contexto asociado con el manejador de cola *Hobj* se establece en el contexto del mensaje que se ha recuperado (a menos que se establezca la opción MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT o MQGMO_BROWSE_MSG_UNDER_CURSOR, en cuyo caso el contexto se marca como no disponible).

   Puede utilizar el contexto guardado en una llamada MQPUT o MQPUT1 posterior especificando las opciones MQPMO_PASS_IDENTITY_CONTEXT o MQPMO_PASS_ALL_CONTEXT. Esto permite que el contexto del mensaje recibido se transfiera en su totalidad o en parte a otro mensaje (por ejemplo, cuando el mensaje se reenvía a otra cola). Para obtener más información sobre el contexto del mensaje, consulte Contexto de mensaje.
8. Si incluye la opción MQGMO_CONVERT en el parámetro *GetMsgOpts*, los datos de los mensajes de la aplicación se convierten a la representación solicitada por la aplicación receptora, antes de que los datos se coloquen en el parámetro *Buffer*:

   - El campo *Format* de la información de control del mensaje identifica la estructura de los datos de la aplicación y los campos *CodedCharSetId* y *Encoding* de la información de control del mensaje especifica el identificador del juego de caracteres y la codificación.
   - La aplicación que emite la llamada MQGET especifica en los campos *CodedCharSetId* y *Encoding* del parámetro *MsgDesc* el identificador del juego de caracteres y la codificación a los que se van a convertir los datos de los mensajes de la aplicación.

   Cuando hace falta convertir los datos del mensaje, la conversión se realiza mediante el propio gestor de colas o mediante una salida escrita por el usuario, en función del valor del campo *Format* de la información de control del mensaje:

   - Los siguientes nombres de formato son formatos que convierte el gestor de colas; estos formatos se denominan formatos "incorporados":
     - MQFMT_ADMIN
     - MQFMT_CICS (solamente para z/OS)
     - MQFMT_COMMAND_1
     - MQFMT_COMMAND_2
     - MQFMT_DEAD_LETTER_HEADER
     - MQFMT_DIST_HEADER
     - MQFMT_EVENT versión 1
     - MQFMT_EVENT version 2 (solamente z/OS)
     - MQFMT_IMS
     - MQFMT_IMS_VAR_STRING
     - MQFMT_MD_EXTENSION
     - MQFMT_PCF
     - MQFMT_REF_MSG_HEADER
     - MQFMT_RF_HEADER
     - MQFMT_RF_HEADER_2
     - MQFMT_STRING
     - MQFMT_TRIGGER

- MQFMT_WORK_INFO_HEADER (solamente para z/OS)
- MQFMT_XMIT_Q_HEADER

- El nombre de formato MQFMT_NONE es un valor especial que indica que la naturaleza de los datos del mensaje no está definida. Como consecuencia, el gestor de colas no intenta la conversión cuando se recupera el mensaje de la cola.

  Nota: Si se especifica MQGMO_CONVERT en la llamada MQGET para un mensaje que tiene un nombre de formato MQFMT_NONE y el juego de caracteres o la codificación del mensaje es distinto del especificado en el parámetro *MsgDesc*, se devuelve el mensaje en el parámetro *Buffer* (suponiendo que no haya más errores), pero la llamada se completa con el código de terminación MQCC_WARNING y el código de razón MQRC_FORMAT_ERROR.

  Puede utilizar MQFMT_NONE cuando la naturaleza de los datos del mensaje significa que no requieren conversión o cuando las aplicaciones emisoras o receptoras han acordado entre ellas la forma de envío de los datos del mensaje.
- Los demás nombres de formato pasan el mensaje a una salida escrita por el usuario para su conversión. La salida tiene el mismo nombre que el formato, aparte de las adiciones específicas del entorno. Los nombres de formato especificados por el usuario no deben empezar por las letras de WebSphere MQ.

  Consulte "salida de conversión de datos" en la página 888 para obtener más información sobre la salida de conversión de datos.

Los datos de usuario del mensaje se pueden convertir entre cualquier juego de caracteres y codificación admitidos. No obstante, tenga en cuenta que, si el mensaje contiene una o más estructuras de cabecera WebSphere MQ, el mensaje no puede convertirse de o a un juego de caracteres que tenga caracteres de doble byte o de varios bytes para cualquiera de los caracteres que son válidos en los nombres de colas. Si se intenta, se devuelven los códigos de razón MQRC_SOURCE_CCSID_ERROR o MQRC_TARGET_CCSID_ERROR y el mensaje sin convertir. El juego de caracteres Unicode UCS-2 es un ejemplo de este tipo de juego de caracteres.

En la devolución de la llamada MQGET, el código de razón siguiente indica que el mensaje se ha convertido correctamente:

- MQRC_NONE

El código de razón siguiente indica que el mensaje *puede* haberse convertido correctamente; la aplicación debe comprobar los campos *CodedCharSetId* y *Encoding* del parámetro *MsgDesc* para averiguar:

- MQRC_TRUNCATED_MSG_ACCEPTED

Los demás códigos de razón indican que el mensaje no se ha convertido.

Nota: La interpretación de este código de razón es correcta para las conversiones realizadas mediante una salida escrita por el usuario *únicamente* si la salida se ajusta a las directrices de proceso descritas en "salida de conversión de datos" en la página 888.

9. Si utilice la interfaz orientada a objetos para obtener mensajes, puede optar por no especificar un almacenamiento intermedio para guardar los datos de mensajes de una llamada MQGET. No obstante, en versiones anteriores de WebSphere MQ, era posible que MQGET fallara con el código de razón, incluso si no se había especificado ningún almacenamiento intermedio. En WebSphere MQ versión 7, cuando se recibe un mensaje utilizando una aplicación orientada a objetos sin limitar el tamaño del almacenamiento intermedio de recepción de mensajes, la aplicación no falla con MQRC_CONVERTED_MSG_TOO_BIG y recibe el mensaje convertido. Esto se cumple en los entornos siguientes:

   - .NET, incluidas aplicaciones plenamente gestionadas
   - C++
   - Java (WebSphere MQ classes for Java)

   Nota: Para todos los clientes, si el valor de *sharingConversations* es cero, el canal funciona como lo hacía antes de WebSphere MQ versión 7.0 y el manejo de mensajes vuelve

al comportamiento de la versión 6. En esta situación, si el almacenamiento intermedio es demasiado pequeño para recibir el mensaje convertido, se devuelve el mensaje sin convertir con el código de razón MQRC_CONVERTED_MSG_TOO_BIG. Para obtener más información sobre *sharingConversations*, consulte Utilización de conversaciones compartidas en una aplicación cliente.

10. Para los formatos incorporados, el gestor de colas puede realizar la *conversión predeterminada* de series de caracteres en el mensaje cuando se especifica la opción MQGMO_CONVERT. La conversión predeterminada permite que, para convertir datos de series de caracteres, el gestor de colas utilice un juego de caracteres predeterminado especificado por la instalación que se aproxima al juego de conjunto de caracteres real. Como consecuencia, la llamada MQGET puede realizarse con éxito con el código de terminación MQCC_OK, en vez de terminar con MQCC_WARNING y los códigos de razón MQRC_SOURCE_CCSID_ERROR o MQRC_TARGET_CCSID_ERROR.

    Nota: El resultado de utilizar un juego de caracteres aproximado para convertir datos de series de caracteres es que es posible que algunos se conviertan de forma incorrecta. Para evitarlo, en la serie utilice caracteres que sean comunes para el juego de caracteres actual y el juego de caracteres predeterminado.

    La conversión predeterminada se aplica a los datos de los mensajes de la aplicación y a los campos de tipo carácter de las estructuras MQMD y MQMDE:

    - La conversión predeterminada de los datos de los mensajes de la aplicación sólo se lleva a cabo si se cumplen *todas* las condiciones siguientes:
        - La aplicación especifica MQGMO_CONVERT.
        - El mensaje contiene datos que deben convertirse de o a un juego de caracteres que no se admite.
        - La conversión predeterminada no estaba activada cuando se ha instalado o reiniciado el gestor de colas.
    - La conversión predeterminada de los campos de caracteres de las estructuras MQMD y MQMDE se produce según convenga, si dicha conversión predeterminada está habilitada para el gestor de colas. La conversión se realiza aunque la aplicación no especifique la opción MQGMO_CONVERT en la llamada MQGET.

11. Para el lenguaje de programación Visual Basic, se aplican los puntos siguientes:

    - Si el tamaño del parámetro *Buffer* es menor que la longitud especificada por el parámetro *BufferLength*, la llamada falla con el código de razón MQRC_STORAGE_NOT_AVAILABLE.
    - El parámetro *Buffer* se declara como si fuese de tipo `String`. Si los datos que se van a recuperar de la cola no son de tipo `String` utilice elllamada MQGETAny en lugar de MQGET.

      La llamada MQGETAny tiene los mismos parámetros que la llamada MQGET, excepto que el parámetro *Buffer* se declara como si fuese de tipo Any, permitiendo la recuperación de cualquier tipo de datos. No obstante, esto significa que *Buffer* no se puede comprobar para asegurarse de que su tamaño sea al menos de *BufferLength* bytes.

12. No todas las opciones MQGET se admiten si la lectura anticipada está habilitada. En la tabla siguiente se indica qué opciones están permitidas y si se pueden modificar entre llamadas MQGET.

Tabla 566. Opciones MQGET permitidas cuando la lectura anticipada está habilitada

	Permitidas cuando la lectura anticipada está habilitada y se pueden modificar entre llamadas MQGET	Permitidas cuando la lectura anticipada está habilitada, pero no se pueden modificar entre llamadas MQGET[a]	Opciones MQGET que no están permitidas cuando la lectura anticipada está habilitada[b]
Valores MD de MQGET	MsgId[c] CorrelId[c]	Codificación CodedCharSetId	

Tabla 566. Opciones MQGET permitidas cuando la lectura anticipada está habilitada (continuación)			
	Permitidas cuando la lectura anticipada está habilitada y se pueden modificar entre llamadas MQGET	**Permitidas cuando la lectura anticipada está habilitada, pero no se pueden modificar entre llamadas MQGET[a]**	**Opciones MQGET que no están permitidas cuando la lectura anticipada está habilitada[b]**
Opciones MQGET MQGMO	MQGMO_WAIT MQGMO_NO_WAIT MQGMO_FAIL_IF_QUIESCING MQGMO_BROWSE_FIRST[d] MQGMO_BROWSE_NEXT[d] MQGMO_BROWSE_MESSAGE _UNDER_CURSOR[d]	MQGMO_SYNCPOINT_IF_PERSISTENT MQGMO_NO_SYNCPOINT MQGMO_ACCEPT_TRUNCATED_MSG MQGMO_CONVERT MQGMO_LOGICAL_ORDER 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 MQGMO_SYNCPOINT MQGMO_MARK_SKIP _BACKOUT MQGMO_MSG_UNDER _CURSOR[d] MQGMO_LOCK MQGMO_UNLOCK
Valores MQGMO		MsgHandle	

a. Si estas opciones se modifican entre llamadas MQGET se devuelve el código de razón MQRC_OPTIONS_CHANGED.

b. Si estas opciones se especifican en la primera llamada MQGET, la lectura anticipada está inhabilitada. Si estas opciones se especifican en una llamada MQGET posterior, se devuelve el código de razón MQRC_OPTIONS_ERROR.

c. Las aplicaciones cliente han de tener presente que si los valores MsgId y CorrelId se modifican entre llamadas MQGET, es posible que los mensajes con los valores anteriores ya se hayan enviado al cliente y permanezcan en el almacenamiento intermedio de lectura anticipada del cliente hasta que se consuman (o se depuren automáticamente).

d. La primera llamada MQGET determina si los mensajes se han de examinar u obtener de una cola cuando la lectura anticipada está habilitada. Si la aplicación intenta utilizar una combinación de opciones de examen y de obtención, se devuelve el código de razón MQRC_OPTIONS_CHANGED.

e. MQGMO_MSG_UNDER_CURSOR no es posible con lectura anticipada. Los mensajes se pueden examinar u obtener cuando la lectura anticipada está habilitada, pero no una combinación de ambos.

13. Las aplicaciones pueden obtener mensajes sin confirmar de forma destructiva sólo si los mensajes se han colocado en la misma unidad de trabajo local desde la que se obtienen. Las aplicaciones no pueden obtener mensajes sin confirmar de forma no destructiva.
14. Los mensajes bajo un cursor para examinar se pueden recuperar en una unidad de trabajo. No es posible recuperar un mensaje no confirmado de esta forma.

Invocación de C

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

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
CALL 'MQGET' USING HCONN, HOBJ, MSGDESC, GETMSGOPTS, BUFFERLENGTH,
BUFFER, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

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

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación de High Level Assembler

```
         CALL MQGET,(HCONN,HOBJ,MSGDESC,GETMSGOPTS,BUFFERLENGTH,
               BUFFER,DATALENGTH,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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
```

Invocación en Visual Basic

```
MQGET Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer,
DataLength, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
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-Consultar atributos de objeto

La llamada MQINQ devuelve una matriz de enteros y un conjunto de series de caracteres que contienen los atributos de un objeto.

Son válidos los siguientes tipos de objeto:

- Gestor de colas
- Cola
- Lista de nombres
- Definición de proceso

Sintaxis

MQINQ (*Hconn*, *Hobj*, *SelectorCount*, *Selectores*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *CompCode*, *Reason*)

Parámetros

Hconn
Tipo: MQHCONN -entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En aplicaciones z/OS para CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, se puede omitir la llamada MQCONN y se puede especificar el valor siguiente para *Hconn*:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Hobj
Tipo: MQHOBJ -entrada

Este descriptor de contexto representa el objeto (de cualquier tipo) con los atributos necesarios. El descriptor de contexto debe ser devuelto por una llamada MQOPEN anterior que haya especificado la opción MQOO_INQUIRE .

SelectorCount
Tipo: MQLONG -entrada

Este es el recuento de selectores que se proporcionan en la matriz *Selectors* . Es el número de atributos que se van a devolver. Cero es un valor válido. El número máximo permitido es 256.

Selectores
Tipo: MQLONG × *SelectorCount* -entrada

Esta es una matriz de selectores de atributos *SelectorCount* ; cada selector identifica un atributo (entero o carácter) con un valor que es necesario.

Cada selector debe ser válido para el tipo de objeto que *Hobj* representa, de lo contrario la llamada falla con el código de terminación MQCC_FAILED y el código de razón MQRC_SELECTOR_ERROR.

En el caso especial de las colas:

- Si el selector no es válido para colas de cualquier tipo, la llamada falla con el código de terminación MQCC_FAILED y el código de razón MQRC_SELECTOR_ERROR.
- Si el selector sólo se aplica a colas de tipos distintos al tipo del objeto, la llamada se realiza correctamente con el código de terminación MQCC_WARNING y el código de razón MQRC_SELECTOR_NOT_FOR_TYPE.
- Si la cola que se está consultando es una cola de clúster, los selectores que son válidos dependen de cómo se haya resuelto la cola; consulte "Notas de uso" en la página 700 para obtener más detalles.

Los selectores se pueden especificar en cualquier orden. Los valores de atributo que corresponden a selectores de atributos enteros (selectoresMQIA_*) se devuelven en *IntAttrs* en el mismo orden en el que aparecen estos selectores en *Selectors*. Los valores de atributo que corresponden a selectores de atributo de carácter (selectoresMQCA_*) se devuelven en *CharAttrs* en el mismo orden en el que se producen dichos selectores. Los selectores MQIA_* se pueden intercalar con los selectores MQCA_* ; sólo es importante el orden relativo dentro de cada tipo.

Nota:

1. Los selectores de atributo de entero y carácter se asignan dentro de dos rangos diferentes; los selectores MQIA_* residen dentro del rango de MQIA_FIRST a MQIA_LAST, y los selectores MQCA_* dentro del rango de MQCA_FIRST a MQCA_LAST.

   Para cada rango, las constantes MQIA_LAST_USED y MQCA_LAST_USEDdefinen el valor más alto que acepta el gestor de colas.
2. Si todos los selectores MQIA_* aparecen en primer lugar, se pueden utilizar los mismos números de elemento para direccionar los elementos correspondientes en las matrices *Selectors* y *IntAttrs* .
3. Si el parámetro *SelectorCount* es cero, no se hace referencia a *Selectors* . En este caso, la dirección de parámetro que pasan los programas escritos en C o S/390 assembler podría ser nula.

Los atributos que se pueden consultar se listan en las tablas siguientes. Para los selectores MQCA_* , la constante que define la longitud en bytes de la serie resultante en *CharAttrs* se proporciona entre paréntesis.

Las tablas siguientes listan los selectores, por objeto, en orden alfabético, como se indica a continuación:

- Selectores de atributos de Tabla 567 en la página 687 MQINQ para colas
- Selectores de atributos de Tabla 568 en la página 690 MQINQ para listas de nombres
- Selectores de atributos de Tabla 569 en la página 690 MQINQ para definiciones de proceso
- Selectores de atributos de Tabla 570 en la página 691 MQINQ para el gestor de colas

Todos los selectores están soportados en todas las plataformas IBM WebSphere MQ , excepto donde se indica en la columna **Nota** como se indica a continuación:

NOz/OS
Soportado en todas las plataformas **excepto** z/OS

z/OS
solo soportado en z/OS

Tabla 567. Selectores de atributos de MQINQ para colas

Selector	Longitud del campo	Descripción	Nota
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Fecha de la modificación más reciente	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Hora de la modificación más reciente	
MQCA_BACKOUT_REQ_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de reposición en cola de restitución excesivo	

Tabla 567. Selectores de atributos de MQINQ para colas (continuación)

Selector	Longitud del campo	Descripción	Nota
MQCA_BASE_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de la cola en la que se resuelve el alias	
MQCA_CF_STRUC_NAME	MQ_CF_STRUC_NAME_LENGTH	Nombre de estructura de recurso de acoplamiento	z/OS
MQCA_CLUS_CHL_NAME	MQ_CHANNEL_NAME_LENGTH	Nombre del canal de clúster emisor que utiliza esta cola como cola de transmisión.	NOz/OS
MQCA_CLUSTER_NAME	MQ_CLUSTER_NAME_LENGTH	Nombre del clúster	
MQCA_CLUSTER_NAMELIST	MQ_NAMELIST_NAME_LENGTH	Lista de nombres de clúster	
MQCA_CREATION_DATE	MQ_CREATION_DATE_LENGTH	Fecha de creación de cola	
MQCA_CREATION_TIME	MQ_CREATION_TIME_LENGTH	Tiempo de creación de cola	
MQCA_INITIATION_Q_NAME	MQ_Q_NAME_LENGTH	Nombre cola iniciación	
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	Nombre de definición de proceso	
MQCA_Q_DESC	MQ_Q_DESC_LENGTH	Descripción de la cola	
MQCA_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de cola	
MQCA_REMOTE_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	Nombre del gestor de colas remoto	
MQCA_REMOTE_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de la cola remota tal como se conoce en el gestor de colas remoto	
MQCA_STORAGE_CLASS	MQ_STORAGE_CLASS_LENGTH	Nombre de clase de almacenamiento	z/OS
MQCA_TRIGGER_DATA	MQ_TRIGGER_DATA_LENGTH	Datos desencadenantes	
MQCA_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de cola de transmisión	
MQIA_ACCOUNTING_Q	MQLONG	Controla la recopilación de datos de contabilidad para cola	NOz/OS
MQIA_BACKOUT_THRESHOLD	MQLONG	Umbral de restituciones	
MQIA_CLWL_Q_PRIORITY	MQLONG	Prioridad de cola	
MQIA_CLWL_Q_RANK	MQLONG	Rango de cola	
MQIA_CLWL_USEQ	MQLONG	Utilizar colas remotas	
MQIA_CURRENT_Q_DEPTH	MQLONG	Número de mensajes en cola	
MQIA_DEF_BIND	MQLONG	Enlace predeterminado	
MQIA_DEF_INPUT_OPEN_OPTION	MQLONG	Opción de apertura para entrada predeterminada	
MQIA_DEF_PERSISTENCE	MQLONG	Persistencia de mensajes predeterminada	

Tabla 567. Selectores de atributos de MQINQ para colas (continuación)

Selector	Longitud del campo	Descripción	Nota
MQIA_DEF_PRIORITY	MQLONG	Prioridad de mensajes predeterminada	
MQIA_DEFINITION_TYPE	MQLONG	Tipo de definición de cola	
MQIA_DIST_LISTS	MQLONG	Soporte de lista de distribución	NOz/OS
MQIA_HARDEN_GET_BACKOUT	MQLONG	Si se debe reforzar el recuento de restituciones	
MQIA_INDEX_TYPE	MQLONG	Tipo de índice mantenido para cola	z/OS
MQIA_INHIBIT_GET	MQLONG	Si se permiten operaciones get	
MQIA_INHIBIT_PUT	MQLONG	Si se permiten operaciones de colocación	
MQIA_MAX_MSG_LENGTH	MQLONG	Longitud máxima de mensaje	
MQIA_MAX_Q_DEPTH	MQLONG	Número máximo de mensajes permitidos en la cola	
MQIA_MSG_DELIVERY_SEQUENCE	MQLONG	Si la prioridad del mensaje es relevante	
MQIA_NPM_CLASS	MQLONG	Nivel de fiabilidad para mensajes no persistentes	
MQIA_OPEN_INPUT_COUNT	MQLONG	Número de llamadas MQOPEN que tienen la cola abierta para entrada	
MQIA_OPEN_OUTPUT_COUNT	MQLONG	Número de llamadas MQOPEN que tienen la cola abierta para salida	
MQIA_PROPERTY_CONTROL	MQLONG	Atributo de control de propiedad	
MQIA_Q_DEPTH_HIGH_EVENT	MQLONG	Atributo de control para sucesos de profundidad de cola alta	NOz/OS
MQIA_Q_DEPTH_HIGH_LIMIT	MQLONG	Límite alto para profundidad de cola	NOz/OS
MQIA_Q_DEPTH_LOW_EVENT	MQLONG	Atributo de control para sucesos de profundidad de cola baja	NOz/OS
MQIA_Q_DEPTH_LOW_LIMIT	MQLONG	Límite bajo para profundidad de cola	NOz/OS
MQIA_Q_DEPTH_MAX_EVENT	MQLONG	Atributo de control para sucesos máximos de profundidad de cola	NOz/OS
MQIA_Q_SERVICE_INTERVAL	MQLONG	Límite para intervalo de servicio de cola	NOz/OS
MQIA_Q_SERVICE_INTERVAL_EVENT	MQLONG	Atributo de control para sucesos de intervalo de servicio de cola	NOz/OS
MQIA_Q_TYPE	MQLONG	Tipo de cola	
MQIA_QSG_DISP	MQLONG	Disposición del grupo de compartición de colas	z/OS
MQIA_RETENTION_INTERVAL	MQLONG	Intervalo de retención de cola	

Tabla 567. Selectores de atributos de MQINQ para colas (continuación)

Selector	Longitud del campo	Descripción	Nota
MQIA_SCOPE	MQLONG	Ámbito de definición de cola	NOz/OS
MQIA_SHAREABILITY	MQLONG	Si la cola se puede compartir para entrada	
MQIA_STATISTICS_Q	MQLONG	Controla la recopilación de datos estadísticos para la cola	NOz/OS
MQIA_TRIGGER_CONTROL	MQLONG	Activar control	
MQIA_TRIGGER_DEPTH	MQLONG	Profundidad de desencadenante	
MQIA_TRIGGER_MSG_PRIORITY	MQLONG	Prioridad de mensaje de umbral para desencadenantes	
MQIA_TRIGGER_TYPE	MQLONG	Tipo de desencadenante	
MQIA_USAGE	MQLONG	Utilización	

Tabla 568. Selectores de atributos de MQINQ para listas de nombres

Selector	Longitud del campo	Descripción	Nota
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Fecha de la modificación más reciente	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Hora de la modificación más reciente	
MQCA_NAMELIST_DESC	MQ_NAMELIST_DESC_LENGTH	Descripción de lista de nombres	
MQCA_NAMELIST_NAME	MQ_NAMELIST_NAME_LENGTH	Nombre del objeto de lista de nombres	
MQIA_NAMELIST_TYPE	MQLONG	Tipo de lista de nombres	z/OS
MQCA_NAMES	MQ_Q_NAME_LENGTH × Número de nombres en la lista	Nombres en la lista de nombres	
MQIA_NAME_COUNT	MQLONG	Número de nombres en la lista de nombres	
MQIA_QSG_DISP	MQLONG	Disposición del grupo de compartición de colas	z/OS

Tabla 569. Selectores de atributos de MQINQ para definiciones de proceso

Selector	Longitud del campo	Descripción	Nota
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Fecha de la modificación más reciente	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Hora de la modificación más reciente	
MQCA_APPL_ID	MQ_PROCESS_APPL_ID_LENGTH	Identificador de aplicación	
MQCA_ENV_DATA	MQ_PROCESS_ENV_DATA_LENGTH	Datos de entorno	

Tabla 569. Selectores de atributos de MQINQ para definiciones de proceso (continuación)			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQCA_PROCESS_DESC	MQ_PROCESS_DESC_LENGTH	Descripción de la definición de proceso	
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	Nombre de definición de proceso	
MQCA_USER_DATA	MQ_PROCESS_USER_DATA_LENGTH	Datos de usuario	
MQIA_APPL_TYPE	MQLONG	Tipo de aplicación	
MQIA_QSG_DISP	MQLONG	Disposición del grupo de compartición de colas	z/OS

Tabla 570. Selectores de atributos de MQINQ para el gestor de colas			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Fecha de la modificación más reciente	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Hora de la modificación más reciente	
MQCA_CHANNEL_AUTO_DEF_EXIT	MQ_EXIT_NAME_LENGTH	Nombre de salida de definición de canal automática	
MQCA_CHINIT_SERVICE_PARM		Reservado para uso de IBM	
MQCA_CLUSTER_WORKLOAD_DATA	MQ_EXIT_DATA_LENGTH	Datos pasados a salida de carga de trabajo de clúster	
MQCA_CLUSTER_WORKLOAD_EXIT	MQ_EXIT_NAME_LENGTH	Nombre de salida de carga de trabajo de clúster	
MQCA_COMMAND_INPUT_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de cola de entrada de mandatos del sistema	
MQCA_DEAD_LETTER_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de la cola de mensajes no entregados	
MQCA_DEF_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	Nombre de cola de transmisión predeterminado	
MQCA_DNS_GROUP	MQ_DNS_GROUP_NAME_LENGTH	Nombre del grupo para el escucha TCP que maneja las transmisiones de entrada para que se una el grupo de compartición de colas. El nombre se aplica cuando se utiliza Workload Manager Dynamic Domain Name Services.	z/OS
MQCA_IGQ_USER_ID	MQ_USER_ID_LENGTH	Identificador de usuario de transferencia a colas dentro del grupo	z/OS
MQCA_INSTALLATION_DESC	MQ_INSTALLATION_DESC_LENGTH	Descripción de la instalación asociada	No z/OS. NOIBM i

Tabla 570. Selectores de atributos de MQINQ para el gestor de colas (continuación)			
Selector	**Longitud del campo**	**Descripción**	**Nota**
MQCA_INSTALLATION_NAME	MQ_INSTALLATION_NAME_LENGTH	Nombre de la instalación asociada con el gestor de colas	No z/OS. NOIBM i
MQCA_INSTALLATION_PATH	MQ_INSTALLATION_PATH_LENGTH	Vía de acceso donde está instalado elIBM WebSphere MQ asociado	No z/OS. NOIBM i
MQCA_LU_GROUP_NAME	MQ_LU_NAME_LENGTH	Nombre de LU genérico para el escucha de LU 6.2 que maneja las transmisiones de entrada para que las utilice el grupo de compartición de colas	z/OS
MQCA_LU_NAME	MQ_LU_NAME_LENGTH	Nombre de la LU a utilizar para las transmisiones de LU de salida 6.2 . Establezca este nombre en la misma LU que utiliza el escucha para las transmisiones de entrada	z/OS
MQCA_LU62_ARM_SUFFIX	MQ_ARM_SUFFIX_LENGTH	Sufijo del SYS1.PARMLIB miembro APPCPM*xx*, que nombra el LUADD para este iniciador de canal	z/OS
MQCA_PARENT	MQ_Q_MGR_NAME_LENGTH	Nombre de un gestor de colas conectado jerárquicamente que está nominado como padre de este gestor de colas	
MQCA_Q_MGR_DESC	MQ_Q_MGR_DESC_LENGTH	Descripción del gestor de colas	
MQCA_Q_MGR_IDENTIFIER	MQ_Q_MGR_IDENTIFIER_LENGT	Identificador de gestor de colas (H)	
MQCA_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	Nombre del gestor de colas local	
MQCA_QSG_NAME	MQ_QSG_NAME_LENGTH	Nombre del grupo de compartición de colas	z/OS
MQCA_REPOSITORY_NAME	MQ_CLUSTER_NAME_LENGTH	Nombre del clúster para el que el gestor de colas proporciona servicios de repositorio	
MQCA_REPOSITORY_NAMELIST	MQ_NAMELIST_NAME_LENGTH	Nombre del objeto de lista de nombres que contiene nombres de clústeres para los que el gestor de colas proporciona servicios de repositorio	
MQCA_TCP_NAME	MQ_TCP_NAME_LENGTH	Nombre del sistema TCP/IP que está utilizando	z/OS
MQIA_ACCOUNTING_CONN_OVERRIDE	MQLONG	Alterar temporalmente valores de contabilidad	NOz/OS

Tabla 570. Selectores de atributos de MQINQ para el gestor de colas (continuación)

Selector	Longitud del campo	Descripción	Nota
MQIA_ACCOUNTING_INTERVAL	MQLONG	Frecuencia con la que se escriben registros de contabilidad intermedios	NOz/OS
MQIA_ACCOUNTING_MQI	MQLONG	Controla la recopilación de información de contabilidad para datos MQI	NOz/OS
MQIA_ACCOUNTING_Q	MQLONG	Controla la recopilación de información de contabilidad para colas	NOz/OS
MQIA_ACTIVE_CHANNELS	MQLONG	Número máximo de canales que pueden estar activos en cualquier momento	z/OS
MQIA_ADOPTNEWMCA_CHECK	MQLONG	Elementos que se comprueban para determinar si se debe adoptar un MCA. La comprobación se realiza cuando se detecta un nuevo canal de entrada que tiene el mismo nombre que un MCA que ya está activo.	z/OS
MQIA_ADOPTNEWMCA_INTERVAL	MQLONG	Cantidad de tiempo, en segundos, que el nuevo canal espera a que finalice el canal huérfano	NOz/OS
MQIA_ADOPTNEWMCA_TYPE	MQLONG	Indica si se debe reiniciar automáticamente una instancia huérfana de un MCA de un tipo de canal determinado cuando se detecta una nueva solicitud de canal de entrada que coincide con los parámetros AdoptNewMCACheck	z/OS
MQIA_AUTHORITY_EVENT	MQLONG	Atributo de control para sucesos de autorización	NOz/OS
MQIA_BRIDGE_EVENT	MQLONG	Atributo de control para sucesos de puente IMS	z/OS
MQIA_CHANNEL_AUTO_DEF	MQLONG	Atributo de control para definición de canal automática	NOz/OS
MQIA_CHANNEL_AUTO_DEF_EVENT	MQLONG	Atributo de control para sucesos de definición de canal automática	NOz/OS
MQIA_CHANNEL_EVENT	MQLONG	Atributo de control para sucesos de canal	
MQIA_CHINIT_ADAPTERS	MQLONG	Número de subtareas de adaptador a utilizar para procesar llamadas de IBM WebSphere MQ	z/OS
MQIA_CHINIT_DISPATCHERS	MQLONG	Número de asignadores a utilizar para el iniciador de canal	z/OS
MQIA_CHINIT_TRACE_AUTO_START	MQLONG	Indica si se debe iniciar automáticamente el rastreo de iniciador de canal	z/OS

Tabla 570. Selectores de atributos de MQINQ para el gestor de colas (continuación)

Selector	Longitud del campo	Descripción	Nota
MQIA_CHINIT_TRACE_TABLE_SIZE	MQLONG	Tamaño del espacio de datos de rastreo (en MB) del iniciador de canal	z/OS
MQIA_CLUSTER_WORKLOAD_LENGTH	MQLONG	Longitud de carga de trabajo de clúster.	
MQIA_CLWL_MRU_CHANNELS	MQLONG	Número de canales utilizados más recientemente para el equilibrio de carga de trabajo de clúster	
MQIA_CLWL_USEQ	MQLONG	Utilizar colas remotas	
MQIA_CODED_CHAR_SET_ID	MQLONG	Identificador de juego de caracteres codificado	
MQIA_COMMAND_EVENT	MQLONG	Atributo de control para sucesos de mandato	
MQIA_COMMAND_LEVEL	MQLONG	Nivel de mandatos soportado por el gestor de colas	
MQIA_CONFIGURATION_EVENT	MQLONG	Atributo de control para sucesos de configuración	NOz/OS
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	MQLONG	Tipo de cola de transmisión predeterminada que se debe utilizar para canales de clúster emisor.	NOz/OS
MQIA_DIST_LISTS	MQLONG	Soporte de lista de distribución	NOz/OS
MQIA_DNS_WLM	MQLONG	Si el escucha TCP que maneja las transmisiones de entrada para el grupo de compartición de colas se registra en el Gestor de carga de trabajo para Dynamic Domain Name Services	z/OS
MQIA_EXPIRY_INTERVAL	MQLONG	Intervalo entre exploraciones de mensajes caducados	z/OS
MQIA_GROUP_UR	MQLONG	Atributo de control para determinar si las unidades de recuperación GROUP están habilitadas para este gestor de colas. La disposición de unidad de recuperación GROUP sólo está disponible si el gestor de colas es miembro de un grupo de compartición de colas	z/OS
MQIA_IGQ_PUT_AUTHORITY	MQLONG	Autorización de transferencia a colas dentro del grupo	z/OS
MQIA_INHIBIT_EVENT	MQLONG	Atributo de control para sucesos de inhibición	NOz/OS
MQIA_INTRA_GROUP_QUEUING	MQLONG	Soporte de transferencia a colas dentro del grupo	z/OS

Tabla 570. Selectores de atributos de MQINQ para el gestor de colas (continuación)

Selector	Longitud del campo	Descripción	Nota
MQIA_LISTENER_TIMER	MQLONG	El intervalo de tiempo (en segundos) entre IBM WebSphere MQ intenta reiniciar el escucha si APPC o TCP/IP han fallado.	z/OS
MQIA_LOCAL_EVENT	MQLONG	Atributo de control para sucesos locales	NOz/OS
MQIA_LOGGER_EVENT	MQLONG	Atributo de control para sucesos de inhibición	NOz/OS
MQIA_LU62_CHANNELS	MQLONG	Número máximo de canales que pueden ser actuales, o clientes que se pueden conectar, utilizando el protocolo de transmisión LU 6.2	z/OS
MQIA_MSG_MARK_BROWSE_INTERVAL	MQLONG	Intervalo de tiempo (en milisegundos) después del cual el gestor de colas puede eliminar automáticamente una marca de los mensajes de examen. **Atención:** No debe establecer este valor por debajo del valor predeterminado de 5000.	
MQIA_MAX_CHANNELS	MQLONG	Número máximo de canales que pueden ser actuales (incluidos los canales de conexión de servidor con clientes conectados)	z/OS
MQIA_MAX_HANDLES	MQLONG	Número máximo de descriptores de contexto	
MQIA_MAX_MSG_LENGTH	MQLONG	Longitud máxima de mensaje	
MQIA_MAX_PRIORITY	MQLONG	Prioridad máxima	
MQIA_MAX_UNCOMMITTED_MSGS	MQLONG	Número máximo de mensajes no confirmados dentro de una unidad de trabajo	
MQIA_OUTBOUND_PORT_MAX	MQLONG	Con MQIA_OUTBOUND_PORT_MIN, define el rango de números de puerto a utilizar al enlazar canales de salida	z/OS
MQIA_OUTBOUND_PORT_MIN	MQLONG	Con MQIA_OUTBOUND_PORT_MAX, define el rango de números de puerto a utilizar al enlazar canales de salida	z/OS
MQIA_PERFORMANCE_EVENT	MQLONG	Atributo de control para sucesos de rendimiento	NOz/OS
MQIA_PLATFORM	MQLONG	Plataforma en la que reside el gestor de colas	
MQIA_PROT_POLICY_CAPABILITY	MQLONG	Indica si las prestaciones de seguridad de WebSphere MQ Advanced Message Security están disponibles para un gestor de colas.	

Tabla 570. Selectores de atributos de MQINQ para el gestor de colas (continuación)

Selector	Longitud del campo	Descripción	Nota
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	MQLONG	El número de intentos de volver a procesar un mensaje de mandato fallido bajo el punto de sincronización	
MQIA_PUBSUB_MODE	MQLONG	Si el motor de publicación/suscripción y la interfaz de publicación/suscripción en cola se están ejecutando. Las aplicaciones para publicar o suscribirse utilizando la interfaz de programación de aplicaciones requieren el motor de publicación/suscripción. Las colas supervisadas por la interfaz de publicación/suscripción en cola requieren que la interfaz de publicación/suscripción en cola esté en ejecución.	
MQIA_PUBSUB_NP_MSG	MQLONG	Si se debe descartar (o conservar) un mensaje de entrada no entregado	
MQIA_PUBSUB_NP_RESP	MQLONG	Controla el comportamiento de los mensajes de respuesta no entregados	
MQIA_PUBSUB_SYNC_PT	MQLONG	Si sólo se procesan los mensajes persistentes (o todos) bajo el punto de sincronización	
MQIA_QMGR_CFCONLOS	MQLONG	Especifica la acción que se debe realizar cuando el gestor de colas pierde la conectividad con la estructura de administración o con cualquier estructura CF con CFCONLOS establecido en ASQMGR	z/OS
MQIA_RECEIVE_TIMEOUT	MQLONG	Aproximadamente el tiempo que un canal TCP/IP espera para recibir datos, incluidas las pulsaciones, de su asociado, antes de volver al estado inactivo. El valor es numérico, calificado por MQIA_RECEIVE_TIMEOUT_TYPE.	z/OS
MQIA_RECEIVE_TIMEOUT_MIN	MQLONG	Tiempo mínimo que un canal TCP/IP espera para recibir datos, incluidas las pulsaciones, de su asociado, antes de volver al estado inactivo	z/OS
MQIA_RECEIVE_TIMEOUT_TYPE	MQLONG	Aproximadamente el tiempo que un canal TCP/IP espera para recibir datos, incluidas las pulsaciones, de su asociado, antes de volver al estado inactivo. MQIA_RECEIVE_TIMEOUT_TYPE es el calificador aplicado a MQIA_RECEIVE_TIMEOUT.	z/OS

Tabla 570. Selectores de atributos de MQINQ para el gestor de colas (continuación)

Selector	Longitud del campo	Descripción	Nota
MQIA_REMOTE_EVENT	MQLONG	Atributo de control para sucesos remotos	NOz/OS
MQIA_SECURITY_CASE	MQLONG	Caso de perfiles de seguridad	z/OS
MQIA_SSL_EVENT	MQLONG	Atributo de control para sucesos de canal	
MQIA_SSL_FIPS_REQUIRED	MQLONG	Utilizar sólo algoritmos certificados por FIPS para criptografía	
MQIA_SSL_RESET_COUNT	MQLONG	Cuenta restablecimiento clave SSL	
MQIA_START_STOP_EVENT	MQLONG	Atributo de control para sucesos de detención de inicio	NOz/OS
MQIA_STATISTICS_AUTO_CLUSSDR	MQLONG	Controla la recopilación de información de supervisión de estadísticas para canales emisores de clúster	NOz/OS
MQIA_STATISTICS_CHANNEL	MQLONG	Controla la recopilación de datos estadísticos para canales	NOz/OS
MQIA_STATISTICS_INTERVAL	MQLONG	Frecuencia con la que se escriben datos de supervisión de estadísticas	NOz/OS
MQIA_STATISTICS_MQI	MQLONG	Controla la recopilación de información de supervisión de estadísticas para el gestor de colas	NOz/OS
MQIA_STATISTICS_Q	MQLONG	Controla la recopilación de datos de estadísticas para colas	NOz/OS
MQIA_SYNCPOINT	MQLONG	disponibilidad de punto de sincronización	
MQIA_TCP_CHANNELS	MQLONG	Número máximo de canales que pueden ser actuales, o clientes que se pueden conectar, utilizando el protocolo de transmisión TCP/IP	z/OS
MQIA_TCP_KEEP_ALIVE	MQLONG	Indica si se debe utilizar el recurso TCP KEEPALIVE para comprobar que el otro extremo de la conexión sigue estando disponible	z/OS
MQIA_TCP_STACK_TYPE	MQLONG	Si el iniciador de canal puede utilizar sólo el espacio de direcciones TCP/IP especificado en TCPNAME, o puede enlazarse opcionalmente a cualquier dirección TCP/IP seleccionada	z/OS
MQIA_TRACE_ROUTE_RECORDING	MQLONG	Controla el registro de la información de ruta de rastreo	z/OS
MQIA_TREE_LIFE_TIME	MQLONG	Duración de los temas no administrativos no utilizados	
MQIA_TRIGGER_INTERVAL	MQLONG	Activar intervalo	

IntAttrCount
Tipo: MQLONG -entrada

Es el número de elementos de la matriz *IntAttrs* . Cero es un valor válido.

Si IntAttrCount es como mínimo el número de selectores MQIA_* en el parámetro *Selectors* , se devuelven todos los atributos enteros solicitados.

IntAttrs
Tipo: MQLONG ×*IntAttrCount* -salida

Esta es una matriz de valores de atributo de entero de *IntAttrCount* .

Los valores de atributo de entero se devuelven en el mismo orden que los selectores MQIA_* en el parámetro *Selectors* . Si la matriz contiene más elementos que el número de selectores MQIA_* , el exceso de elementos no se modificará.

Si *Hobj* representa una cola, pero un selector de atributos no se aplica a ese tipo de cola, se devuelve el valor específico MQIAV_NOT_APPLICABLE . Se devuelve para el elemento correspondiente en la matriz *IntAttrs* .

Si el parámetro *IntAttrCount* o *SelectorCount* es cero, no se hace referencia a *IntAttrs* . En este caso, la dirección de parámetro que pasan los programas escritos en C o S/390 assembler podría ser nula.

CharAttrLongitud
Tipo: MQLONG -entrada

Es la longitud en bytes del parámetro *CharAttrs* .

CharAttrLength debe ser como mínimo la suma de las longitudes de los atributos de caracteres solicitados (consulte *Selectors*). Cero es un valor válido.

CharAttrs
Tipo: MQCHAR × *CharAttrLength* -output

Es el almacenamiento intermedio en el que se devuelven los atributos de carácter, concatenados entre sí. La longitud del almacenamiento intermedio la proporciona el parámetro *CharAttrLength* .

Los atributos de carácter se devuelven en el mismo orden que los selectores MQCA_* en el parámetro *Selectors* . La longitud de cada serie de atributo es fija para cada atributo (consulte *Selectors*), y el valor que contiene se rellena a la derecha con espacios en blanco si es necesario. Puede proporcionar un almacenamiento intermedio mayor que el necesario para contener todos los atributos de caracteres solicitados y el relleno. Los bytes más allá del último valor de atributo devuelto no se modifican.

Si *Hobj* representa una cola, pero un selector de atributos no se aplica a ese tipo de cola, se devuelve una serie de caracteres que consta totalmente de asteriscos (*). El asterisco se devuelve como el valor de ese atributo en *CharAttrs*.

Si el parámetro *CharAttrLength* o *SelectorCount* es cero, no se hace referencia a *CharAttrs* . En este caso, la dirección de parámetro que pasan los programas escritos en C o S/390 assembler podría ser nula.

CompCode
Tipo: MQLONG -salida

El código de terminación:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay razón para informar.

Si *CompCode* es MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT
(2008, X'7D8') No se permite espacio suficiente para los atributos de tipo carácter.

MQRC_INT_ATTR_COUNT_TOO_SMALL
(2022, X'7E6') No se permite suficiente espacio para atributos enteros.

MQRC_SELECTOR_NOT_FOR_TYPE
(2068, X'814') El selector no es aplicable al tipo de cola.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') Ha fallado la salida de API.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se puede cargar la salida de API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario y de inicio difieren.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_CF_STRUC_FAILED
(2373, X'945') La estructura del recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') Estructura de recurso de acoplamiento en uso.

MQRC_CHAR_ATTR_LENGTH_ERROR
(2006, X'7D6') La longitud de los atributos de caracteres no es válida.

MQRC_CHAR_ATTRS_ERROR
(2007, X'7D7') Serie de atributos de tipo carácter no válida.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Solicitud de espera rechazada por CICS.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No autorizado para la conexión.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Se está cerrando la conexión.

MQRC_HCONN_ERROR
(2018, X'7E2') El descriptor de conexión no es válido.

MQRC_HOBJ_ERROR
(2019, X'7E3') Descriptor de contexto de objeto no válido.

MQRC_INT_ATTR_COUNT_ERROR
(2021, X'7E5') Recuento de atributos enteros no válido.

MQRC_INT_ATTRS_ARRAY_ERROR
(2023, X'7E7') La matriz de atributos enteros no es válida.

MQRC_NOT_OPEN_FOR_INQUIRE
(2038, X'7F6') Cola no abierta para consulta.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Definición de objeto cambiada desde su apertura.

MQRC_OBJECT_DAMAGED
(2101, X'835') Objeto dañado.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos de conjunto de páginas.

MQRC_Q_DELETED
(2052, X'804') Cola suprimida.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Gestor de colas no disponible para la conexión.

MQRC_Q_MGR_STOPPING
(2162, X'872') Concluyendo el gestor de colas.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay suficientes recursos del sistema disponibles.

MQRC_SELECTOR_COUNT_ERROR
(2065, X'811') Recuento de selectores no válido.

MQRC_SELECTOR_ERROR
(2067, X'813') El selector de atributos no es válido.

MQRC_SELECTOR_LIMIT_EXCEEDED
(2066, X'812') Recuento de selectores demasiado grande.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Llamada suprimida por el programa de salida.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos; consulte Códigos de razón

Notas de uso

1. Los valores devueltos son una instantánea de los atributos seleccionados. No hay ninguna garantía de que los atributos permanezcan iguales antes de que la aplicación pueda actuar sobre los valores devueltos.
2. Al abrir una cola modelo, se crea una cola local dinámica. Se crea una cola local dinámica incluso si abre la cola modelo para consultar sus atributos.

   Los atributos de la cola dinámica son en gran medida los mismos que los atributos de la cola modelo en el momento en que se crea la cola dinámica. Si luego utiliza la llamada MQINQ en esta cola, el gestor de colas devuelve los atributos de la cola dinámica y no los atributos de la cola modelo. Consulte Tabla 573 en la página 818 para obtener detalles sobre qué atributos de la cola modelo hereda la cola dinámica.
3. Si el objeto que se está consultando es una cola alias, los valores de atributo devueltos por la llamada MQINQ son los atributos de la cola alias. No son los atributos de la cola base o tema en el que se resuelve el alias.
4. Si el objeto que se está consultando es una cola de clúster, los atributos que se pueden consultar dependen de cómo se abra la cola:

- Puede abrir una cola de clúster para realizar consultas más una o más de las operaciones de entrada, examinar o establecer. Para ello, debe haber una instancia local de la cola de clúster para que la apertura sea satisfactoria. En este caso, los atributos que se pueden consultar son los atributos que son válidos para las colas locales.

  Si la cola de clúster está abierta para consultas sin entrada, examen o conjunto especificado, la llamada devuelve el código de terminación MQCC_WARNING y el código de razón MQRC_SELECTOR_NOT_FOR_TYPE (2068) si intenta consultar atributos que son válidos sólo para colas locales y no para colas de clúster.
- Puede abrir una cola de clúster para realizar consultas al pasar el nombre del gestor de colas base del gestor de colas conectado.

  Para ello, debe haber una instancia local de la cola de clúster para que la apertura sea satisfactoria. Si no se pasa el gestor de colas base, la llamada devuelve el código de terminación MQCC_WARNING y el código de razón MQRC_SELECTOR_NOT_FOR_TYPE (2068) si intenta consultar atributos que son válidos sólo para colas locales y no para colas de clúster
- Si la cola de clúster se abre sólo para consultas, o para consultas y salidas, sólo se pueden consultar los atributos listados. El atributo **QType** tiene el valor MQQT_CLUSTER en este caso:
  - MQCA_Q_DESC
  - MQCA_Q_NAME
  - MQIA_DEF_BIND
  - MQIA_DEF_PERSISTENCE
  - MQIA_DEF_PRIORITY
  - MQIA_INHIBIT_PUT
  - MQIA_Q_TYPE

  Puede abrir la cola de clúster sin ningún enlace fijo. Puede abrirlo con MQOO_BIND_NOT_FIXED especificado en la llamada MQOPEN . De forma alternativa, especifique MQOO_BIND_AS_Q_DEFy establezca el atributo **DefBind** de la cola en MQBND_BIND_NOT_FIXED. Si abre una cola de clúster sin ningún enlace fijo, las llamadas MQINQ sucesivas para la cola pueden consultar distintas instancias de la cola de clúster. Sin embargo, es típico que todas las instancias tengan los mismos valores de atributo.
- Se puede definir un objeto de cola alias para un clúster. Puesto que TARGTYPE y TARGET no son atributos de clúster, el proceso que realiza un proceso MQOPEN en la cola alias no reconoce el objeto en el que se resuelve el alias.

  Durante el MQOPENinicial, la cola alias se resuelve en un gestor de colas y una cola del clúster. La resolución de nombres tiene lugar de nuevo en el gestor de colas remoto y es aquí donde se resuelve el TARGTPYE de la cola alias.

  Si la cola alias se resuelve en un alias de tema, la publicación de mensajes colocados en la cola alias tiene lugar en este gestor de colas remoto.

  Consulte Colas de clúster

5. Es posible que desee consultar una serie de atributos y, a continuación, establecer algunos de ellos utilizando la llamada MQSET . Para programar consultas y establecer de forma eficiente, coloque los atributos que se van a establecer al principio de las matrices de selector. Si lo hace, se pueden utilizar las mismas matrices con recuentos reducidos para MQSET.
6. Si se produce más de una de las situaciones de aviso (consulte el parámetro *CompCode*), el código de razón devuelto es el primero de la lista siguiente que se aplica:
   a. MQRC_SELECTOR_NOT_FOR_TYPE
   b. MQRC_INT_ATTR_COUNT_TOO_SMALL
   c. MQRC_CHAR_ATTRS_TOO_SHORT
7. El tema siguiente tiene información sobre los atributos de objeto:

- “Atributos para colas” en la página 816
- “Atributos de las listas de nombres” en la página 849
- “Atributos de las definiciones de proceso” en la página 851
- “Atributos para el gestor de colas” en la página 780

Invocación en C

```
MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
     CALL 'MQINQ' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,
                        INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,
                        CHARATTRS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
           IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
        CALL MQINQ,(HCONN,HOBJ,SELECTORCOUNT,SELECTORS,INTATTRCOUNT,  X
              INTATTRS,CHARATTRLENGTH,CHARATTRS,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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
```

Invocación en Visual Basic

```
MQINQ Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
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-Consultar propiedad de mensaje

La llamada MQINQMP devuelve el valor de una propiedad de un mensaje.

Sintaxis

MQINQMP (*Hconn*, *Hmsg*, *InqPropOpts*, *Name*, *PropDesc*, *Type*, *ValueLength*, *Value*, *DataLength*, *CompCode*, *Reason*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro *Hmsg* .

Si el manejador de mensajes se ha creado utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en la hebra consultando una propiedad del manejador de mensajes, de lo contrario, la llamada falla con MQRC_CONNECTION_BROKEN.

Msj
Tipo: MQHMSG-entrada

Este es el manejador de mensajes que se debe consultar. El valor ha sido devuelto por una llamada **MQCRTMH** anterior.

InqPropOpts
Tipo: MQIMPO-entrada/salida

Consulte el tipo de datos MQIMPO para obtener más detalles.

Nombre
Tipo: MQCHARV-entrada/salida

El nombre de la propiedad que se va a consultar.

Si no se puede encontrar ninguna propiedad con este nombre, la llamada falla con la razón MQRC_PROPERTY_NOT_AVAILABLE.

Puede utilizar el signo de porcentaje de carácter comodín (%) al final del nombre de propiedad. El comodín coincide con cero o más caracteres, incluido el carácter de punto (.). Esto permite a una aplicación consultar el valor de muchas propiedades. Llame a MQINQMP con la opción MQIMPO_INQ_FIRST para obtener la primera propiedad coincidente y de nuevo con la opción MQIMPO_INQ_NEXT para obtener la siguiente propiedad coincidente. Cuando no hay más propiedades coincidentes disponibles, la llamada falla con MQRC_PROPERTY_NOT_AVAILABLE. Si el campo *ReturnedName* de la estructura InqPropOpts se inicializa con una dirección o un desplazamiento para el nombre devuelto de la propiedad, esto se completa al devolver MQINQMP con el nombre de la propiedad que ha coincidido. Si el campo *VSBufSize* del *ReturnedName* en la estructura de Opts InqPropes menor que la longitud del nombre de propiedad devuelto, el código de terminación se establece en MQCC_FAILED con la razón MQRC_PROPERTY_NAME_TOO_BIG.

Las propiedades que tienen sinónimos conocidos se devuelven de la siguiente manera:

1. Propiedades con el prefijo "mqps." se devuelven como el nombre de propiedad WebSphere MQ . Por ejemplo, "MQTopicString" es el nombre devuelto en lugar de "mqps.Top"
2. Propiedades con el prefijo "jms." o "mcd." se devuelven como el nombre de campo de cabecera JMS, por ejemplo, "JMSExpiration" es el nombre devuelto en lugar de "jms.Exp".
3. Propiedades con el prefijo "usr." se devuelven sin ese prefijo, por ejemplo, se devuelve "Color" en lugar de "usr.Color".

Las propiedades con sinónimos sólo se devuelven una vez.

En el lenguaje de programación C, las variables de macro siguientes se definen para consultar todas las propiedades y, a continuación, todas las propiedades que empiezan por "usr.":

MQPROP_INQUIRE_ALL
Consultar todas las propiedades del mensaje.

MQPROP_INQUIRE_ALL se puede utilizar de la siguiente manera:

```
MQCHARV Name = {MQPROP_INQUIRE_ALL};
```

MQPROP_INQUIRE_ALL_USR
Consultar todas las propiedades del mensaje que inician "usr.". El nombre devuelto se devuelve sin el "usr." .

Si se especifica MQIMP_INQ_NEXT pero el nombre ha cambiado desde la llamada anterior o ésta es la primera llamada, entonces MQIMPO_INQ_FIRST está implícito.

Consulte Nombres de propiedad y Restricciones de nombre de propiedad para obtener más información sobre el uso de nombres de propiedad.

PropDesc
Tipo: MQPD-salida

Esta estructura se utiliza para definir los atributos de una propiedad, incluido lo que sucede si la propiedad no está soportada, a qué contexto de mensaje pertenece la propiedad y en qué mensajes debe copiarse la propiedad. Consulte MQPD para obtener detalles de esta estructura.

type
Tipo: MQLONG-entrada/salida

Al volver de la llamada MQINQMP, este parámetro se establece en el tipo de datos *Valor*. El tipo de datos puede ser cualquiera de los siguientes:

MQTYPE_BOOLEAN
Un booleano.

MQTYPE_BYTE_STRING
una serie de bytes.

MQTYPE_INT8
Un entero con signo de 8 bits.

MQTYPE_INT16
Un entero con signo de 16 bits.

MQTYPE_INT32
Un entero con signo de 32 dígitos.

MQTYPE_INT64
Un entero con signo de 64 bits.

MQTYPE_FLOAT32
Un número de coma flotante de 32 bits.

MQTYPE_FLOAT64
Un número de coma flotante de 64 bits.

MQTYPE_STRING
Una serie de caracteres.

MQTYPE_NULL
La propiedad existe pero tiene un valor nulo.

Si el tipo de datos del valor de propiedad no se reconoce, se devuelve MQTYPE_STRING y se coloca una representación de serie del valor en el área *Valor* . Se puede encontrar una representación de serie del tipo de datos en el campo *TypeString* del parámetro *InqPropOpts* . Se devuelve un código de finalización de aviso con la razón MQRC_PROP_TYPE_NOT_SUPPORTED.

Además, si se especifica la opción MQIMPO_CONVERT_TYPE, se solicita la conversión del valor de propiedad. Utilice *Tipo* como entrada para especificar el tipo de datos con el que desea que se devuelva la propiedad. Consulte la descripción de la opción MQIMPO_CONVERT_TYPE de la estructura MQIMPO para obtener detalles de la conversión de tipos de datos.

Si no solicita la conversión de tipo, puede utilizar el valor siguiente en la entrada:

MQTYPE_AS_SET
El valor de la propiedad se devuelve sin convertir su tipo de datos.

ValueLength
Tipo: MQLONG - entrada

Longitud en bytes del área Valor. Especifique cero para las propiedades para las que no necesita el valor devuelto. Estas podrían ser propiedades diseñadas por una aplicación para tener un valor nulo o

una serie vacía. Especifique también cero si se ha especificado la opción MQIMPO_QUERY_LENGTH ; en este caso, no se devuelve ningún valor.

Valor
Tipo: MQBYTEx*ValueLength* -salida

Este es el área que contiene el valor de propiedad consultado. El almacenamiento intermedio debe estar alineado en un límite adecuado para el valor que se devuelve. Si no lo hace, puede producirse un error cuando se accede al valor más tarde.

Si *ValueLength* es menor que la longitud del valor de propiedad, la mayor parte posible del valor de propiedad se mueve a *Valor* y la llamada falla con el código de terminación MQCC_FAILED y la razón MQRC_PROPERTY_VALUE_TOO_BIG.

El juego de caracteres de los datos en *Valor* se proporciona mediante el campo ReturnedCCSID en el parámetro InqPropOpts. La codificación de los datos en *Valor* se proporciona mediante el campo ReturnedEncoding en el parámetro InqPropOpts.

En el lenguaje de programación C, el parámetro se declara como puntero a void; la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si el parámetro *ValueLength* es cero, no se hace referencia a *Valor* y su valor pasado por los programas escritos en C o System/390 assembler puede ser nulo.

DataLength
Tipo: MQLONG - salida

Es la longitud en bytes del valor de propiedad real tal como se devuelve en el área *Valor* .

Si *DataLength* es menor que la longitud del valor de propiedad, *DataLength* se sigue rellenando a la devolución de la llamada MQINQMP. Esto permite a la aplicación determinar el tamaño del almacenamiento intermedio necesario para acomodar el valor de propiedad y, a continuación, volver a emitir la llamada con un almacenamiento intermedio del tamaño adecuado.

También se pueden devolver los valores siguientes.

Si el parámetro *Tipo* se establece en MQTYPE_STRING o MQTYPE_BYTE_STRING:

MQVL_EMPTY_STRING
La propiedad existe pero no contiene caracteres ni bytes.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_WARNING:

MQRC_PROP_NAME_NOT_CONVERT
(2492, X'09BC') No se ha convertido el nombre de propiedad devuelto.

MQRC_PROP_VALUE_NOT_CONVERTED
(2466, X'09A2') Valor de propiedad no convertido.

MQRC_PROP_TYPE_NOT_SUPPORTED
(2467, X'09A3') El tipo de datos de propiedad no está soportado.

MQRC_RFH_FORMAT_ERROR
(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'0852 ') No se puede cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH
(2157, X'086D') Los ASID primario y de inicio difieren.

MQRC_BUFFER_ERROR
(2004, X'07D4') El parámetro de valor no es válido.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') El parámetro de longitud de valor no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_DATA_LENGTH_ERROR
(2010, X'07DA') El parámetro de longitud de datos no es válido.

MQRC_IMPO_ERROR
(2464, X'09A0') Consultar estructura de opciones de propiedad de mensaje no válida.

MQRC_HMSG_ERROR
(2460, X'099C') Descriptor de mensaje no válido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

MQRC_OPTIONS_ERROR
(2046, X'07F8') Opciones no válidas o no coherentes.

MQRC_PD_ERROR
(2482, X'09B2') La estructura del descriptor de propiedades no es válida.

MQRC_PROP_CONV_NOT_SUPPORTED
(2470, X'09A6') La conversión del tipo de datos real al solicitado no está soportada.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Nombre de propiedad no válido.

MQRC_PROPERTY_NAME_TOO_BIG
(2465, X'09A1') Nombre de propiedad demasiado grande para el almacenamiento intermedio de nombre devuelto.

MQRC_PROPERTY_NOT_AVAILABLE
(2471, X'09A7) Propiedad no disponible.

MQRC_PROPERTY_VALUE_TOO_BIG
(2469, X'09A5') Valor de propiedad demasiado grande para el área Valor.

MQRC_PROP_NUMBER_FORMAT_ERROR
(2472, X'09A8') Se ha encontrado un error de formato de número en los datos de valor.

ERROR_TIPO_PROPIEDAD_MQRC
(2473, X'09A9') Tipo de propiedad solicitado no válido.

MQRC_SOURCE_CCSID_ERROR
(2111, X'083F') Identificador de juego de caracteres codificado de nombre de propiedad no válido.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'0871 ') Almacenamiento insuficiente disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'0893 ') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Invocación en C

```
MQINQMP (Hconn, Hmsg, &InqPropOpts, &Name, &PropDesc, &Type,
ValueLength, Value, &DataLength, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
CALL 'MQINQMP' USING HCONN, HMSG, INQMSGOPTS, NAME, PROPDESC, TYPE,
VALUELENGTH, VALUE, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQINQMP (Hconn, Hmsg, InqPropOpts, Name, PropDesc, Type,
ValueLength, Value, DataLength, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
CALL MQINQMP,(HCONN,HMSG,INQMSGOPTS,NAME,PROPDESC,TYPE,
VALUELENGTH,VALUE,DATALENGTH,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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-Convertir descriptor de mensaje en almacenamiento intermedio

La llamada MQMHBUF convierte un manejador de mensajes en un almacenamiento intermedio y es el inverso de la llamada MQBUFMH.

Sintaxis

MQMHBUF (*Hconn*, *Hmsg*, *MsgHBufOpts*, *Nombre*, *MsgDesc*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro *Hmsg* .

Si el descriptor de mensaje se ha creado utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en la hebra suprimiendo el descriptor de mensaje. Si no se establece una conexión válida, la llamada falla con MQRC_CONNECTION_BROKEN.

Msj
Tipo: MQHMSG-entrada

Este es el descriptor de mensaje para el que se necesita un almacenamiento intermedio. El valor ha sido devuelto por una llamada MQCRTMH anterior.

MsgHBufOpts
Tipo: MQMHBO-entrada

La estructura MQMHBO permite a las aplicaciones especificar opciones que controlan cómo se producen los almacenamientos intermedios a partir de los manejadores de mensajes.

Consulte “MQMHBO-Descriptor de mensaje para opciones de almacenamiento intermedio” en la página 451 para obtener los detalles.

Nombre
Tipo: MQCHARV-entrada

El nombre de la propiedad o propiedades que se van a poner en el almacenamiento intermedio.

Si no se encuentra ninguna propiedad que coincida con el nombre, la llamada falla con MQRC_PROPERTY_NOT_AVAILABLE.

Puede utilizar un comodín para colocar más de una propiedad en el almacenamiento intermedio. Para ello, utilice el carácter comodín '%' al final del nombre de propiedad. Este comodín coincide con cero o más caracteres, incluyendo '.' .

En el lenguaje de programación C, se definen las siguientes variables de macro para consultar todas las propiedades y todas las propiedades que empiezan por 'usr':

MQPROP_INQUIRE_ALL
Colocar todas las propiedades del mensaje en el almacenamiento intermedio

MQPROP_INQUIRE_ALL_USR
Ponga todas las propiedades del mensaje que empiezan con los caracteres 'usr.' en el almacenamiento intermedio.

Consulte Nombres de propiedad y Restricciones de nombre de propiedad para obtener más información sobre el uso de nombres de propiedad.

MsgDesc
Tipo: MQMD - entrada/salida

La estructura *MsgDesc* describe el contenido del área de almacenamiento intermedio.

En la salida, los campos *Encoding*, *CodedCharSetId* y *Format* se establecen para describir correctamente la codificación, el identificador de juego de caracteres y el formato de los datos en el área de almacenamiento intermedio tal como los graba la llamada.

Los datos de esta estructura están en el juego de caracteres y la codificación de la aplicación.

BufferLength
Tipo: MQLONG - entrada

BufferLength es la longitud del área de almacenamiento intermedio, en bytes.

Buffer
Tipo: MQBYTExBufferLength - salida

Buffer define el área que contendrá las propiedades del mensaje. Debe alinear el almacenamiento intermedio en un límite de 4 bytes.

Si *BufferLength* es menor que la longitud necesaria para almacenar las propiedades en *Buffer*, MQMHBUF falla con MQRC_PROPERTY_VALUE_TOO_BIG.

El contenido del almacenamiento intermedio puede cambiar incluso si la llamada falla.

DataLength
Tipo: MQLONG - salida

DataLength es la longitud, en bytes, de las propiedades devueltas en el almacenamiento intermedio. Si el valor es cero, ninguna propiedad coincide con el valor proporcionado en *Name* y la llamada falla con el código de razón MQRC_PROPERTY_NOT_AVAILABLE.

Si *BufferLength* es menor que la longitud necesaria para almacenar las propiedades en el almacenamiento intermedio, la llamada MQMHBUF falla con MQRC_PROPERTY_VALUE_TOO_BIG, pero se sigue entrando un valor en *DataLength*. Esto permite a la aplicación determinar el tamaño del almacenamiento intermedio necesario para acomodar las propiedades y, a continuación, volver a emitir la llamada con el *BufferLength*necesario.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

El código de razón que califica *CompCode*.

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_MHBO_ERROR
(2501, X'095C') El descriptor de contexto de mensaje para la estructura de opciones de almacenamiento intermedio no es válido.

MQRC_BUFFER_ERROR
(2004, X'07D4') El parámetro de almacenamiento intermedio no es válido.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'07D9') Se ha perdido la conexión con el gestor de colas.

MQRC_DATA_LENGTH_ERROR
(2010, X'07DA') El parámetro de longitud de datos no es válido.

MQRC_HMSG_ERROR
(2460, X'099C') Descriptor de mensaje no válido.

MQRC_MD_ERROR
(2026, X'07EA') Descriptor de mensaje no válido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Opciones no válidas o no coherentes.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') El nombre de propiedad no es válido.

MQRC_PROPERTY_NOT_AVAILABLE
(2471, X'09A7') Propiedad no disponible.

MQRC_PROPERTY_VALUE_TOO_BIG
(2469, X'09A5') El valor BufferLength es demasiado pequeño para contener las propiedades especificadas.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Invocación en C

```
MQMHBUF (Hconn, Hmsg, &MsgHBufOpts, &Name, &MsgDesc, BufferLength, Buffer,
         &DataLength, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Notas de uso

MQMHBUF convierte un descriptor de mensaje en un almacenamiento intermedio.

Puede utilizarlo con una salida de API MQGET para acceder a determinadas propiedades, utilizando las API de propiedades de mensajes y, a continuación, volver a pasarlas en un almacenamiento intermedio a una aplicación diseñada para utilizar cabeceras MQRFH2 en lugar de manejadores de mensajes.

Esta llamada es la inversa de la llamada MQBUFMH, que puede utilizar para analizar propiedades de mensaje de un almacenamiento intermedio en un descriptor de mensaje.

Invocación en COBOL

```
     CALL 'MQMHBUF' USING HCONN, HMSG, MSGHBUFOPTS, NAME, MSGDESC,
                          BUFFERLENGTH, BUFFER, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQMHBUF (Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer,
              DataLength, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
CALL MQMHBUF,(HCONN,HMSG,MSGHBUFOPTS,NAME,MSGDESC,BUFFERLENGTH,
                BUFFER,DATALENGTH,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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-Abrir objeto

La llamada MQOPEN establece el acceso a un objeto.

Son válidos los siguientes tipos de objeto:

- Cola (incluidas las listas de distribución)
- Lista de nombres
- Definición de proceso
- Gestor de colas
- Tema

Sintaxis

MQOPEN (*Hconn*, *ObjDesc*, *Options*, *Hobj*, *CompCode*, *Reason*)

Parámetros

Hconn

Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* lo ha devuelto una llamada MQCONN o MQCONNX anterior.

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn*:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

ObjDesc
Tipo: MQOD-entrada/salida

Esta es una estructura que identifica el objeto que se va a abrir; consulte "MQOD-Descriptor de objeto" en la página 454 para obtener más detalles.

Si el campo *ObjectName* del parámetro *ObjDesc* es el nombre de una cola modelo, una cola local dinámicase crea con los atributos de la cola modelo; esto sucede independientemente de las opciones que especifique en el parámetro *Options* . Las operaciones posteriores que utilizan *Hobj* devueltas por la llamada MQOPEN se realizan en la nueva cola dinámica y no en la cola modelo. Esto es cierto incluso para las llamadas MQINQ y MQSET. El nombre de la cola modelo en el parámetro *ObjDesc* se sustituye por el nombre de la cola dinámica creada. El tipo de la cola dinámica viene determinado por el valor del atributo *DefinitionType* de la cola modelo (consulte "Atributos para colas" en la página 816). Para obtener información sobre las opciones de cierre aplicables a las colas dinámicas, consulte la descripción de la llamada MQCLOSE.

opciones
Tipo: MQLONG - entrada

Debe especificar al menos una de las opciones siguientes:

- MQOO_BROWSE
- MQOO_INPUT_ * (sólo uno de estos)
- MQOO_INQUIRE
- MQOO_OUTPUT
- MQOO_SET
- MQOO_BIND_ * (sólo uno de estos)

Consulte la tabla siguiente para obtener detalles de estas opciones; se pueden especificar otras opciones según sea necesario. Si se necesita más de una opción, los valores pueden ser:

- Sumado (no añadir la misma constante más de una vez), o
- Combinado utilizando la operación OR bit a bit (si el lenguaje de programación da soporte a operaciones de bits).

Las combinaciones que no son válidas se anotan; todas las demás combinaciones son válidas. Sólo se permiten las opciones que son aplicables al tipo de objeto especificado por *ObjDesc* . La tabla siguiente muestra opciones MQOPEN válidas para consultas y temas.

Opción	Alias [1]	Local y modelo	Remotos	Clúster no local	Lista de distribución	Tema
MQOO_INPUT_AS_Q_DEF	Sí	Sí	No	No	No	No
MQOO_INPUT_SHARED	Sí	Sí	No	No	No	No
MQOO_INPUT_EXCLUSIVE	Sí	Sí	No	No	No	No
MQOO_OUTPUT	Sí	Sí	Sí	Sí	Sí	Sí
MQOO_BROWSE	Sí	Sí	No	No	No	No
MQOO_CO_OP	Sí	Sí	No	No	No	No
MQOO_INQUIRE	Sí	Sí	[2]	Sí	No	No
MQOO_SET	Sí	Sí	[2]	No	No	No
MQOO_BIND_ON_OPEN [3]	Sí	Sí	Sí	Sí	Sí	No

Opción	Alias [1]	Local y modelo	Remotos	Clúster no local	Lista de distribución	Tema
MQOO_BIND_NOT_FIXED [3]	Sí	Sí	Sí	Sí	Sí	No
MQOO_BIND_ON_GROUP [3]	Sí	Sí	Sí	Sí	Sí	No
MQOO_BIND_AS_Q_DEF [3]	Sí	Sí	Sí	Sí	Sí	No
MQOO_SAVE_ALL_CONTEXT	Sí	Sí	No	No	No	No
MQOO_PASS_IDENTITY_CONTEXT	Sí	Sí	Sí	Sí	Sí	[4]
MQOO_PASS_ALL_CONTEXT	Sí	Sí	Sí	Sí	Sí	Sí
MQOO_SET_IDENTITY_CONTEXT	Sí	Sí	Sí	Sí	Sí	[4]
MQOO_SET_ALL_CONTEXT	Sí	Sí	Sí	Sí	Sí	Sí
MQOO_NO_READ_AHEAD	Sí	Sí	No	No	No	No
MQOO_READ_AHEAD	Sí	Sí	No	No	No	No
MQOO_READ_AHEAD_AS_Q_DEF	Sí	Sí	No	No	No	No
MQOO_ALTERNATE_USER_AUTHORITY	Sí	Sí	Sí	Sí	Sí	Sí
MQOO_FAIL_IF_QUIESCING	Sí	Sí	Sí	Sí	Sí	Sí
MQOO_RESOLVE_LOCAL_Q	Sí	Sí	Sí	Sí	No	No
MQOO_RESOLVE_LOCAL_TOPIC	No	No	No	No	No	Sí
MQOO_NO_MULTICAST	No	No	No	No	No	Sí

Nota:

1. La validez de las opciones para los alias depende de la validez de la opción para la cola en la que se resuelve el alias.
2. Esta opción sólo es válida para la definición local de una cola remota.
3. Esta opción se puede especificar para cualquier tipo de cola, pero se ignora si la cola no es una cola de clúster. Sin embargo, el atributo de cola *DefBind* altera temporalmente la cola base incluso cuando la cola alias no está en un clúster.
4. Estos atributos se pueden utilizar con un tema, pero sólo afectan al contexto establecido para el mensaje retenido, no a los campos de contexto enviados a cualquier suscriptor.

Opciones de acceso: las opciones siguientes controlan el tipo de operaciones que se pueden realizar en el objeto:

MQOO_INPUT_AS_Q_DEF
Abra la cola para obtener mensajes utilizando el valor predeterminado definido por la cola.

La cola se abre para su uso con las llamadas MQGET posteriores. El tipo de acceso es compartido o exclusivo, en función del valor del atributo de cola *DefInputOpenOption* ; consulte “Atributos para colas” en la página 816 para obtener más detalles.

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas.

MQOO_INPUT_SHARED
Abra la cola para obtener mensajes con acceso compartido.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada puede realizarse correctamente si la cola está abierta actualmente por esta u otra aplicación con MQOO_INPUT_SHARED, pero falla con el código de razón MQRC_OBJECT_IN_USE si la cola está abierta actualmente con MQOO_INPUT_EXCLUSIVE.

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas.

MQOO_INPUT_EXCLUSIVE
Abra la cola para obtener mensajes con acceso exclusivo.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada falla con el código de razón MQRC_OBJECT_IN_USE si la cola está abierta actualmente por esta u otra aplicación para cualquier tipo de entrada (MQOO_INPUT_SHARED o MQOO_INPUT_EXCLUSIVE).

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas.

MQOO_OUTPUT

Abra la cola para transferir mensajes, o un tema o serie de tema para publicar mensajes.

La cola o tema se abre para su uso con llamadas MQPUT posteriores.

Una llamada MQOPEN con esta opción puede ser satisfactoria incluso si el atributo de cola *InhibitPut* se establece en MQQA_PUT_INITED (aunque las llamadas MQPUT posteriores fallan mientras el atributo se establece en este valor).

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución y los temas.

Las siguientes notas se aplican a estas opciones:

- Sólo se puede especificar una de estas opciones.
- Una llamada MQOPEN con una de estas opciones puede ser satisfactoria incluso si el atributo de cola *InhibitGet* se establece en MQQA_GET_INSITED (aunque las llamadas MQGET posteriores fallan mientras el atributo se establece en este valor).
- Si la cola se define como no compartible (es decir, el atributo de cola *Shareability* tiene el valor MQQA_NOT_SHAREABLE), los intentos de abrir la cola para acceso compartido se tratan como intentos de abrir la cola con acceso exclusivo.
- Si se abre una cola alias con una de estas opciones, la prueba de uso exclusivo (o si otra aplicación tiene uso exclusivo) se realiza en la cola base en la que se resuelve el alias.
- Estas opciones no son válidas si *ObjectQMgrName* es el nombre de un alias de gestor de colas; esto es cierto incluso si el valor del atributo *RemoteQMgrName* en la definición local de una cola remota utilizada para el alias de gestor de colas es el nombre del gestor de colas local.

MQOO_BROWSE

Abra la cola para examinar los mensajes.

La cola se abre para su uso con llamadas MQGET posteriores con una de las opciones siguientes:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

Esto está permitido incluso si la cola está abierta actualmente para MQOO_INPUT_EXCLUSIVE. Una llamada MQOPEN con la opción MQOO_BROWSE establece un cursor para examinar y lo sitúa lógicamente antes del primer mensaje en la cola; consulte Campo MQGMO-Options para obtener más información.

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas. Tampoco es válido si *ObjectQMgrName* es el nombre de un alias de gestor de colas; esto es cierto incluso si el valor del atributo *RemoteQMgrName* en la definición local de una cola remota utilizada para el alias de gestor de colas es el nombre del gestor de colas local.

MQOO_CO_OP

Abierto como miembro cooperante del conjunto de descriptores de contexto.

Esta opción sólo es válida con la opción MQOO_BROWSE. Si se especifica sin MQOO_BROWSE, MQOPEN devuelve con MQRC_OPTIONS_ERROR.

El descriptor de contexto devuelto se considera miembro de un conjunto de descriptores de contexto cooperantes para llamadas MQGET posteriores con una de las opciones siguientes:

- MQGMO_MARK_BROWSE_CO_OP

- MQGMO_UNMARKED_BROWSE_MSG
- MQGMO_UNMARK_BROWSE_CO_OP

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas.

MQOO_INQUIRE
Abrir objeto para consultar atributos.

La cola, la lista de nombres, la definición de proceso o el gestor de colas se abre para su uso con llamadas MQINQ posteriores.

Esta opción es válida para todos los tipos de objetos que no sean listas de distribución. No es válido si *ObjectQMgrName* es el nombre de un alias de gestor de colas; esto es cierto incluso si el valor del atributo *RemoteQMgrName* en la definición local de una cola remota utilizada para el alias de gestor de colas es el nombre del gestor de colas local.

MQOO_SET
Abra la cola para establecer atributos.

La cola se abre para utilizarla con las llamadas MQSET posteriores.

Esta opción es válida para todos los tipos de cola que no sean listas de distribución. No es válido si *ObjectQMgrName* es el nombre de una definición local de una cola remota; esto es cierto incluso si el valor del atributo *RemoteQMgrName* en la definición local de una cola remota utilizada para el alias de gestor de colas es el nombre del gestor de colas local.

Opciones de enlace: Las opciones siguientes se aplican cuando el objeto que se está abriendo es una cola de clúster; estas opciones controlan el enlace del descriptor de contexto de cola a una instancia de la cola de clúster:

MQOO_BIND_ON_OPEN
El gestor de colas local enlaza el descriptor de contexto de cola con una instancia de la cola de destino cuando se abre la cola. Como resultado, todos los mensajes colocados utilizando este descriptor de contexto se envían a la misma instancia de la cola de destino y por la misma ruta.

Esta opción solo es válida para colas y solo afecta a colas de clúster. Si se especifica en una cola que no sea de clúster, la opción se pasará por alto.

MQOO_BIND_NOT_FIXED
Esto detiene el gestor de colas local que enlaza el descriptor de contexto de cola con una instancia de la cola de destino. Como resultado, las llamadas MQPUT sucesivas que utilizan este manejador envían los mensajes a *distintas* instancias de la cola de destino, o a la misma instancia pero por rutas diferentes. También permite que el gestor de colas local, un gestor de colas remoto o un agente de canal de mensajes (MCA) cambien la instancia seleccionada más tarde, según las condiciones de red.

Nota: Las aplicaciones cliente y servidor que necesitan intercambiar una *serie* de mensajes para completar una transacción no deben utilizar MQOO_BIND_NOT_FIXED (o MQOO_BIND_AS_Q_DEF cuando *DefBind* tiene el valor MQBND_BIND_NOT_FIXED), porque los mensajes sucesivos de la serie se pueden enviar a distintas instancias de la aplicación de servidor.

Si se especifica MQOO_BROWSE o una de las opciones MQOO_INPUT_ * para una cola de clúster, se fuerza al gestor de colas a seleccionar la instancia local de la cola de clúster. Como resultado, el enlace del descriptor de contexto de cola se arregla, incluso si se especifica MQOO_BIND_NOT_FIXED.

Si se especifica MQOO_INQUIRE con MQOO_BIND_NOT_FIXED, las llamadas MQINQ sucesivas que utilizan ese manejador pueden consultar distintas instancias de la cola de clúster, aunque normalmente todas las instancias tienen los mismos valores de atributo.

MQOO_BIND_NOT_FIXED sólo es válido para las colas y sólo afecta a las colas de clúster. Si se especifica en una cola que no sea de clúster, la opción se pasará por alto.

MQOO_BIND_ON_GROUP
Permite a una aplicación solicitar que un grupo de mensajes se asigne a la misma instancia de destino.

Esta opción solo es válida para colas y solo afecta a colas de clúster. Si se especifica en una cola que no sea de clúster, la opción se pasará por alto.

MQOO_BIND_AS_Q_DEF
El gestor de colas local enlaza el descriptor de contexto de cola de la forma definida por el atributo de cola *DefBind* . El valor de este atributo es MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXED o MQBND_BIND_ON_GROUP.

MQOO_BIND_AS_Q_DEF es el valor predeterminado cuando no se especifica MQOO_BIND_ON_OPEN, MQOO_BIND_NOT_FIXED o MQOO_BIND_ON_GROUP.

MQOO_BIND_AS_Q_DEF ayuda a la documentación del programa. No está previsto que esta opción se utilice con cualquiera de las otras dos opciones de vinculación, pero debido a que su valor es cero, no se puede detectar dicho uso.

Opciones de contexto: Las opciones siguientes controlan el proceso del contexto de mensaje:

MQOO_SAVE_ALL_CONTEXT
La información de contexto está asociada con este descriptor de contexto de cola. Esta información se establece a partir del contexto de cualquier mensaje recuperado utilizando este descriptor de contexto. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Esta información de contexto se puede pasar a un mensaje que luego se coloca en una cola utilizando las llamadas MQPUT o MQPUT1 . Consulte las opciones MQPMO_PASS_IDENTITY_CONTEXT y MQPMO_PASS_ALL_CONTEXT descritas en “MQPMO- Opciones de transferencia de mensajes” en la página 475.

Hasta que un mensaje se haya recuperado correctamente, el contexto no se puede pasar a un mensaje que se está colocando en una cola.

Un mensaje recuperado utilizando una de las opciones de examen MQGMO_BROWSE_ * no tiene guardada su información de contexto (aunque los campos de contexto del parámetro *MsgDesc* se establecen después de un examen).

Esta opción sólo es válida para colas locales, alias y modelo; no es válida para colas remotas, listas de distribución y objetos que no son colas. Debe especificarse una de las opciones MQOO_INPUT_ *.

MQOO_PASS_IDENTITY_CONTEXT
Esto permite especificar la opción MQPMO_PASS_IDENTITY_CONTEXT en el parámetro *PutMsgOpts* cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad de una cola de entrada que se ha abierto con la opción MQOO_SAVE_ALL_CONTEXT. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto.

Debe especificarse la opción MQOO_OUTPUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

MQOO_PASS_ALL_CONTEXT
Esto permite especificar la opción MQPMO_PASS_ALL_CONTEXT en el parámetro *PutMsgOpts* cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad y origen de una cola de entrada que se ha abierto con la opción MQOO_SAVE_ALL_CONTEXT. Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto .

Esta opción implica MQOO_PASS_IDENTITY_CONTEXT, por lo que no es necesario especificarlo. Debe especificarse la opción MQOO_OUTPUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

MQOO_SET_IDENTITY_CONTEXT
Esto permite especificar la opción MQPMO_SET_IDENTITY_CONTEXT en el parámetro *PutMsgOpts* cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad contenida en el parámetro *MsgDesc* especificado en la llamada MQPUT o MQPUT1 . Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto .

Esta opción implica MQOO_PASS_IDENTITY_CONTEXT, por lo que no es necesario especificarlo. Debe especificarse la opción MQOO_OUTPUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

MQOO_SET_ALL_CONTEXT
Esto permite especificar la opción MQPMO_SET_ALL_CONTEXT en el parámetro *PutMsgOpts* cuando se coloca un mensaje en una cola; esto proporciona al mensaje la información de contexto de identidad y origen contenida en el parámetro *MsgDesc* especificado en la llamada MQPUT o MQPUT1 . Para obtener más información sobre el contexto de mensaje, consulte Contexto de mensaje y Control de la información de contexto .

Esta opción implica las siguientes opciones, por lo que no es necesario especificarlas:

- MQOO_PASS_IDENTITY_CONTEXT
- MQOO_PASS_ALL_CONTEXT
- MQOO_SET_IDENTITY_CONTEXT

Debe especificarse la opción MQOO_OUTPUT.

Esta opción es válida para todos los tipos de cola, incluidas las listas de distribución.

Opciones de lectura anticipada:

Cuando llama a MQOPEN con MQOO_READ_AHEAD, el cliente de WebSphere MQ sólo habilita la lectura anticipada si se cumplen determinadas condiciones. Estas condiciones incluyen:

- El cliente y el gestor de colas remoto deben ser de WebSphere MQ Versión 7 o posterior.
- La aplicación cliente debe compilarse y enlazarse con las bibliotecas de cliente MQI WebSphere MQ con hebras.
- El canal de cliente debe utilizar el protocolo TCP/IP
- El canal debe tener un valor SharingConversations (SHARECNV) distinto de cero tanto en las definiciones de canal de cliente y servidor.

Las opciones siguientes controlan si los mensajes no persistentes se envían al cliente antes de que una aplicación los solicite. Las siguientes notas se aplican a las opciones de lectura anticipada:

- Sólo se puede especificar una de estas opciones.
- Estas opciones sólo son válidas para colas locales, alias y modelo. No son válidos para colas remotas, listas de distribución, temas o gestores de colas.
- Estas opciones sólo son aplicables cuando también se especifica una de las opciones MQOO_BROWSE, MQOO_INPUT_SHARED y MQOO_INPUT_EXCLUSIVE, aunque no es un error especificar estas opciones con MQOO_INQUIRE o MQOO_SET.
- Si la aplicación no se ejecuta como un cliente de IBM WebSphere MQ , estas opciones se ignoran.

MQOO_NO_READ_AHEAD
Los mensajes no persistentes no se envían al cliente antes de que una aplicación los solicite.

MQOO_READ_AHEAD
Los mensajes no persistentes se envían al cliente antes de que una aplicación los solicite.

MQOO_READ_AHEAD_AS_Q_DEF
El comportamiento de lectura anticipada viene determinado por el atributo de lectura anticipada predeterminado de la cola que se está abriendo. Este es el valor predeterminado.

Otras opciones: Las opciones siguientes controlan la comprobación de autorización, lo que sucede cuando el gestor de colas se está desactivando temporalmente, si se debe resolver el nombre de cola local y la multidifusión:

MQOO_ALTERNATE_USER_AUTHORITY
El campo *AlternateUserId* del parámetro *ObjDesc* contiene un identificador de usuario que se debe utilizar para validar esta llamada MQOPEN. La llamada sólo puede realizarse correctamente si este *AlternateUserId* está autorizado para abrir el objeto con las opciones de acceso especificadas, independientemente de si el identificador de usuario bajo el que se ejecuta la aplicación está autorizado para hacerlo. Sin embargo, esto no se aplica a las opciones de contexto especificadas, que siempre se comprueban con el identificador de usuario bajo el que se ejecuta la aplicación.

Esta opción es válida para todos los tipos de objeto.

MQOO_FAIL_IF_QUIESCING
La llamada MQOPEN falla si el gestor de colas está en estado de desactivación temporal.

En z/OS, para una aplicación CICS o IMS , esta opción también fuerza que la llamada MQOPEN falle si la conexión está en estado de desactivación temporal.

Esta opción es válida para todos los tipos de objeto.

Para obtener información sobre los canales de cliente, consulte Visión general de los clientes MQI de IBM WebSphere MQ .

MQOO_RESOLVE_LOCAL_Q
Rellene ResolvedQName en la estructura MQOD con el nombre de la cola local que se ha abierto. De forma similar, el nombre ResolvedQMgrse rellena con el nombre del gestor de colas local que aloja la cola local. Si la estructura MQOD es inferior a la versión 3, MQOO_RESOLVE_LOCAL_Q se ignora sin que se devuelva ningún error.

La cola local siempre se devuelve cuando se abre una cola local, alias o modelo, pero este no es el caso cuando, por ejemplo, se abre una cola remota o una cola de clúster no local sin la opción MQOO_RESOLVE_LOCAL_Q; el nombre ResolvedQName y ResolvedQMgrse rellenan con el nombre RemoteQName y RemoteQMgrque se encuentra en la definición de cola remota, o de forma similar con la cola de clúster remoto elegida.

Si especifica MQOO_RESOLVE_LOCAL_Q al abrir, por ejemplo, una cola remota, ResolvedQName es la cola de transmisión a la que se colocan los mensajes. El nombre ResolvedQMgrse rellena con el nombre del gestor de colas local que aloja la cola de transmisión.

Si tiene autorización para examinar, entrar o salir en una cola, tiene la autorización necesaria para especificar este distintivo en la llamada MQOPEN. No se necesita ninguna autorización especial.

Esta opción sólo es válida para colas y gestores de colas.

MQOO_RESOLVE_LOCAL_TOPIC
Rellene ResolvedQName en la estructura MQOD con el nombre del tema administrativo abierto.

MQOO_NO_MULTIDIFUSIÓN
Los mensajes de publicación no se envían utilizando multidifusión.

Esta opción sólo es válida con la opción MQOO_OUTPUT. Si se especifica sin MQOO_OUTPUT, MQOPEN devuelve con MQRC_OPTIONS_ERROR.

Esta opción sólo es válida para un tema.

Hobj
Tipo: MQHOBJ - salida

Este descriptor de contexto representa el acceso que se ha establecido al objeto. Debe especificarse en las llamadas IBM WebSphere MQ posteriores que operan en el objeto. Deja de ser válido cuando se emite la llamada MQCLOSE o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto.

El ámbito del descriptor de contexto de objeto devuelto es el mismo que el ámbito del descriptor de contexto de conexión especificado en la llamada. Consulte Parámetro MQCONN-Hconn para obtener información sobre el ámbito del descriptor de contexto.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

El código de razón que califica *CompCode*.

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_WARNING:

MQRC_MULTIPLE_RAZONES
(2136, X'858 ') Se han devuelto varios códigos de razón.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ALIAS_BASE_Q_TYPE_ERROR
(2001, X'7D1') La cola base de alias no es un tipo válido.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_NOT_AVAILABLE
(2345, X' 929 ') Recurso de acoplamiento no disponible.

MQRC_CF_STRUC_AUTH_FAILED
(2348, X'92C') La comprobación de autorización de la estructura del recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_ERROR
(2349, X'92D') La estructura del recurso de acoplamiento no es válida.

MQRC_CF_STRUC_FAILED
(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') La cabecera de lista de la estructura de recurso de acoplamiento se está utilizando.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') CICS ha rechazado la solicitud de espera.

MQRC_CLUSTER_EXIT_ERROR
(2266, X'8DA') La salida de carga de trabajo del clúster ha fallado.

MQRC_CLUSTER_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida
(2268, X'8DC') Llamadas de colocación inhibidas para todas las colas del clúster.

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') La resolución de nombres de clúster ha fallado.

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Error de recurso de clúster.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') El subsistema Db2 no está disponible.

MQRC_DEF_XMIT_Q_TYPE_ERROR
(2198, X'896 ') La cola de transmisión predeterminada no es local.

MQRC_DEF_XMIT_Q_USAGE_ERROR
(2199, X'897 ') Error de uso de cola de transmisión predeterminado.

MQRC_DYNAMIC_Q_NAME_ERROR
(2011, X'7DB') Nombre de cola dinámica no válido.

MQRC_HANDLE_NOT_AVAILABLE
(2017, X'7E1') No hay más manejadores disponibles.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR
(2019, X'7E3') El manejador de objeto no es válido.

MQRC_MULTIPLE_RAZONES
(2136, X'858 ') Se han devuelto varios códigos de razón.

MQRC_NAME_IN_USE
(2201, X'899 ') Nombre en uso.

MQRC_NAME_NOT_VALID_FOR_TYPE
(2194, X'892 ') El nombre de objeto no es válido para el tipo de objeto.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_OBJECT_ALREADY_EXISTS
(2100, X'834 ') El objeto existe.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OBJECT_IN_USE
(2042, X'7FA') El objeto ya está abierto con opciones en conflicto.

MQRC_OBJECT_LEVEL_INCOMPATIBLE
(2360, X' 938 ') Nivel de objeto no compatible.

MQRC_OBJECT_NAME_ERROR
(2152, X'868 ') Nombre de objeto no válido.

MQRC_OBJECT_NOT_UNIQUE
(2343, X' 927 ') Objeto no exclusivo.

MQRC_OBJECT_Q_MGR_NAME_ERROR
(2153, X'869 ') Nombre de gestor de colas de objeto no válido.

MQRC_OBJECT_RECORDS_ERROR
(2155, X'86B') Los registros de objeto no son válidos.

MQRC_OBJECT_STRING_ERROR
(2441, X'0989 ') El campo Objectstring no es válido

MQRC_OBJECT_TYPE_ERROR
(2043, X'7FB') Tipo de objeto no válido.

MQRC_OD_ERROR
(2044, X'7FC') La estructura de descriptor de objeto no es válida.

MQRC_OPTION_NOT_VALID_FOR_TYPE
(2045, X'7FD') Opción no válida para el tipo de objeto.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_PAGESET_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_Q_DELETED
(2052, X'804') La cola se ha suprimido.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_Q_TYPE_ERROR
(2057, X'809 ') Tipo de cola no válido.

MQRC_RECS_PRESENT_ERROR
(2154, X'86A') Número de registros presentes no válido.

MQRC_REMOTE_Q_NAME_ERROR
(2184, X'888 ') El nombre de cola remota no es válido.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_RESPONSE_RECORDS_ERROR
(2156, X'86C') Los registros de respuesta no son válidos.

MQRC_SECURITY_ERROR
(2063, X'80F') Se ha producido un error de seguridad.

MQRC_SELECTOR_SYNTAX_ERROR
2459 (X'099B') Se ha emitido una llamada MQOPEN, MQPUT1 o MQSUB, pero se ha especificado una serie de selección que contenía un error de sintaxis.

MQRC_STOPPED_BY_CLUSTER_EXIT
(2188, X'88C') Llamada rechazada por salida de carga de trabajo de clúster.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

MQRC_UNKNOWN_ALIAS_BASE_Q
(2082, X'822 ') Cola base alias desconocida.

MQRC_UNKNOWN_DEF_XMIT_Q
(2197, X'895 ') Cola de transmisión predeterminada desconocida.

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825 ') Nombre de objeto desconocido.

MQRC_UNKNOWN_OBJECT_Q_MGR
(2086, X'826 ') Gestor de colas de objetos desconocido.

MQRC_UNKNOWN_REMOTE_Q_MGR
(2087, X'827 ') Gestor de colas remoto desconocido.

MQRC_UNKNOWN_XMIT_Q
(2196, X'894 ') Cola de transmisión desconocida.

MQRC_ERR_CF_LEVEL
(2366, X'93E') La estructura del recurso de acoplamiento tiene un nivel incorrecto.

MQRC_XMIT_Q_TYPE_ERROR
(2091, X'82B') Cola de transmisión no local.

MQRC_XMIT_Q_USAGE_ERROR
(2092, X'82C') Cola de transmisión con un uso incorrecto.

Para obtener información detallada sobre estos códigos, consulte:

- Códigos de razón para todas las demás plataformas IBM WebSphere MQ excepto z/OS.

Notas de uso general

1. El objeto abierto es uno de los siguientes:
   - Una cola para:
     - Obtener o examinar mensajes (utilizando la llamada MQGET)
     - Transferir mensajes (utilizando la llamada MQPUT)
     - Consultar los atributos de la cola (utilizando la llamada MQINQ)
     - Establecer los atributos de la cola (utilizando la llamada MQSET)

     Si la cola especificada es una cola modelo, se crea una cola local dinámica. Consulte el parámetro *ObjDesc* descrito en “MQOPEN-Abrir objeto” en la página 713.

     Una lista de distribución es un tipo especial de objeto de cola que contiene una lista de colas. Se puede abrir para transferir mensajes, pero no para obtener o examinar mensajes, ni para consultar o establecer atributos. Consulte la nota de uso 8 para obtener más detalles.

     Una cola que tiene `QSGDISP(GROUP)` es un tipo especial de definición de cola que no se puede utilizar con las llamadas MQOPEN o MQPUT1 .
   - Una lista de nombres para consultar los nombres de las colas de la lista (utilizando la llamada MQINQ).
   - Una definición de proceso para consultar sobre los atributos de proceso (utilizando la llamada MQINQ).

- El gestor de colas para consultar los atributos del gestor de colas local (utilizando la llamada MQINQ).
- Un tema para publicar un mensaje (utilizando la llamada MQPUT)

2. Una aplicación puede abrir el mismo objeto más de una vez. Se devuelve un descriptor de objeto diferente para cada apertura. Cada descriptor de contexto que se devuelve se puede utilizar para las funciones para las que se ha realizado la apertura correspondiente.
3. Si el objeto que se está abriendo es una cola distinta de una cola de clúster, toda la resolución de nombres dentro del gestor de colas local tiene lugar en el momento de la llamada MQOPEN. Puede incluir:

   - Resolución del nombre de una definición local de una cola remota al nombre del gestor de colas remoto y el nombre por el que se conoce la cola en el gestor de colas remoto
   - Resolución del nombre del gestor de colas remoto en el nombre de una cola de transmisión local
   - (Sóloz/OS) Resolución del nombre del gestor de colas remoto al nombre de la cola de transmisión compartida utilizada por el agente de IGQ (sólo se aplica si los gestores de colas local y remoto pertenecen al mismo grupo de compartición de colas)
   - Resolución de alias para el nombre de una cola base o un objeto de tema.

   Sin embargo, tenga en cuenta que las llamadas MQINQ o MQSET posteriores para el descriptor de contexto se relacionan únicamente con el nombre que se ha abierto y no con el objeto resultante después de que se haya producido la resolución de nombres. Por ejemplo, si el objeto abierto es un alias, los atributos devueltos por la llamada MQINQ son los atributos del alias, no los atributos de la cola base o un objeto de tema en el que se resuelve el alias.

   Si el objeto que se está abriendo es una cola de clúster, la resolución de nombres se puede producir en el momento de la llamada MQOPEN, o se puede aplazar hasta más adelante. El punto en el que se produce la resolución está controlado por las opciones MQOO_BIND_ * especificadas en la llamada MQOPEN:

   - MQOO_BIND_ON_OPEN
   - MQOO_BIND_NOT_FIXED
   - MQOO_BIND_AS_Q_DEF
   - MQOO_BIND_ON_GROUP

   Consulte Resolución de nombres para obtener más información sobre la resolución de nombres para colas de clúster.
4. Una llamada MQOPEN con la opción MQOO_BROWSE establece un cursor para examinar, para su uso con llamadas MQGET que especifican el descriptor de objeto y una de las opciones de examinar. Esto permite explorar la cola sin alterar su contenido. Un mensaje que se ha encontrado examinando puede eliminarse de la cola utilizando la opción MQGMO_MSG_UNDER_CURSOR.

   Varios cursores de examen pueden estar activos para una sola aplicación emitiendo varias solicitudes MQOPEN para la misma cola.
5. A las aplicaciones iniciadas por un supervisor desencadenante se les pasa el nombre de la cola asociada con la aplicación cuando se inicia la aplicación. Este nombre de cola se puede especificar en el parámetro *ObjDesc* para abrir la cola. Para conocer detalles, consulte “MQTMC2 -Mensaje de desencadenante 2 (formato de caracteres)” en la página 584.
6. En IBM i, las aplicaciones que se ejecutan en modalidad de compatibilidad se conectan automáticamente al gestor de colas mediante la primera llamada MQOPEN emitida por la aplicación (si la aplicación todavía no se ha conectado al gestor de colas utilizando la llamada MQCONN).

   Las aplicaciones que no se ejecutan en modalidad de compatibilidad deben emitir la llamada MQCONN o MQCONNX para conectarse al gestor de colas explícitamente, antes de utilizar la llamada MQOPEN para abrir un objeto.

Opciones de lectura anticipada

Cuando llama a MQOPEN con MQOO_READ_AHEAD, el cliente de WebSphere MQ sólo habilita la lectura anticipada si se cumplen determinadas condiciones. Estas condiciones incluyen:

- El cliente y el gestor de colas remoto deben ser de WebSphere MQ Versión 7 o posterior.
- La aplicación cliente debe compilarse y enlazarse con las bibliotecas de cliente MQI WebSphere MQ con hebras.
- El canal de cliente debe utilizar el protocolo TCP/IP
- El canal debe tener un valor SharingConversations (SHARECNV) distinto de cero tanto en las definiciones de canal de cliente y servidor.

Las notas siguientes se aplican al uso de opciones de lectura anticipada.

1. Las opciones de lectura anticipada sólo son aplicables cuando también se especifica una de las opciones MQOO_BROWSE, MQOO_INPUT_SHARED y MQOO_INPUT_EXCLUSIVE. No se genera un error si se especifican opciones de lectura anticipada con las opciones MQOO_ INQUIRE o MQOO_SET.
2. La lectura anticipada no está habilitada cuando se solicita si las opciones utilizadas en la primera llamada MQGET no están soportadas para su uso con la lectura anticipada. Además, la lectura anticipada está inhabilitada cuando el cliente se está conectando a un gestor de colas que no soporta la lectura anticipada.
3. Si la aplicación no se ejecuta como un cliente IBM WebSphere MQ , las opciones de lectura anticipada se ignoran.

Colas de clúster

Las notas siguientes se aplican al uso de colas de clúster.

1. Cuando una cola de clúster se abre por primera vez, y el gestor de colas local no es un gestor de colas de depósito completo, el gestor de colas local obtiene información sobre la cola de clúster de un gestor de colas de depósito completo. Cuando la red está ocupada, el gestor de colas local puede tardar varios segundos en recibir la información necesaria del gestor de colas de repositorio. Como resultado, es posible que la aplicación que emite la llamada MQOPEN tenga que esperar hasta 10 segundos antes de que se devuelva el control de la llamada MQOPEN. Si el gestor de colas local no recibe la información necesaria sobre la cola de clúster dentro de este tiempo, la llamada falla con el código de razón MQRC_CLUSTER_RESOLUTION_ERROR.
2. Cuando se abre una cola de clúster y hay varias instancias de la cola en el clúster, la instancia abierta depende de las opciones especificadas en la llamada MQOPEN:
   - Si las opciones especificadas incluyen alguna de las siguientes:
     - MQOO_BROWSE
     - MQOO_INPUT_AS_Q_DEF
     - MQOO_INPUT_EXCLUSIVE
     - MQOO_INPUT_SHARED
     - MQOO_SET

     la instancia de la cola de clúster abierta debe ser la instancia local. Si no hay ninguna instancia local de la cola, la llamada MQOPEN falla.
   - Si las opciones especificadas no incluyen ninguna de las opciones descritas anteriormente, pero incluyen una o ambas de las siguientes:
     - MQOO_INQUIRE
     - MQOO_OUTPUT

     la instancia abierta es la instancia local si hay una, y una instancia remota de lo contrario (si se utilizan los valores predeterminados de CLWLUSEQ). Sin embargo, la instancia elegida por el gestor de colas puede ser alterada por una salida de carga de trabajo de clúster (si la hay).

3. Si hay una suscripción para la cola, pero no la reconoce un repositorio completo, el objeto no está presente en el clúster y la llamada falla con el código de razón MQRC_OBJECT_NAME.

Para obtener más información sobre las colas de clúster, consulte Colas de clúster.

Listas de distribución

Las siguientes notas se aplican al uso de listas de distribución.

Las listas de distribución están soportadas en los entornos siguientes: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más clientes MQI de IBM WebSphere MQ conectados a estos sistemas.

1. Los campos de la estructura MQOD deben establecerse de la forma siguiente al abrir una lista de distribución:
   - *Version* debe ser MQOD_VERSION_2 o superior.
   - *ObjectType* debe ser MQOT_Q.
   - *ObjectName* debe estar en blanco o la serie nula.
   - *ObjectQMgrName* debe estar en blanco o la serie nula.
   - *RecsPresent* Tiene que ser mayor que cero.
   - Uno de *ObjectRecOffset* y *ObjectRecPtr* debe ser cero y el otro distinto de cero.
   - No más de uno de *ResponseRecOffset* y *ResponseRecPtr* puede ser distinto de cero.
   - Debe haber *RecsPresent* registros de objeto, direccionado por *ObjectRecOffset* o *ObjectRecPtr*. Los registros de objeto deben establecerse en los nombres de las colas de destino que se van a abrir.
   - Si uno de *ResponseRecOffset* y *ResponseRecPtr* es distinto de cero, debe haber *RecsPresent* registros de respuesta presentes. Los establece el gestor de colas si la llamada se completa con el código de razón MQRC_MULTIPLE_REASON.

   Un MQOD version-2 también se puede utilizar para abrir una única cola que no esté en una lista de distribución, asegurándose de que *RecsPresent* sea cero.
2. Sólo las siguientes opciones de apertura son válidas en el parámetro *Options* :
   - MQOO_OUTPUT
   - MQOO_PASS_ * _CONTEXTO
   - MQOO_SET_ * _CONTEXTO
   - MQOO_ALTERNATE_USER_AUTHORITY
   - MQOO_FAIL_IF_QUIESCING
3. Las colas de destino de la lista de distribución pueden ser colas locales, alias o remotas, pero no pueden ser colas modelo. Si se especifica una cola modelo, dicha cola no se puede abrir, con el código de razón MQRC_Q_TYPE_ERROR. Sin embargo, esto no impide que otras colas de la lista se abran correctamente.
4. Los parámetros de código de terminación y código de razón se establecen de la forma siguiente:
   - Si las operaciones abiertas para las colas de la lista de distribución son satisfactorias o fallan de la misma forma, los parámetros de código de terminación y código de razón se establecen para describir el resultado común. Los registros de respuesta MQRR (si los proporciona la aplicación) no se establecen en este caso.

     Por ejemplo, si cada apertura es satisfactoria, el código de terminación se establece en MQCC_OK y el código de razón se establece en MQRC_NONE; si cada apertura falla porque no existe ninguna de las colas, los parámetros se establecen en MQCC_FAILED y MQRC_UNKNOWN_OBJECT_NAME.
   - Si las operaciones abiertas para las colas de la lista de distribución no son todas satisfactorias o fallan de la misma forma:
     - El parámetro de código de finalización se establece en MQCC_WARNING si al menos una apertura se ha realizado correctamente y en MQCC_FAILED si todo ha fallado.

- El parámetro de código de razón se establece en MQRC_MULTIPLE_REASON.
- Los registros de respuesta (si los proporciona la aplicación) se establecen en los códigos de terminación individuales y los códigos de razón para las colas de la lista de distribución.

5. Cuando una lista de distribución se ha abierto correctamente, el descriptor de contexto *Hobj* devuelto por la llamada se puede utilizar en llamadas MQPUT posteriores para colocar mensajes en las colas de la lista de distribución y en una llamada MQCLOSE para renunciar al acceso a la lista de distribución. La única opción de cierre válida para una lista de distribución es MQCO_NONE.

   La llamada MQPUT1 también se puede utilizar para colocar un mensaje en una lista de distribución; la estructura MQOD que define las colas de la lista se especifica como un parámetro en dicha llamada.
6. Cada destino abierto correctamente en la lista de distribución cuenta como un descriptor de contexto independiente al comprobar si la aplicación ha superado el número máximo permitido de descriptores de contexto (consulte el atributo de gestor de colas *MaxHandles*). Esto es cierto incluso cuando dos o más de los destinos de la lista de distribución se resuelven en la misma cola física. Si la llamada MQOPEN o MQPUT1 para una lista de distribución haría que el número de descriptores de contexto utilizados por la aplicación excediera *MaxHandles*, la llamada falla con el código de razón MQRC_HANDLE_NOT_AVAILABLE.
7. Cada destino que se abre correctamente tiene el valor de su atributo *OpenOutputCount* incrementado en uno. Si dos o más de los destinos de la lista de distribución se resuelven en la misma cola física, el atributo *OpenOutputCount* de dicha cola se incrementa en el número de destinos de la lista de distribución que se resuelven en dicha cola.
8. Cualquier cambio en las definiciones de cola que hubiera hecho que un descriptor de contexto no fuera válido si las colas se hubieran abierto individualmente (por ejemplo, un cambio en la vía de acceso de resolución), no hace que el descriptor de contexto de lista de distribución no sea válido. Sin embargo, da como resultado una anomalía para esa cola concreta cuando se utiliza el descriptor de contexto de lista de distribución en una llamada MQPUT posterior.
9. Una lista de distribución sólo puede contener un destino.

Colas remotas

Las notas siguientes se aplican al uso de colas remotas.

Una cola remota se puede especificar de una de dos maneras en el parámetro *ObjDesc* de esta llamada.

- Especificando para *ObjectName* el nombre de una definición local de la cola remota. En este caso, *ObjectQMgrName* hace referencia al gestor de colas local y se puede especificar como espacios en blanco o (en el lenguaje de programación C) como una serie nula.

  La validación de seguridad realizada por el gestor de colas local verifica que el usuario tiene autorización para abrir la definición local de la cola remota.
- Especificando para *ObjectName* el nombre de la cola remota tal como la conoce el gestor de colas remoto. En este caso, *ObjectQMgrName* es el nombre del gestor de colas remoto.

  La validación de seguridad realizada por el gestor de colas local verifica que el usuario está autorizado a enviar mensajes a la cola de transmisión como resultado del proceso de resolución de nombres.

En cualquier caso:

- El gestor de colas local no envía mensajes al gestor de colas remoto para comprobar que el usuario tiene autorización para colocar mensajes en la cola.
- Cuando llega un mensaje al gestor de colas remoto, el gestor de colas remoto puede rechazarlo porque el usuario que ha originado el mensaje no está autorizado.

Consulte los campos *ObjectName* y *ObjectQMgrName* descritos en “MQOD-Descriptor de objeto” en la página 454 para obtener más información.

Objetos

Seguridad

Las notas siguientes están relacionadas con los aspectos de seguridad del uso de MQOPEN.

El gestor de colas realiza comprobaciones de seguridad cuando se emite una llamada MQOPEN, para verificar que el identificador de usuario bajo el que se ejecuta la aplicación tiene el nivel de autorización adecuado antes de que se permita el acceso. La comprobación de autorización se realiza en el nombre del objeto que se está abriendo, y no en el nombre o nombres, lo que resulta después de que se haya resuelto un nombre.

Si el objeto que se está abriendo es una cola alias que apunta a un objeto de tema, el gestor de colas realiza una comprobación de seguridad en el nombre de cola alias, antes de realizar una comprobación de seguridad para el tema como si el objeto de tema se hubiera utilizado directamente.

Si el objeto que se está abriendo es un objeto de tema, ya sea con *ObjectName* solo o utilizando *ObjectString* (con o sin una base *ObjectName*), el gestor de colas realiza la comprobación de seguridad utilizando la serie de tema resultante, tomada del objeto de tema especificado en *ObjectName*y, si es necesario, concatenándola con la proporcionada en *ObjectString*y, a continuación, buscando el objeto de tema más cercano en o por encima de ese punto del árbol de temas para realizar la comprobación de seguridad. Es posible que no sea el mismo objeto de tema que se ha especificado en *ObjectName*.

Si el objeto que se está abriendo es una cola modelo, el gestor de colas realiza una comprobación de seguridad completa con el nombre de la cola modelo y el nombre de la cola dinámica que se crea. Si la cola dinámica resultante se abre entonces de forma explícita, se realiza una comprobación de seguridad de recursos adicional con respecto al nombre de la cola dinámica.

Atributos

Las notas siguientes están relacionadas con los atributos.

Los atributos de un objeto pueden cambiar mientras una aplicación tiene el objeto abierto. En muchos casos, la aplicación no se da cuenta de ello, pero para determinados atributos el gestor de colas marca el descriptor de contexto como ya no válido. Estos atributos son:

- Cualquier atributo que afecte a la resolución de nombres del objeto. Esto se aplica independientemente de las opciones de apertura utilizadas e incluye lo siguiente:
  - Un cambio en el atributo *BaseQName* de una cola alias que está abierta.
  - Un cambio en el atributo *TargetType* de una cola alias que está abierta.
  - Un cambio en los atributos de cola *RemoteQName* o *RemoteQMgrName* , para cualquier descriptor de contexto que esté abierto para esta cola, o para una cola que se resuelva a través de esta definición como un alias de gestor de colas.
  - Cualquier cambio que haga que un descriptor de contexto abierto actualmente para una cola remota se resuelva en una cola de transmisión diferente, o que no se pueda resolver en una en absoluto. Por ejemplo, esto puede incluir:
    - Un cambio en el atributo *XmitQName* de la definición local de una cola remota, tanto si la definición se está utilizando para una cola como para un alias de gestor de colas.
    - (Sóloz/OS) Un cambio en el valor del atributo de gestor de colas *IntraGroupQueuing* , o un cambio en la definición de la cola de transmisión compartida (SYSTEM.QSG.TRANSMIT.QUEUE) utilizado por el agente de IGQ.

    Hay una excepción a esto: la creación de una nueva cola de transmisión. Un descriptor de contexto que se habría resuelto en esta cola si estuviera presente cuando se abrió el descriptor de contexto, pero que en su lugar se hubiera resuelto en la cola de transmisión predeterminada, no se ha convertido en no válido.
  - Un cambio en el atributo de gestor de colas *DefXmitQName* . En este caso, todos los descriptores de contexto abiertos que se han resuelto en la cola especificada anteriormente (que se ha resuelto sólo porque era la cola de transmisión predeterminada) se marcan como no válidos. Los manejadores que se han resuelto en esta cola por otras razones no se ven afectados.

- El atributo de cola *Shareability* , si hay dos o más descriptores de contexto que proporcionan actualmente acceso MQOO_INPUT_SHARED para esta cola, o para una cola que se resuelve en esta cola. Si es así, *todos* los descriptores de contexto que están abiertos para esta cola, o para una cola que se resuelve en esta cola, se marcan como no válidos, independientemente de las opciones de apertura.

  En z/OS, los descriptores de contexto descritos anteriormente se marcan como no válidos si uno o varios descriptores de contexto proporcionan actualmente acceso MQOO_INPUT_SHARED o MQOO_INPUT_EXCLUSIVE a la cola.
- El atributo de cola *Usage* , para todos los descriptores de contexto que están abiertos para esta cola, o para una cola que se resuelve en esta cola, independientemente de las opciones de apertura.

Cuando un descriptor de contexto se marca como no válido, todas las llamadas posteriores (distintas de MQCLOSE) que utilizan este descriptor de contexto fallan con el código de razón MQRC_OBJECT_CHANGED. La aplicación debe emitir una llamada MQCLOSE (utilizando el descriptor de contexto original) y, a continuación, volver a abrir la cola. Las actualizaciones no confirmadas en el descriptor de contexto antiguo de las llamadas satisfactorias anteriores se pueden seguir confirmando o restituyendo, según lo requiera la lógica de la aplicación.

Si el cambio de un atributo hace que esto suceda, utilice una versión de fuerza especial de la llamada.

Invocación en C

```
MQOPEN (Hconn, &ObjDesc, Options, &Hobj, &CompCode,
       &Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
CALL 'MQOPEN' USING HCONN, OBJDESC, OPTIONS, HOBJ, COMPCODE, REASON
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQOPEN (Hconn, ObjDesc, Options, Hobj, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
CALL MQOPEN,(HCONN,OBJDESC,OPTIONS,HOBJ,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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
```

Invocación en Visual Basic

```
MQOPEN Hconn, ObjDesc, Options, Hobj, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
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-Colocar mensaje

La llamada MQPUT coloca un mensaje en una cola o lista de distribución, o en un tema. La cola, la lista de distribución o el tema ya deben estar abiertos.

Sintaxis

MQPUT (*Hconn*, *Hobj*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Razón*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* lo ha devuelto una llamada MQCONN o MQCONNX anterior.

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn*:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Hobj
Tipo: MQHOBJ - entrada

Este descriptor de contexto representa la cola a la que se añade el mensaje o el tema en el que se publica el mensaje. El valor de *Hobj* lo ha devuelto una llamada MQOPEN anterior que ha especificado la opción MQOO_OUTPUT.

MsgDesc
Tipo: MQMD - entrada/salida

Esta estructura describe los atributos del mensaje que se envía y recibe información sobre el mensaje una vez completada la solicitud de colocación. Para obtener detalles, consulte "MQMD - Descriptor de mensaje" en la página 392.

Si la aplicación proporciona un MQMD version-1 , los datos de mensaje pueden tener como prefijo una estructura MQMDE para especificar valores para los campos que existen en el MQMD version-2 pero no en el MQMD version-1. El campo *Formato* de MQMD debe establecerse en MQFMT_MD_EXTENSION para indicar que existe un MQMDE. Consulte "MQMDE-Extensión de descriptor de mensaje" en la página 445 para obtener más detalles.

La aplicación no necesita proporcionar una estructura MQMD si se proporciona un manejador de mensajes válido en los campos *OriginalMsgHandle* o *NewMsgHandle* de la estructura MQPMO. Si no se proporciona nada en uno de estos campos, el descriptor del mensaje se toma del descriptor asociado con los manejadores de mensajes.

Si utiliza o tiene previsto utilizar salidas de API, le recomendamos que proporcione explícitamente una estructura MQMD y no utilice los descriptores de mensaje asociados con los descriptores de mensaje. Esto se debe a que la salida de API asociada a la llamada MQPUT o MQPUT1 no puede determinar qué valores MQMD utiliza el gestor de colas para completar la solicitud MQPUT o MQPUT1 .

PutMsgOpts
Tipo: MQPMO-entrada/salida

Para obtener detalles, consulte "MQPMO-Opciones de transferencia de mensajes" en la página 475.

BufferLength
Tipo: MQLONG - entrada

La longitud del mensaje en *Buffer*. Cero es válido e indica que el mensaje no contiene datos de aplicación. El límite superior para *BufferLength* depende de varios factores:

- Si el destino es una cola local o se resuelve en una cola local, el límite superior depende de si:
  - El gestor de colas local da soporte a la segmentación.
  - La aplicación emisora especifica el distintivo que permite al gestor de colas segmentar el mensaje. Este distintivo es MQMF_SEGMENTATION_ALLOWED y se puede especificar en un MQMD de version-2 o en un MQMDE utilizado con un MQMD de version-1 .

  Si se cumplen ambas condiciones, *BufferLength* no puede exceder de 999 999 999 menos el valor del campo *Offset* en MQMD. El mensaje lógico más largo que se puede transferir es, por lo tanto, 999 999 999 bytes (cuando *Offset* es cero). Sin embargo, las restricciones de recursos impuestas por el sistema operativo o el entorno en el que se ejecuta la aplicación pueden dar como resultado un límite inferior.

  Si una o ambas de las condiciones anteriores no se cumplen, *BufferLength* no puede superar el atributo *MaxMsgLength* de la cola y el atributo *MaxMsgLength* del gestor de colas.
- Si el destino es una cola remota o se resuelve en una cola remota, se aplican las condiciones para las colas locales, *pero en cada gestor de colas a través del cual debe pasar el mensaje para llegar a la cola de destino*; en concreto:
  1. La cola de transmisión local utilizada para almacenar el mensaje temporalmente en el gestor de colas local
  2. Colas de transmisión intermedias (si las hay) utilizadas para almacenar el mensaje en los gestores de colas de la ruta entre los gestores de colas local y de destino
  3. La cola de destino en el gestor de colas de destino

Por lo tanto, el mensaje más largo que se puede transferir está gobernado por el más restrictivo de estas colas y gestores de colas.

Cuando un mensaje está en una cola de transmisión, la información adicional reside en los datos del mensaje, y esto reduce la cantidad de datos de aplicación que pueden transportarse. En esta situación, reste los bytes MQ_MSG_HEADER_LENGTH de los valores *MaxMsgLength* de las colas de transmisión al determinar el límite para *BufferLength*.

Nota: Sólo se puede diagnosticar de forma síncrona el incumplimiento de la condición 1 (con el código de razón MQRC_MSG_TOO_BIG_FOR_Q o MQRC_MSG_TOO_BIG_FOR_Q_MGR) cuando se transfiere el mensaje. Si no se cumplen las condiciones 2 o 3, el mensaje se redirige a una cola de mensajes no entregados, ya sea en un gestor de colas intermedio o en el gestor de colas de destino. Si esto sucede, se genera un mensaje de informe si el remitente lo ha solicitado.

Buffer
Tipo: MQBYTExBufferLongitud-entrada

Se trata de un almacenamiento intermedio que contiene los datos de aplicación que se van a enviar. El almacenamiento intermedio debe estar alineado en un límite adecuado a la naturaleza de los datos del mensaje. La alineación de 4 bytes es adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen estructuras de cabecera WebSphere MQ), pero algunos mensajes pueden requerir una alineación más estricta. Por ejemplo, un mensaje que contiene un entero binario de 64 bits puede requerir una alineación de 8 bytes.

Si *Buffer* contiene datos numéricos o de caracteres, establezca los campos *CodedCharSetId* y *Encoding* del parámetro *MsgDesc* en los valores adecuados para los datos; esto permite al receptor del mensaje convertir los datos (si es necesario) en el juego de caracteres y la codificación utilizados por el receptor.

Nota: Todos los demás parámetros de la llamada MQPUT deben estar en el juego de caracteres y la codificación del gestor de colas local (proporcionados por el atributo de gestor de colas *CodedCharSetId* y MQENC_NATIVE).

En el lenguaje de programación C, el parámetro se declara como puntero a void; la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si el parámetro *BufferLength* es cero, no se hará referencia a *Buffer*; en tal caso, la dirección del parámetro que han pasado los programas escritos en C o el ensamblador System/390 puede ser nula.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

El código de razón que califica *CompCode*.

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_WARNING:

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') El mensaje lógico no está completo.

MQRC_INCONSISTENT_PERSISTENCE
(2185, X'889 ') Especificación de persistencia incoherente.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_MULTIPLE_RAZONES
(2136, X'858 ') Se han devuelto varios códigos de razón.

MQRC_PRIORITY_EXCEEDS_MAXIMUM
(2049, X'801 ') La prioridad de mensaje sobrepasa el valor máximo soportado.

MQRC_UNKNOWN_REPORT_OPTION
(2104, X'838 ') No se reconocen las opciones de informe en el descriptor de mensaje.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ALIAS_TARGTYPE_CAMBIADO
(2480, X'09B0') El tipo de destino de suscripción ha cambiado de cola a objeto de tema.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BACKED_OUT
(2003, X'7D3') Unidad de trabajo restituida.

MQRC_BUFFER_ERROR
(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CALL_INTERRUPTED
(2549, X'9F5') MQPUT o MQCMIT se ha interrumpido y el proceso de reconexión no puede restablecer un resultado definido.

MQRC_CF_STRUC_FAILED
(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CFGR_ERROR
(2416, X' 970 ') La estructura de parámetro de grupo PCF MQCFGR en los datos del mensaje no es válida.

MQRC_CFH_ERROR
(2235, X'8BB') La estructura de cabecera PCF no es válida.

MQRC_CFIF_ERROR
(2414, X'96E') La estructura del parámetro de filtro de enteros PCF en los datos del mensaje no es válida.

MQRC_CFIL_ERROR
(2236, X'8BC') Estructura de parámetro de lista de enteros PCF o estructura de parámetro de lista de enteros PCIF*64 no válida.

MQRC_CFIN_ERROR
(2237, X'8BD') Estructura de parámetro de entero PCF o estructura de parámetro de entero PCIF*64 no válida.

MQRC_CFSF_ERROR
(2415, X'96F') La estructura del parámetro de filtro de serie PCF en los datos del mensaje no es válida.

MQRC_CFSL_ERROR
(2238, X'8BE') La estructura de parámetro de lista de series PCF no es válida.

MQRC_CFST_ERROR
(2239, X'8BF') La estructura del parámetro de serie PCF no es válida.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') CICS ha rechazado la solicitud de espera.

MQRC_CLUSTER_EXIT_ERROR
(2266, X'8DA') La salida de carga de trabajo del clúster ha fallado.

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') La resolución de nombres de clúster ha fallado.

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Error de recurso de clúster.

MQRC_COD_NOT_VALID_FOR_XCF_Q
(2106, X'83A') La opción de informe COD no es válida para la cola XCF.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

ERROR DE MQRC_CONTENT_
2554 (X'09FA') No se ha podido analizar el contenido del mensaje para determinar si el mensaje debe entregarse a un suscriptor con un selector de mensajes ampliado.

MQRC_CONTEXT_HANDLE_ERROR
(2097, X'831 ') El descriptor de contexto de cola al que se hace referencia no guarda el contexto.

MQRC_CONTEXT_NOT_AVAILABLE
(2098, X'832 ') Contexto no disponible para el descriptor de contexto de cola al que se hace referencia.

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_DH_ERROR
(2135, X'857 ') La estructura de cabecera de distribución no es válida.

MQRC_DLH_ERROR
(2141, X'85D') La estructura de cabecera de letra muerda no es válida.

MQRC_EPH_ERROR
(2420, X' 974 ') La estructura PCF incorporada no es válida.

MQRC_EXPIRY_ERROR
(2013, X'7DD') Tiempo de caducidad no válido.

MQRC_FEEDBACK_ERROR
(2014, X'7DE') Código de comentarios no válido.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GROUP_ID_ERROR
(2258, X'8D2') Identificador de grupo no válido.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HEADER_ERROR
(2142, X'85E') La estructura de cabecera MQ no es válida.

MQRC_HOBJ_ERROR
(2019, X'7E3') El manejador de objeto no es válido.

MQRC_IIH_ERROR
(2148, X'864 ') IMS estructura de cabecera de información no válida.

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') El mensaje lógico no está completo.

MQRC_INCONSISTENT_PERSISTENCE
(2185, X'889 ') Especificación de persistencia incoherente.

MQRC_INCONSISTENT_UOW
(2245, X'8C5') Especificación incoherente de unidad de trabajo.

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MD_ERROR
(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MDE_ERROR
(2248, X'8C8') Extensión de descriptor de mensaje no válida.

MQRC_MISSING_REPLY_TO_Q
(2027, X'7EB') Falta la cola de respuesta o se ha utilizado MQPMO_SUPPRESS_REPLYTO

MQRC_MISSING_WIH
(2332, X'91C') Los datos de mensaje no empiezan por MQWIH.

MQRC_MSG_FLAGS_ERROR
(2249, X'8C9') Los distintivos de mensaje no son válidos.

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Longitud de mensaje mayor que el máximo para la cola.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') Longitud de mensaje mayor que el máximo para el gestor de colas.

MQRC_MSG_TYPE_ERROR
(2029, X'7ED') El tipo de mensaje en el descriptor de mensaje no es válido.

MQRC_MULTIPLE_RAZONES
(2136, X'858 ') Se han devuelto varios códigos de razón.

MQRC_NO_DESTINATIONS_AVAILABLE
(2270, X'8DE') No hay colas de destino disponibles.

MQRC_NOT_OPEN_FOR_OUTPUT
(2039, X'7F7') Cola no abierta para salida.

MQRC_NOT_OPEN_FOR_PASS_ALL
(2093, X'82D') Cola no abierta para pasar todo el contexto.

MQRC_NOT_OPEN_FOR_PASS_IDENT
(2094, X'82E') La cola no está abierta para el contexto de identidad de paso.

MQRC_NOT_OPEN_FOR_SET_ALL
(2095, X'82F') Cola no abierta para establecer todo el contexto.

MQRC_NOT_OPEN_FOR_SET_IDENT
(2096, X'830 ') La cola no está abierta para el contexto de identidad establecido.

MQRC_OBJECT_CHANGED
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

ERROR DE MQRC_OFFSET_
(2251, X'8CB') El desplazamiento de segmento de mensaje no es válido.

MQRC_OPEN_FAILED
(2137, X'859 ') El objeto no se ha abierto correctamente.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_ORIGINAL_LENGTH_ERROR
(2252, X'8CC') Longitud original no válida.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_PAGESET_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_PCF_ERROR
(2149, X'865 ') Estructuras PCF no válidas.

MQRC_PERSISTENCE_ERROR
(2047, X'7FF') Persistencia no válida.

MQRC_PERSISTENT_NOT_ALLOWED
(2048, X'800 ') La cola no admite mensajes persistentes.

MQRC_PMO_ERROR
(2173, X'87D') Estructura de opciones de transferencia de mensaje no válida.

MQRC_PMO_RECORD_FLAGS_ERROR
(2158, X'86E') Los distintivos de registro de mensajes de colocación no son válidos.

MQRC_PRIORITY_ERROR
(2050, X'802 ') La prioridad del mensaje no es válida.

MQRC_PUBLICATION_FAILURE
(2502, X'9C6') La publicación no se ha entregado a ninguno de los suscriptores.

MQRC_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida
(2051, X'803 ') Llamadas de colocación inhibidas para la cola, para la cola en la que se resuelve esta cola o el tema.

MQRC_PUT_MSG_RECORDS_ERROR
(2159, X'86F') Los registros de mensajes de colocación no son válidos.

MQRC_PUT_NO_RETENIDO
(2479, X'09AF') No se ha podido retener la publicación

MQRC_Q_DELETED
(2052, X'804') La cola se ha suprimido.

MQRC_Q_FULL
(2053, X'805 ') La cola ya contiene el número máximo de mensajes.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808 ') No hay espacio disponible en disco para la cola.

MQRC_RECONNECT_FAILED
(2548, X'9F4') Después de la reconexión, se ha producido un error al restablecer los descriptores de contexto para una conexión reconectable.

MQRC_RECS_PRESENT_ERROR
(2154, X'86A') Número de registros presentes no válido.

MQRC_REPORT_OPTIONS_ERROR
(2061, X'80D') Las opciones de informe del descriptor de mensaje no son válidas.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_RESPONSE_RECORDS_ERROR
(2156, X'86C') Los registros de respuesta no son válidos.

MQRC_RFH_ERROR
(2334, X'91E') La estructura MQRFH o MQRFH2 no es válida.

MQRC_RMH_ERROR
(2220, X'8AC') La estructura de cabecera de mensaje de referencia no es válida.

MQRC_SEGMENT_LENGTH_ZERO
(2253, X'8CD') La longitud de los datos en el segmento de mensaje es cero.

MQRC_SEGMENTS_NOT_SUPPORTED
(2365, X'93D') Segmentos no soportados.

MQRC_SELECTION_NOT_AVAILABLE
2551 (X'09F7') Existe un posible suscriptor para la publicación, pero el gestor de colas no puede comprobar si se envía la publicación al suscriptor.

MQRC_STOPPED_BY_CLUSTER_EXIT
(2188, X'88C') Llamada rechazada por salida de carga de trabajo de clúster.

MQRC_STORAGE_CLASS_ERROR
(2105, X'839 ') Error de clase de almacenamiento.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818 ') El soporte de punto de sincronización no está disponible.

ERROR MQRC_TM
(2265, X'8D9') La estructura de mensajes del desencadenante no es válida.

ERROR MQRC_TMC
(2191, X'88F') La estructura del mensaje desencadenante de caracteres no es válida.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WIH_ERROR
(2333, X'91D') La estructura MQWIH no es válida.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

MQRC_XQH_ERROR
(2260, X'8D4') La estructura de cabecera de cola de transmisión no es válida.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso de temas

1. Las notas siguientes se aplican al uso de temas:

   a. Cuando se utiliza MQPUT para publicar mensajes sobre un tema, donde uno o más suscriptores de ese tema no pueden recibir la publicación debido a un problema con su cola de suscriptores (por ejemplo, está llena), el código de razón devuelto a la llamada MQPUT y el comportamiento de entrega depende del valor de los atributos PMSGDLV o NPMSGDLV en el TOPIC. Tenga en cuenta que la entrega de una publicación a la cola de mensajes no entregados cuando se especifica MQRO_DEAD_LETTER_Q, o descartar el mensaje cuando se especifica MQRO_DISCARD_MSG, se considera una entrega correcta del mensaje. Si no se entrega ninguna de las publicaciones, MQPUT devuelve MQRC_PUBLICATION_FAILURE. Esto puede ocurrir en los siguientes casos:

      - Un mensaje se publica en un TOPIC con PMSGDLV o NPMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALL y cualquier suscripción (duradera o no) tiene una cola que no puede recibir la publicación.
      - Un mensaje se publica en un TOPIC con PMSGDLV o NPMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALLDUR y una suscripción duradera tiene una cola que no puede recibir la publicación.

      MQPUT puede devolver con MQRC_NONE aunque las publicaciones no se hayan podido entregar a algunos suscriptores en los casos siguientes:

      - Un mensaje se publica en un TOPIC con PMSGDLV o NPMSGDLV (dependiendo de la persistencia del mensaje) establecido en ALLAVAIL y cualquier suscripción, duradera o no, tiene una cola que no puede recibir la publicación.
      - Un mensaje se publica en un TOPIC con PMSGDLV o NPMSGDLV (en función de la persistencia del mensaje) establecido en ALLDUR y una suscripción no duradera tiene una cola que no puede recibir la publicación.

      Puede utilizar el atributo de tema USEDLQ para determinar si se utiliza la cola de mensajes no entregados cuando los mensajes de publicación no se pueden entregar a su cola de suscriptor correcta. Para obtener más información sobre el uso de USEDLQ, consulte DEFINE TOPIC.

   b. Si no hay suscriptores para el tema que se está utilizando, el mensaje publicado no se envía a ninguna cola y se descarta. No importa si el mensaje es persistente o no persistente, o si tiene una caducidad ilimitada o tiene una hora de caducidad, todavía se descarta si no hay suscriptores. La excepción a esto es si el mensaje se va a retener, en cuyo caso, aunque no se envía a las colas de ningún suscriptor, se almacena en el tema que se va a entregar a cualquier suscripción nueva o a cualquier suscriptor que solicite publicaciones retenidas utilizando MQSUBRQ.

MQPUT y MQPUT1

Puede utilizar las llamadas MQPUT y MQPUT1 para colocar mensajes en una cola; la llamada a utilizar depende de las circunstancias

- Utilice la llamada MQPUT para colocar varios mensajes en la *misma* cola.

  En primer lugar, se emite una llamada MQOPEN que especifica la opción MQOO_OUTPUT, seguida de una o más solicitudes MQPUT para añadir mensajes a la cola; finalmente, la cola se cierra con una llamada MQCLOSE. Esto proporciona un mejor rendimiento que el uso repetido de la llamada MQPUT1 .
- Utilice la llamada MQPUT1 para colocar sólo *un* mensaje en una cola.

  Esta llamada encapsula las llamadas MQOPEN, MQPUT y MQCLOSE en una sola llamada, minimizando el número de llamadas que se deben emitir.

Colas de destino

Las notas siguientes se aplican al uso de colas de destino:

1. Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes,el orden de esos mensajes se conserva si se cumplen las condiciones detalladas. Algunas condiciones se aplican tanto a las colas de destino locales como a las remotas; otras condiciones se aplican sólo a las colas de destino remotas.

   Condiciones que se aplican a las colas de destino locales y remotas

   - Todas las llamadas MQPUT están dentro de la misma unidad de trabajo, o ninguna de ellas está dentro de una unidad de trabajo.

     Tenga en cuenta que cuando los mensajes se colocan en una cola determinada dentro de una sola unidad de trabajo, los mensajes de otras aplicaciones pueden intercalarse con la secuencia de mensajes de la cola.
   - Todas las llamadas MQPUT se realizan utilizando el mismo descriptor de objeto *Hobj*.

     En algunos entornos, la secuencia de mensajes también se conserva cuando se utilizan distintos manejadores de objetos, si las llamadas se realizan desde la misma aplicación. El significado de *la misma aplicación* viene determinado por el entorno:

     – En z/OS, la aplicación es:

       - Para CICS, la tarea CICS
       - Para IMS, la tarea
       - Para el lote z/OS , la tarea

     – En IBM i, la aplicación es el trabajo.

     – En sistemas Windows y UNIX , la aplicación es la hebra.
   - Los mensajes tienen todos la misma prioridad.
   - Los mensajes no se colocan en una cola de clúster con MQOO_BIND_NOT_FIXED especificado (o con MQOO_BIND_AS_Q_DEF en vigor cuando el atributo de cola DefBind tiene el valor MQBND_BIND_NOT_FIXED).

   Condiciones adicionales que se aplican a las colas de destino remoto

   - Sólo hay una vía de acceso desde el gestor de colas emisor al gestor de colas de destino.

     Si algunos mensajes de la secuencia pueden ir a una vía de acceso diferente (por ejemplo, debido a la reconfiguración, el equilibrio de tráfico o la selección de vía de acceso en función del tamaño del mensaje), el orden de los mensajes en el gestor de colas de destino no se puede garantizar.
   - Los mensajes no se colocan temporalmente en colas de mensajes no entregados en los gestores de colas de envío, intermedios o de destino.

Si uno o varios de los mensajes se colocan temporalmente en una cola de mensajes no entregados (por ejemplo, porque una cola de transmisión o la cola de destino está llena temporalmente), los mensajes pueden llegar a la cola de destino fuera de secuencia.

- Los mensajes son todos persistentes o todos no persistentes.

  Si un canal de la ruta entre los gestores de colas de envío y de destino tiene su atributo *NonPersistentMsgSpeed* establecido en MQNPMS_FAST, los mensajes no persistentes pueden saltar por delante de los mensajes persistentes, lo que da como resultado que el orden de los mensajes persistentes relativos a los mensajes no persistentes no se conserve. Sin embargo, se conserva el orden de los mensajes persistentes relativos entre sí y de los mensajes no persistentes relativos entre sí.

Si estas condiciones no se cumplen, puede utilizar grupos de mensajes para conservar el orden de los mensajes, pero esto requiere que las aplicaciones de envío y recepción utilicen el soporte de agrupación de mensajes. Para obtener más información sobre los grupos de mensajes, consulte:

- MQMD - Campo MsgFlags
- MQPMO_LOGICAL_ORDER
- MQGMO_LOGICAL_ORDER

Listas de distribución

Las siguientes notas se aplican al uso de listas de distribución.

Las listas de distribución están soportadas en los entornos siguientes: AIX, HP-UX, IBM i, Solaris, Linux, Windows, más clientes MQI de WebSphere MQ conectados a estos sistemas.

1. Puede colocar mensajes en una lista de distribución utilizando version-1 o un MQPMO version-2 . Si utiliza un MQPMO version-1 (o un MQPMO version-2 con *RecsPresent* igual a cero), la aplicación no puede proporcionar registros de mensajes de colocación ni registros de respuesta. No puede identificar las colas que encuentran errores si el mensaje se envía correctamente a algunas colas de la lista de distribución y no a otras.

   Si la aplicación proporciona registros de mensajes de colocación o registros de respuesta, establezca el campo *Version* en MQPMO_VERSION_2.

   También puede utilizar un MQPMO version-2 para enviar mensajes a una sola cola que no esté en una lista de distribución, asegurándose de que *RecsPresent* sea cero.
2. Los parámetros de código de terminación y código de razón se establecen de la forma siguiente:

   - Si todas las colocaciones en las colas de la lista de distribución son satisfactorias o fallan de la misma forma, el código de terminación y los parámetros de código de razón se establecen para describir el resultado común. Los registros de respuesta MQRR (si los proporciona la aplicación) no se establecen en este caso.

     Por ejemplo, si cada transferencia es satisfactoria, el código de terminación y el código de razón se establecen en MQCC_OK y MQRC_NONE; si cada transferencia falla porque todas las colas están inhibidas para la transferencia, los parámetros se establecen en MQCC_FAILED y MQRC_PUT_inhibitITED.
   - Si las colocaciones en las colas de la lista de distribución no son todas satisfactorias o fallan de la misma forma:
     - El parámetro de código de terminación se establece en MQCC_WARNING si al menos una operación de transferencia ha sido satisfactoria, y en MQCC_FAILED si todas las operaciones han fallado.
     - El parámetro de código de razón se establece en MQRC_MULTIPLE_REASON.
     - Los registros de respuesta (si los proporciona la aplicación) se establecen en los códigos de terminación individuales y los códigos de razón para las colas de la lista de distribución.

Si la colocación en un destino falla porque la apertura para dicho destino ha fallado, los campos del registro de respuesta se establecen en MQCC_FAILED y MQRC_OPEN_FAILED; dicho destino se incluye en *InvalidDestCount*.

3. Si un destino de la lista de distribución se resuelve en una cola local, el mensaje se coloca en esa cola en formato normal (es decir, no como un mensaje de lista de distribución). Si más de un destino se resuelve en la misma cola local, se coloca un mensaje en la cola para cada destino.

   Si un destino de la lista de distribución se resuelve en una cola remota, se coloca un mensaje en la cola de transmisión adecuada. Cuando varios destinos se resuelven en la misma cola de transmisión, se puede colocar en la cola de transmisión un único mensaje de lista de distribución que contenga esos destinos, incluso si esos destinos no estaban adyacentes en la lista de destinos proporcionada por la aplicación. Sin embargo, esto sólo se puede hacer si la cola de transmisión da soporte a mensajes de lista de distribución (consulte DistLists).

   Si la cola de transmisión no soporta listas de distribución, se coloca una copia del mensaje en formato normal en la cola de transmisión para cada destino que utiliza dicha cola de transmisión.

   Si una lista de distribución con los datos de mensaje de aplicación es demasiado grande para una cola de transmisión, el mensaje de lista de distribución se divide en mensajes de lista de distribución más pequeños, cada uno de los cuales contiene menos destinos. Si los datos del mensaje de aplicación solo se ajustan a la cola, los mensajes de lista de distribución no se pueden utilizar en absoluto, y el gestor de colas genera una copia del mensaje en formato normal para cada destino que utiliza dicha cola de transmisión.

   Si distintos destinos tienen una prioridad de mensaje o una persistencia de mensaje diferentes (esto puede ocurrir cuando la aplicación especifica MQPRI_PRIORITY_AS_Q_DEF o MQPER_PERSISTENCE_AS_Q_DEF), los mensajes no se conservan en el mismo mensaje de lista de distribución. En su lugar, el gestor de colas genera tantos mensajes de lista de distribución como sean necesarios para acomodar los diferentes valores de prioridad y persistencia.

4. Una colocación en una lista de distribución puede dar como resultado:

   - Un único mensaje de lista de distribución, o
   - Un número de mensajes de lista de distribución más pequeños, o
   - Una combinación de mensajes de lista de distribución y mensajes normales, o
   - Sólo mensajes normales.

   Cuál de los anteriores se produce depende de si:

   - Los destinos de la lista son locales, remotos o una mezcla.
   - Los destinos tienen la misma prioridad de mensaje y persistencia de mensaje.
   - Las colas de transmisión pueden contener mensajes de lista de distribución.
   - Las longitudes máximas de mensajes de las colas de transmisión son lo suficientemente grandes para acomodar el mensaje en formato de lista de distribución.

   Sin embargo, independientemente de cuál de las situaciones anteriores, cada mensaje *físico* resultante (es decir, cada mensaje normal o mensaje de lista de distribución resultante de la operación de transferir) cuenta como un solo mensaje *uno* cuando:

   - Comprobación de si la aplicación ha superado el número máximo permitido de mensajes en una unidad de trabajo (consulte el atributo de gestor de colas *MaxUncommittedMsgs*).
   - Comprobando si se cumplen las condiciones de desencadenamiento.
   - Incrementando las profundidades de cola y comprobando si se superaría la profundidad máxima de cola de las colas.

5. Cualquier cambio en las definiciones de cola que hubiera hecho que un descriptor de contexto no fuera válido si las colas se hubieran abierto individualmente (por ejemplo, un cambio en la vía de acceso de resolución), no hace que el descriptor de contexto de lista de distribución no sea válido. Sin embargo, da como resultado una anomalía para esa cola concreta cuando se utiliza el descriptor de contexto de lista de distribución en una llamada MQPUT posterior.

Cabeceras

Si un mensaje se coloca con una o más estructuras de cabecera WebSphere MQ al principio de los datos de mensaje de aplicación, el gestor de colas realiza determinadas comprobaciones en las estructuras de cabecera para verificar que son válidas. Si el gestor de colas detecta un error, la llamada falla con un código de razón adecuado. Las comprobaciones realizadas varían en función de las estructuras concretas que estén presentes:

- Las comprobaciones sólo se realizan si se utiliza un MQMD version-2 o posterior en la llamada MQPUT o MQPUT1 . Las comprobaciones no se realizan si se utiliza un MQMD version-1 , aunque haya un MQMDE al principio de los datos del mensaje.
- Las estructuras que no están soportadas por el gestor de colas local y las estructuras que siguen a la primera MQDLH del mensaje no se validan.
- El gestor de colas valida completamente las estructuras MQDH y MQMDE.
- El gestor de colas valida parcialmente otras estructuras (no se comprueban todos los campos).

Las comprobaciones generales realizadas por el gestor de colas son las siguientes:

- El campo *StrucId* debe ser válido.
- El campo *Version* debe ser válido.
- El campo *StrucLength* debe especificar un valor lo suficientemente grande para incluir la estructura más cualquier dato de longitud variable que forme parte de la estructura.
- El campo *CodedCharSetId* no debe ser cero, o un valor negativo que no es válido (MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR y MQCCSI_UNDEFINED *no* son válidos en la mayoría de las estructuras de cabecera de WebSphere MQ).
- El parámetro *BufferLength* de la llamada debe especificar un valor que sea lo suficientemente grande para incluir la estructura (la estructura no debe extenderse más allá del final del mensaje).

Además de los controles generales de las estructuras, deberán cumplirse las siguientes condiciones:

- La suma de las longitudes de las estructuras de un mensaje PCF debe ser igual a la longitud especificada por el parámetro *BufferLength* en la llamada MQPUT o MQPUT1 . Un mensaje PCF es un mensaje que tiene un nombre de formato de MQFMT_ADMIN, MQFMT_EVENT o MQFMT_PCF.
- Una estructura WebSphere MQ no se debe truncar, excepto en las situaciones siguientes en las que se permiten estructuras truncadas:
  - Mensajes que son mensajes de informe.
  - Mensajes PCF.
  - Mensajes que contienen una estructura MQDLH. (Las estructuras *que siguen* a la primera MQDLH se pueden truncar; las estructuras que preceden a la MQDLH no se pueden truncar.)
- Una estructura de WebSphere MQ no debe dividirse en dos o más segmentos; la estructura debe estar totalmente contenida en un segmento.

Almacenamiento intermedio

Para el lenguaje de programación Visual Basic, se aplican los puntos siguientes:

- Si el tamaño del parámetro *Buffer* es menor que la longitud especificada por el parámetro *BufferLength* , la llamada falla con el código de razón MQRC_BUFFER_LENGTH_ERROR.
- El parámetro *Buffer* se declara como si fuese de tipo `String`. Si los datos que se van a colocar en la cola no son del tipo `String`, utilice elLlamada MQPUTAny en lugar de MQPUT.

  La llamada MQPUTAny tiene los mismos parámetros que la llamada MQPUT, excepto que el parámetro *Buffer* se declara como de tipo `Any`, lo que permite colocar cualquier tipo de datos en la cola. No obstante, esto significa que *Buffer* no se puede comprobar para asegurarse de que su tamaño sea al menos de *BufferLength* bytes.

Invocación en C

```
MQPUT (Hconn, Hobj, &MsgDesc, &PutMsgOpts, BufferLength, Buffer,
      &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
     CALL 'MQPUT' USING HCONN, HOBJ, MSGDESC, PUTMSGOPTS, BUFFERLENGTH,
                        BUFFER, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQPUT (Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer,
           CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
CALL MQPUT,(HCONN,HOBJ,MSGDESC,PUTMSGOPTS,BUFFERLENGTH, X
               BUFFER,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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
```

Invocación en Visual Basic

```
MQPUT Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer, CompCode,
      Reason
```

Declare los parámetros como se indica a continuación:

```
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 -Colocar un mensaje

La llamada MQPUT1 coloca un mensaje en una cola, o lista de distribución, o en un tema.

No es necesario que la cola, la lista de distribución o el tema estén abiertos.

Sintaxis

MQPUT1 (*Hconn*, *ObjDesc*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* lo ha devuelto una llamada MQCONN o MQCONNX anterior.

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn*:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

ObjDesc
Tipo: MQOD-entrada/salida

Es una estructura que identifica la cola a la que se añade el mensaje o el tema en el que se publica el mensaje. Para obtener más detalles, consulte “MQOD-Descriptor de objeto” en la página 454.

Si la estructura es una cola, el usuario debe tener autorización para abrir la cola para salida. La cola **no** debe ser una cola modelo.

MsgDesc
Tipo: MQMD - entrada/salida

Esta estructura describe los atributos del mensaje que se envía y recibe información de retorno una vez completada la solicitud de colocación. Para obtener más detalles, consulte "MQMD - Descriptor de mensaje" en la página 392.

Si la aplicación proporciona un MQMD version-1 , los datos de mensaje pueden tener como prefijo una estructura MQMDE para especificar valores para los campos que existen en el MQMD version-2 pero no en el MQMD version-1. Establezca el campo *Formato* de MQMD en MQFMT_MD_EXTENSION para indicar que hay un MQMDE presente. Consulte "MQMDE-Extensión de descriptor de mensaje" en la página 445 para obtener más detalles.

La aplicación no necesita proporcionar una estructura MQMD si se proporciona un manejador de mensajes válido en el campo *MsgHandle* de la estructura MQGMO o en los campos *OriginalMsgHandle* o *NewMsgHandle* de la estructura MQPMO. Si no se proporciona nada en uno de estos campos, el descriptor del mensaje se toma del descriptor asociado con los manejadores de mensajes.

PutMsgOpts
Tipo: MQPMO-entrada/salida

Para obtener más detalles, consulte "MQPMO-Opciones de transferencia de mensajes" en la página 475.

BufferLength
Tipo: MQLONG - entrada

La longitud del mensaje en *Buffer*. Cero es válido e indica que el mensaje no contiene datos de aplicación. El límite superior depende de varios factores; consulte la descripción del parámetro *BufferLength* de la llamada MQPUT para obtener más detalles.

Buffer
Tipo: MQBYTExBufferLongitud-entrada

Es un almacenamiento intermedio que contiene los datos de mensaje de aplicación que se van a enviar. Alinee el almacenamiento intermedio en un límite adecuado según la naturaleza de los datos del mensaje. La alineación de 4 bytes es adecuada para la mayoría de los mensajes (incluidos los mensajes que contienen estructuras de cabecera de WebSphere MQ), pero es posible que algunos mensajes requieran una alineación más estricta. Por ejemplo, un mensaje que contiene un entero binario de 64 bits puede requerir una alineación de 8 bytes.

Si *Buffer* contiene datos numéricos o de caracteres, establezca los campos *CodedCharSetId* y *Encoding* del parámetro *MsgDesc* en los valores adecuados para los datos; esto permite al receptor del mensaje convertir los datos (si es necesario) en el juego de caracteres y la codificación utilizados por el receptor.

Nota: Todos los demás parámetros de la llamada MQPUT1 deben estar en el juego de caracteres y la codificación del gestor de colas local (proporcionados por el atributo de gestor de colas *CodedCharSetId* y MQENC_NATIVE).

En el lenguaje de programación C, el parámetro se declara como puntero a void; la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si el parámetro *BufferLength* es cero, no se hará referencia a *Buffer*; en tal caso, la dirección del parámetro que han pasado los programas escritos en C o el ensamblador System/390 puede ser nula.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

El código de razón que califica *CompCode*.

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_WARNING:

MQRC_MULTIPLE_RAZONES
(2136, X'858 ') Se han devuelto varios códigos de razón.

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') El grupo de mensajes no está completo.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') El mensaje lógico no está completo.

MQRC_PRIORITY_EXCEEDS_MAXIMUM
(2049, X'801 ') La prioridad de mensaje sobrepasa el valor máximo soportado.

MQRC_UNKNOWN_REPORT_OPTION
(2104, X'838 ') Opciones de informe en descriptor de mensaje no reconocidas.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ALIAS_BASE_Q_TYPE_ERROR
(2001, X'7D1') La cola base de alias no es un tipo válido.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BACKED_OUT
(2003, X'7D3') Unidad de trabajo restituida.

MQRC_BUFFER_ERROR
(2004, X'7D4') El parámetro almacenamiento intermedio no es válido.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_NOT_AVAILABLE
(2345, X' 929 ') recurso de acoplamiento no disponible.

MQRC_CF_STRUC_AUTH_FAILED
(2348, X'92C') La comprobación de autorización de la estructura del recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_ERROR
(2349, X'92D') La estructura del recurso de acoplamiento no es válida.

MQRC_CF_STRUC_FAILED
(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') La cabecera de lista de la estructura de recurso de acoplamiento se está utilizando.

MQRC_CFGR_ERROR
(2416, X' 970 ') La estructura de parámetro de grupo PCF MQCFGR en los datos del mensaje no es válida.

MQRC_CFH_ERROR
(2235, X'8BB') La estructura de cabecera PCF no es válida.

MQRC_CFIF_ERROR
(2414, X'96E') La estructura del parámetro de filtro de enteros PCF en los datos del mensaje no es válida.

MQRC_CFIL_ERROR
(2236, X'8BC') Estructura de parámetro de lista de enteros PCF o estructura de parámetro de lista de enteros PCIF*64 no válida.

MQRC_CFIN_ERROR
(2237, X'8BD') Estructura de parámetro de entero PCF o estructura de parámetro de entero PCIF*64 no válida.

MQRC_CFSF_ERROR
(2415, X'96F') La estructura del parámetro de filtro de serie PCF en los datos del mensaje no es válida.

MQRC_CFSL_ERROR
(2238, X'8BE') La estructura de parámetro de lista de series PCF no es válida.

MQRC_CFST_ERROR
(2239, X'8BF') La estructura del parámetro de serie PCF no es válida.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') CICS ha rechazado la solicitud de espera.

MQRC_CLUSTER_EXIT_ERROR
(2266, X'8DA') La salida de carga de trabajo del clúster ha fallado.

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') La resolución de nombres de clúster ha fallado.

MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Error de recurso de clúster.

MQRC_COD_NOT_VALID_FOR_XCF_Q
(2106, X'83A') La opción de informe COD no es válida para la cola XCF.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_QUIESCING
(2202, X'89A') Conexión en fase de inmovilización.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

ERROR DE MQRC_CONTENT_
2554 (X'09FA') No se ha podido analizar el contenido del mensaje para determinar si el mensaje se puede entregar a un suscriptor con un selector de mensajes ampliado.

MQRC_CONTEXT_HANDLE_ERROR
(2097, X'831 ') El descriptor de contexto de cola al que se hace referencia no guarda el contexto.

MQRC_CONTEXT_NOT_AVAILABLE
(2098, X'832 ') Contexto no disponible para el descriptor de contexto de cola al que se hace referencia.

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_DB2_NOT_AVAILABLE
(2342, X'926') El subsistema DB2 no está disponible.

MQRC_DEF_XMIT_Q_TYPE_ERROR
(2198, X'896 ') La cola de transmisión predeterminada no es local.

MQRC_DEF_XMIT_Q_USAGE_ERROR
(2199, X'897 ') Error de uso de cola de transmisión predeterminado.

MQRC_DH_ERROR
(2135, X'857 ') La estructura de cabecera de distribución no es válida.

MQRC_DLH_ERROR
(2141, X'85D') La estructura de cabecera de letra muerda no es válida.

MQRC_EPH_ERROR
(2420, X' 974 ') La estructura PCF incorporada no es válida.

MQRC_EXPIRY_ERROR
(2013, X'7DD') Tiempo de caducidad no válido.

MQRC_FEEDBACK_ERROR
(2014, X'7DE') Código de comentarios no válido.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Conflicto de unidades de trabajo global.

MQRC_GROUP_ID_ERROR
(2258, X'8D2') Identificador de grupo no válido.

MQRC_HANDLE_IN_USE_FOR_UOW
(2353, X'931') Descriptor de contexto que se utiliza para la unidad de trabajo global.

MQRC_HANDLE_NOT_AVAILABLE
(2017, X'7E1') No hay más manejadores disponibles.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HEADER_ERROR
(2142, X'85E') La estructura de cabecera de WebSphere MQ no es válida.

MQRC_IIH_ERROR
(2148, X'864 ') IMS estructura de cabecera de información no válida.

MQRC_LOCAL_UOW_CONFLICT
(2352, X'930') La unidad de trabajo global entra en conflicto con la unidad de trabajo local.

MQRC_MD_ERROR
(2026, X'7EA') El descriptor de mensaje no es válido.

MQRC_MDE_ERROR
(2248, X'8C8') Extensión de descriptor de mensaje no válida.

MQRC_MISSING_REPLY_TO_Q
(2027, X'7EB') Falta la cola de respuestas.

MQRC_MISSING_WIH
(2332, X'91C') Los datos de mensaje no empiezan por MQWIH.

MQRC_MSG_FLAGS_ERROR
(2249, X'8C9') Los distintivos de mensaje no son válidos.

MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') El número de secuencia de mensaje no es válido.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Longitud de mensaje mayor que el máximo para la cola.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') Longitud de mensaje mayor que el máximo para el gestor de colas.

MQRC_MSG_TYPE_ERROR
(2029, X'7ED') El tipo de mensaje en el descriptor de mensaje no es válido.

MQRC_MULTIPLE_RAZONES
(2136, X'858 ') Se han devuelto varios códigos de razón.

MQRC_NO_DESTINATIONS_AVAILABLE
(2270, X'8DE') No hay colas de destino disponibles.

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_OBJECT_IN_USE
(2042, X'7FA') El objeto ya está abierto con opciones en conflicto.

MQRC_OBJECT_LEVEL_INCOMPATIBLE
(2360, X' 938 ') Nivel de objeto no compatible.

MQRC_OBJECT_NAME_ERROR
(2152, X'868 ') Nombre de objeto no válido.

MQRC_OBJECT_NOT_UNIQUE
(2343, X' 927 ') Objeto no exclusivo.

MQRC_OBJECT_Q_MGR_NAME_ERROR
(2153, X'869 ') Nombre de gestor de colas de objeto no válido.

MQRC_OBJECT_RECORDS_ERROR
(2155, X'86B') Los registros de objeto no son válidos.

MQRC_OBJECT_TYPE_ERROR
(2043, X'7FB') Tipo de objeto no válido.

MQRC_OD_ERROR
(2044, X'7FC') La estructura de descriptor de objeto no es válida.

ERROR DE MQRC_OFFSET_
(2251, X'8CB') El desplazamiento de segmento de mensaje no es válido.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_ORIGINAL_LENGTH_ERROR
(2252, X'8CC') Longitud original no válida.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_PAGESET_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_PCF_ERROR
(2149, X'865 ') Estructuras PCF no válidas.

MQRC_PERSISTENCE_ERROR
(2047, X'7FF') Persistencia no válida.

MQRC_PERSISTENT_NOT_ALLOWED
(2048, X'800 ') La cola no admite mensajes persistentes.

MQRC_PMO_ERROR
(2173, X'87D') Estructura de opciones de transferencia de mensaje no válida.

MQRC_PMO_RECORD_FLAGS_ERROR
(2158, X'86E') Los distintivos de registro de mensajes de colocación no son válidos.

MQRC_PRIORITY_ERROR
(2050, X'802 ') La prioridad del mensaje no es válida.

MQRC_PUBLICATION_FAILURE
(2502, X'9C6') La publicación no se ha entregado a ninguno de los suscriptores.

MQRC_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida
(2051, X'803 ') Llamadas de colocación inhibidas para la cola.

MQRC_PUT_MSG_RECORDS_ERROR
(2159, X'86F') Los registros de mensajes de colocación no son válidos.

MQRC_Q_DELETED
(2052, X'804') La cola se ha suprimido.

MQRC_Q_FULL
(2053, X'805 ') La cola ya contiene el número máximo de mensajes.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_QUIESCING
(2161, X'871') El gestor de colas se está desactivando temporalmente.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808 ') No hay espacio disponible en disco para la cola.

MQRC_Q_TYPE_ERROR
(2057, X'809 ') Tipo de cola no válido.

MQRC_RECS_PRESENT_ERROR
(2154, X'86A') Número de registros presentes no válido.

MQRC_REMOTE_Q_NAME_ERROR
(2184, X'888 ') El nombre de cola remota no es válido.

MQRC_REPORT_OPTIONS_ERROR
(2061, X'80D') Las opciones de informe del descriptor de mensaje no son válidas.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_RESPONSE_RECORDS_ERROR
(2156, X'86C') Los registros de respuesta no son válidos.

MQRC_RFH_ERROR
(2334, X'91E') La estructura MQRFH o MQRFH2 no es válida.

MQRC_RMH_ERROR
(2220, X'8AC') La estructura de cabecera de mensaje de referencia no es válida.

MQRC_SECURITY_ERROR
(2063, X'80F') Se ha producido un error de seguridad.

MQRC_SEGMENT_LENGTH_ZERO
(2253, X'8CD') La longitud de los datos en el segmento de mensaje es cero.

MQRC_SELECTION_NOT_AVAILABLE
2551 (X'09F7') Existe un posible suscriptor para la publicación, pero el gestor de colas no puede comprobar si se envía la publicación al suscriptor.

MQRC_STOPPED_BY_CLUSTER_EXIT
(2188, X'88C') Llamada rechazada por salida de carga de trabajo de clúster.

MQRC_STORAGE_CLASS_ERROR
(2105, X'839 ') Error de clase de almacenamiento.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890') El soporte de almacenamiento externo está lleno.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') No pueden manejarse más mensajes dentro de la unidad de trabajo actual.

MQRC_SYNCPOINT_NOT_AVAILABLE
(2072, X'818 ') El soporte de punto de sincronización no está disponible.

ERROR MQRC_TM
(2265, X'8D9') La estructura de mensajes del desencadenante no es válida.

ERROR MQRC_TMC
(2191, X'88F') La estructura del mensaje desencadenante de caracteres no es válida.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

MQRC_UNKNOWN_ALIAS_BASE_Q
(2082, X'822 ') Cola base alias desconocida.

MQRC_UNKNOWN_DEF_XMIT_Q
(2197, X'895 ') Cola de transmisión predeterminada desconocida.

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825 ') Nombre de objeto desconocido.

MQRC_UNKNOWN_OBJECT_Q_MGR
(2086, X'826 ') Gestor de colas de objetos desconocido.

MQRC_UNKNOWN_REMOTE_Q_MGR
(2087, X'827 ') Gestor de colas remoto desconocido.

MQRC_UNKNOWN_XMIT_Q
(2196, X'894 ') Cola de transmisión desconocida.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X'932') Ha fallado el listado en la unidad de trabajo global.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X'933') No se soporta la mezcla de llamadas de unidad de trabajo.

MQRC_UOW_NOT_AVAILABLE
(2255, X'8CF') La unidad de trabajo no está disponible para ser utilizada por el gestor de colas.

MQRC_WIH_ERROR
(2333, X'91D') La estructura MQWIH no es válida.

MQRC_ERR_CF_LEVEL
(2366, X'93E') La estructura del recurso de acoplamiento tiene un nivel incorrecto.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') La versión del MQMD suministrado es incorrecta.

MQRC_XMIT_Q_TYPE_ERROR
(2091, X'82B') Cola de transmisión no local.

MQRC_XMIT_Q_USAGE_ERROR
(2092, X'82C') Cola de transmisión con un uso incorrecto.

MQRC_XQH_ERROR
(2260, X'8D4') La estructura de cabecera de cola de transmisión no es válida.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso

1. Las llamadas MQPUT y MQPUT1 se pueden utilizar para colocar mensajes en una cola; la llamada que se debe utilizar depende de las circunstancias:

   - Utilice la llamada MQPUT para colocar varios mensajes en la *misma* cola.

     En primer lugar, se emite una llamada MQOPEN que especifica la opción MQOO_OUTPUT, seguida de una o más solicitudes MQPUT para añadir mensajes a la cola; finalmente, la cola se cierra con una llamada MQCLOSE. Esto proporciona un mejor rendimiento que el uso repetido de la llamada MQPUT1 .

   - Utilice la llamada MQPUT1 para colocar sólo *un* mensaje en una cola.

     Esta llamada encapsula las llamadas MQOPEN, MQPUT y MQCLOSE en una sola llamada, minimizando el número de llamadas que se deben emitir.

2. Si una aplicación coloca una secuencia de mensajes en la misma cola sin utilizar grupos de mensajes,el orden de dichos mensajes se conserva si se cumplen determinadas condiciones. Sin embargo, en la mayoría de los entornos, la llamada MQPUT1 no satisface estas condiciones, por lo que no conserva el orden de los mensajes. En su lugar, se debe utilizar la llamada MQPUT en estos entornos. Consulte Notas de uso de MQPUT para obtener detalles.
3. La llamada MQPUT1 se puede utilizar para transferir mensajes a listas de distribución. Para obtener información general sobre esto, consulte las notas de uso para las llamadas MQOPEN y MQPUT.

   Las listas de distribución están soportadas en los entornos siguientes: clientes AIX, HP-UX, IBM i, Solaris, Linux, Windows, más WebSphere MQ conectados a estos sistemas.

   Las diferencias siguientes se aplican cuando se utiliza la llamada MQPUT1 :

   a. Si la aplicación proporciona registros de respuesta MQRR, se deben proporcionar utilizando la estructura MQOD; no se pueden proporcionar utilizando la estructura MQPMO.
   b. MQPUT1 nunca devuelve el código de razón MQRC_OPEN_FAILED en los registros de respuesta; si una cola no se puede abrir, el registro de respuesta para dicha cola contiene el código de razón resultante de la operación de apertura.

      Si una operación abierta para una cola se ejecuta correctamente con un código de terminación de MQCC_WARNING, el código de terminación y el código de razón en el registro de respuesta para dicha cola se sustituyen por los códigos de terminación y razón resultantes de la operación de transferencia.

      Al igual que con las llamadas MQOPEN y MQPUT, el gestor de colas establece los registros de respuesta (si se proporcionan) sólo cuando el resultado de la llamada no es el mismo para todas las colas de la lista de distribución; esto se indica mediante la finalización de la llamada con el código de razón MQRC_MULTIPLE_REASON.
4. Si se utiliza la llamada MQPUT1 para colocar un mensaje en una cola de clúster, la llamada se comporta como si se hubiera especificado MQOO_BIND_NOT_FIXED en la llamada MQOPEN.
5. Si un mensaje se coloca con una o más estructuras de cabecera de WebSphere MQ al principio de los datos del mensaje de aplicación, el gestor de colas realiza determinadas comprobaciones en las estructuras de cabecera para verificar que son válidas. Para obtener más información sobre esto, consulte las notas de uso para la llamada MQPUT.
6. Si surge más de una de las situaciones de aviso (consulte el parámetro *CompCode*), el código de razón devuelto es el *primero* de la lista siguiente que se aplica:

   a. MQRC_MULTIPLE_RAZONES
   b. MQRC_INCOMPLETE_MSG
   c. MQRC_INCOMPLETE_GROUP
   d. MQRC_PRIORITY_EXCEEDS_MAXIMUM o MQRC_UNKNOWN_REPORT_OPTION
7. Para el lenguaje de programación Visual Basic, se aplican los puntos siguientes:

- Si el tamaño del parámetro *Buffer* es menor que la longitud especificada por el parámetro *BufferLength* , la llamada falla con el código de razón MQRC_BUFFER_LENGTH_ERROR.
- El parámetro *Buffer* se declara como si fuese de tipo `String`. Si los datos que se van a colocar en la cola no son del tipo `String`, utilice elMQPUT1Any en lugar de MQPUT1.

  La llamada MQPUT1Any tiene los mismos parámetros que la llamada MQPUT1 , excepto que el parámetro *Buffer* se declara como de tipo Any, lo que permite colocar cualquier tipo de datos en la cola. No obstante, esto significa que *Buffer* no se puede comprobar para asegurarse de que su tamaño sea al menos de *BufferLength* bytes.

8. Cuando se emite una llamada MQPUT1 con MQPMO_SYNCPOINT, el comportamiento predeterminado cambia, de modo que la operación put se completa de forma asíncrona. Esto puede crear cambio de comportamiento en algunas aplicaciones que se basan en la devolución de determinados campos de las estructuras MQOD y MQMD y que ahora contienen valores no definidos. Una aplicación puede especificar MQPMO_SYNC_RESPONSE para asegurarse de que la operación de colocación se realiza de forma síncrona y de que se han completado todos los valores de campo adecuados.

Invocación en C

```
MQPUT1 (Hconn, &ObjDesc, &MsgDesc, &PutMsgOpts,
       BufferLength, Buffer, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
     CALL 'MQPUT1' USING HCONN, OBJDESC, MSGDESC, PUTMSGOPTS,
                         BUFFERLENGTH, BUFFER, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQPUT1 (Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
            CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
         CALL MQPUT1,(HCONN,OBJDESC,MSGDESC,PUTMSGOPTS,BUFFERLENGTH,   X
               BUFFER,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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
```

Invocación en Visual Basic

```
MQPUT1 Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
       CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
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 - Establecer atributos de objeto

Utilice la llamada MQSET para cambiar los atributos de un objeto representado por un descriptor de contexto. El objeto debe ser una cola.

Sintaxis

MQSET (*Hconn*, *Hobj*, *SelectorCount*, *Selectores*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *Compcode*, *Reason*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* lo ha devuelto una llamada MQCONN o MQCONNX anterior.

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn*:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Hobj
Tipo: MQHOBJ - entrada

Este descriptor de contexto representa el objeto de cola con atributos que se van a establecer. El descriptor de contexto ha sido devuelto por una llamada MQOPEN anterior que especificaba la opción MQOO_SET.

SelectorCount
Tipo: MQLONG - entrada

Este es el recuento de selectores que se proporcionan en la matriz *Selectors* . Es el número de atributos que se van a establecer. Cero es un valor válido. El número máximo permitido es 256.

Selectores
Tipo: MQLONGxSelectorRecuento-entrada

Esta es una matriz de selectores de atributos *SelectorCount* ; cada selector identifica un atributo (entero o carácter) con un valor que se va a establecer.

Cada selector debe ser válido para el tipo de cola que representa *Hobj* . Sólo se permiten determinados valores MQIA_ * y MQCA_ *; tal como se indica más adelante.

Los selectores se pueden especificar en cualquier orden. Los valores de atributo que corresponden a selectores de atributos enteros (selectores MQIA_ *) deben especificarse en *IntAttrs* en el mismo orden en el que aparecen estos selectores en *Selectors*. Los valores de atributo que corresponden a selectores de atributo de carácter (selectores MQCA_ *) deben especificarse en *CharAttrs* en el mismo orden en el que se producen dichos selectores. Los selectores MQIA_ * se pueden intercalar con los selectores MQCA_ *; sólo es importante el orden relativo dentro de cada tipo.

Puede especificar el mismo selector más de una vez; si lo hace, el último valor especificado para un selector determinado es el que entra en vigor.

Nota:

1. Los selectores de atributo de entero y carácter se asignan dentro de dos rangos diferentes; los selectores MQIA_ * residen dentro del rango de MQIA_FIRST a MQIA_LAST, y los selectores MQCA_ * dentro del rango de MQCA_FIRST a MQCA_LAST.

   Para cada rango, las constantes MQIA_LAST_USED y MQCA_LAST_USED definen el valor más alto que acepta el gestor de colas.
2. Si todos los selectores MQIA_ * aparecen en primer lugar, se pueden utilizar los mismos números de elemento para direccionar los elementos correspondientes en las matrices *Selectors* y *IntAttrs* .
3. Si el parámetro *SelectorCount* es cero, no se hace referencia a *Selectors* ; en este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler podría ser nula.

Los atributos que se pueden establecer se listan en la tabla siguiente. No se pueden establecer otros atributos utilizando esta llamada. Para los selectores de atributos MQCA_ *, la constante que define la longitud en bytes de la serie necesaria en *CharAttrs* se proporciona entre paréntesis.

Tabla 571. Selectores de atributos MQSET para colas		
Selector	**Descripción**	**Nota**
MQCA_TRIGGER_DATA	Datos de desencadenante (MQ_TRIGGER_DATA_LENGTH).	
MQIA_DIST_LISTS	Soporte de lista de distribución.	1
INHIBIDORES DE MQIA_GET	Si se permiten operaciones get.	
INHIBIDORES DE MQIA_PUT	Si se permiten las operaciones de colocación.	
MQIA_TRIGGER_CONTROL	Control de desencadenante.	
MQIA_PROFUNDIDAD	Profundidad del desencadenante.	
MQIA_TRIGGER_MSG_PRIORITY	Umbral de prioridad del mensaje para activaciones.	
MQIA_TIPO_TRIGGER_TYPE	Tipo de desencadenante.	

Nota:

1. Solo se admite en AIX, HP-UX, IBM i, Solaris, Linux, Windows, más clientes MQI de WebSphere MQ conectados a estos sistemas.

IntAttrCount
Tipo: MQLONG - entrada

Es el número de elementos de la matriz *IntAttrs* y debe ser como mínimo el número de selectores MQIA_ * en el parámetro *Selectors* . Cero es un valor válido si no hay ninguno.

IntAttrs
Tipo: MQLONGxIntAttrCount -entrada

Esta es una matriz de valores de atributo de entero de *IntAttrCount* . Estos valores de atributo deben estar en el mismo orden que los selectores MQIA_ * de la matriz *Selectors* .

Si el parámetro *IntAttrCount* o *SelectorCount* es cero, no se hace referencia a *IntAttrs* ; en este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler podría ser nula.

CharAttrLongitud
Tipo: MQLONG - entrada

Esta es la longitud en bytes del parámetro *CharAttrs* y debe ser como mínimo la suma de las longitudes de los atributos de caracteres especificados en la matriz *Selectors* . Cero es un valor válido si no hay selectores MQCA_ * en *Selectors*.

CharAttrs
Tipo: MQCHARxCharAttrLength -entrada

Es el almacenamiento intermedio que contiene los valores de atributo de carácter, concatenados entre sí. La longitud del almacenamiento intermedio la proporciona el parámetro *CharAttrLength* .

Los atributos de caracteres deben especificarse en el mismo orden que los selectores MQCA_ * en la matriz *Selectors* . La longitud de cada atributo de carácter es fija (consulte *Selectors*). Si el valor que se va a establecer para un atributo contiene menos caracteres no en blanco que la longitud definida del atributo, rellene el valor de *CharAttrs* a la derecha con espacios en blanco para que el valor del atributo coincida con la longitud definida del atributo.

Si el parámetro *CharAttrLength* o *SelectorCount* es cero, no se hace referencia a *CharAttrs* ; en este caso, la dirección de parámetro pasada por los programas escritos en C o System/390 assembler podría ser nula.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

El código de razón que califica *CompCode*.

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_API_EXIT_ERROR
(2374, X'946') La salida de la API ha fallado.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CF_STRUC_FAILED
(2373, X'945') La estructura de recurso de acoplamiento ha fallado.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') La estructura de recurso de acoplamiento se está utilizando.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') La cabecera de lista de la estructura de recurso de acoplamiento se está utilizando.

MQRC_CHAR_ATTR_LENGTH_ERROR
(2006, X'7D6') Longitud de atributos de caracteres no válida.

MQRC_CHAR_ATTRS_ERROR
(2007, X'7D7') Serie de atributos de carácter no válida.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') CICS ha rechazado la solicitud de espera.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') No tiene autorización para la conexión.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_DB2_NOT_AVAILABLE
(2342, X'926') El subsistema DB2 no está disponible.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_HOBJ_ERROR
(2019, X'7E3') El manejador de objeto no es válido.

MQRC_INHIBIDOR_VALOR_ERROR
(2020, X'7E4') El valor del atributo de cola con inhibición-get o con inhibición-put no es válido.

MQRC_INT_ATTR_COUNT_ERROR
(2021, X'7E5') Recuento de atributos enteros no válidos.

MQRC_INT_ATTRS_ARRAY_ERROR
(2023, X'7E7') Matriz de atributos enteros no válida.

MQRC_NOT_OPEN_FOR_SET
(2040, X'7F8') Cola no abierta para el conjunto.

MQRC_OBJECT_CHANGED
(2041, X'7F9') La definición de objeto ha cambiado desde que se ha abierto.

MQRC_OBJECT_DAMAGED
(2101, X'835') El objeto se ha dañado.

MQRC_PAGESET_ERROR
(2193, X'891') Error al acceder al conjunto de datos del conjunto de páginas.

MQRC_Q_DELETED
(2052, X'804') La cola se ha suprimido.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') El nombre del gestor de colas no es válido o no se conoce.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') El gestor de colas no está disponible para efectuar la conexión.

MQRC_Q_MGR_STOPPING
(2162, X'872') El gestor de colas está concluyendo.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SELECTOR_COUNT_ERROR
(2065, X'811 ') Recuento de selectores no válido.

MQRC_SELECTOR_ERROR
(2067, X'813 ') El selector de atributos no es válido.

MQRC_SELECTOR_LIMIT_EXCEEDED
(2066, X'812 ') Recuento de selectores demasiado grande.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') El programa de salida ha suprimido la llamada.

MQRC_TRIGGER_CONTROL_ERROR
(2075, X'81B') El valor del atributo trigger-control no es válido.

MQRC_TRIGGER_DEPTH_ERROR
(2076, X'81C') El valor del atributo de profundidad de desencadenante no es válido.

MQRC_TRIGGER_MSG_PRIORITY_ERR
(2077, X'81D') El valor del atributo trigger-message-priority no es válido.

MQRC_TRIGGER_TYPE_ERROR
(2078, X'81E') El valor del atributo de tipo desencadenante no es válido.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso

1. Utilizando esta llamada, la aplicación puede especificar una matriz de atributos enteros, o una colección de series de atributos de caracteres, o ambos. Si no se producen errores, todos los atributos

especificados se establecen simultáneamente. Si se produce un error (por ejemplo, si un selector no es válido o se intenta establecer un atributo en un valor que no es válido), la llamada falla y no se establece ningún atributo.

2. Los valores de los atributos se pueden determinar utilizando la llamada MQINQ; consulte "MQINQ- Consultar atributos de objeto" en la página 686 para obtener detalles.

   Nota: No todos los atributos con valores que se pueden consultar utilizando la llamada MQINQ pueden cambiar sus valores utilizando la llamada MQSET. Por ejemplo, no se pueden establecer atributos de objeto de proceso o de gestor de colas con esta llamada.

3. Los cambios de atributo se conservan entre los reinicios del gestor de colas (aparte de las alteraciones en las colas dinámicas temporales, que no sobreviven a los reinicios del gestor de colas).
4. No puede cambiar los atributos de una cola modelo utilizando la llamada MQSET. Sin embargo, si abre una cola modelo utilizando la llamada MQOPEN con la opción MQOO_SET, puede utilizar la llamada MQSET para establecer los atributos de la cola local dinámica creada por la llamada MQOPEN.
5. Si el objeto que se está definiendo es una cola de clúster, debe haber una instancia local de la cola de clúster para que la apertura sea satisfactoria.

Para obtener más información sobre los atributos de objeto, consulte:

- "Atributos para colas" en la página 816
- "Atributos de las listas de nombres" en la página 849
- "Atributos de las definiciones de proceso" en la página 851
- "Atributos para el gestor de colas" en la página 780

Invocación en C

```
MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
CALL 'MQSET' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,
                   INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,
                   CHARATTRS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
           IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

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

Declare los parámetros como se indica a continuación:

```
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
```

Invocación en Visual Basic

```
MQSET Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, CompCode, Reason
```

Declare los parámetros como se indica a continuación:

```
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-Establecer propiedad de mensaje

Utilice la llamada MQSET para establecer o modificar una propiedad de un descriptor de mensaje.

Sintaxis

MQSETMP (*Hconn*, *Hmsg*, *SetPropOpts*, *Name*, *PropDesc*, *Type*, *ValueLength*, *Value*, *Compcode*, *Reason*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas.

El valor debe coincidir con el descriptor de conexión que se ha utilizado para crear el descriptor de mensaje especificado en el parámetro *Hmsg* . Si el descriptor de contexto de mensaje se ha creado utilizando MQHC_UNASSOCIATED_HCONN, se debe establecer una conexión válida en la hebra estableciendo una propiedad del descriptor de contexto de mensaje; de lo contrario, la llamada falla con el código de razón MQRC_CONNECTION_BROKEN.

Msj
Tipo: MQHMSG-entrada

Este es el manejador de mensajes que se va a modificar. El valor ha sido devuelto por una llamada MQCRTMH anterior.

SetPropOpts
Tipo: MQSMPO-entrada

Controlar cómo se establecen las propiedades de mensaje.

Esta estructura permite a las aplicaciones especificar opciones que controlan cómo se establecen las propiedades de mensaje. La estructura es un parámetro de entrada en la llamada MQSETMP. Consulte MQSMPO para obtener más información.

Nombre
Tipo: MQCHARV-entrada

Es el nombre de la propiedad que se va a establecer.

Consulte Nombres de propiedad y Restricciones de nombre de propiedad para obtener más información sobre el uso de nombres de propiedad.

PropDesc
Tipo: MQPD-entrada/salida

Esta estructura se utiliza para definir los atributos de una propiedad, incluyendo:

- qué sucede si la propiedad no está soportada
- a qué contexto de mensaje pertenece la propiedad
- en qué mensajes se copia la propiedad a medida que fluye

Consulte MQPD para obtener más información sobre esta estructura.

type
Tipo: MQLONG - entrada

El tipo de datos de la propiedad que se está definiendo. Puede ser uno de los siguientes:

MQTYPE_BOOLEAN
Un valor booleano. *ValueLength* debe ser 4.

MQTYPE_BYTE_STRING
Una serie de bytes. *ValueLength* debe ser cero o mayor.

MQTYPE_INT8
Un entero con signo de 8 bits. *ValueLength* debe ser 1.

MQTYPE_INT16
Un entero con signo de 16 bits. *ValueLength* debe ser 2.

MQTYPE_INT32
Un entero con signo de 32 bits.*ValueLength* debe ser 4.

MQTYPE_INT64
Un entero con signo de 64 bits.*ValueLength* debe ser 8.

MQTYPE_FLOAT32
Un número de coma flotante de 32 bits. *ValueLength* debe ser 4.

Nota: este tipo no está soportado con aplicaciones que utilizan IBM COBOL for z/OS.

MQTYPE_FLOAT64
Un número de coma flotante de 64 bits. *ValueLength* debe ser 8.

Nota: este tipo no está soportado con aplicaciones que utilizan IBM COBOL for z/OS.

MQTYPE_STRING
Una serie de caracteres. *ValueLength* debe ser cero o mayor, o el valor especial MQVL_NULL_TERMINATED.

MQTYPE_NULL
La propiedad existe pero tiene un valor nulo.*ValueLength* debe ser cero.

ValueLength
Tipo: MQLONG - entrada

Longitud en bytes del valor de propiedad en el parámetro *Valor* . Cero sólo es válido para valores nulos o para series o series de bytes. Cero indica que la propiedad existe pero que el valor no contiene caracteres ni bytes.

El valor debe ser mayor o igual que cero o el siguiente valor especial si el parámetro *Tipo* tiene establecido MQTYPE_STRING:

MQVL_NULL_TERMINATED
El valor está delimitado por el primer nulo encontrado en la serie. El valor nulo no se incluye como parte de la serie. Este valor no es válido si no se ha establecido también MQTYPE_STRING.

Nota: El carácter nulo utilizado para terminar una serie si se establece MQVL_NULL_TERMINATED es un valor nulo del juego de caracteres del valor.

Valor
Tipo: MQBYTExValueLongitud-entrada

El valor de la propiedad que se va a establecer. El almacenamiento intermedio debe estar alineado en un límite adecuado a la naturaleza de los datos del valor.

En el lenguaje de programación C, el parámetro se declara como puntero a void; la dirección de cualquier tipo de datos se puede especificar como parámetro.

Si *ValueLength* es cero, no se hace referencia a *Valor* . En este caso, la dirección de parámetro que pasan los programas escritos en C o System/390 assembler puede ser nula.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

El código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_RFH_FORMAT_ERROR
(2421, X'0975 ') No se ha podido analizar una carpeta MQRFH2 que contiene propiedades.

Si *CompCode* es MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'089C') Adaptador no disponible.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') No se ha podido cargar el módulo de servicio del adaptador.

MQRC_ASID_MISMATCH
(2157, X'86D') Los ASID primario e inicial varían.

MQRC_BUFFER_ERROR
(2004, X'07D4') El parámetro de valor no es válido.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') El parámetro de longitud de valor no es válido.

MQRC_CALL_IN_PROGRESS
(2219, X'08AB') Se ha especificado una llamada MQI antes de que se completara la llamada anterior.

MQRC_HMSG_ERROR
(2460, X'099C') El puntero de manejador de mensajes no es válido.

MQRC_MSG_HANDLE_IN_USE
(2499, X'09C3') El descriptor de mensaje ya se está utilizando.

MQRC_OPTIONS_ERROR
(2046, X'07FE') Opciones no válidas o no coherentes.

MQRC_PD_ERROR
(2482, X'09B2') La estructura del descriptor de propiedades no es válida.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Nombre de propiedad no válido.

ERROR_TIPO_PROPIEDAD_MQRC
(2473, X'09A9') Tipo de datos de propiedad no válido.

MQRC_PROP_NUMBER_FORMAT_ERROR
(2472, X'09A8') Se ha encontrado un error de formato de número en los datos de valor.

MQRC_SMPO_ERROR
(2463, X'099F') Establecer estructura de opciones de propiedad de mensaje no válida.

MQRC_SOURCE_CCSID_ERROR
(2111, X'083F') Identificador de juego de caracteres codificado de nombre de propiedad no válido.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Invocación en C

```
MQSETMP (Hconn, Hmsg, &SetPropOpts, &Name, &PropDesc, Type,
ValueLength, &Value, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
     CALL 'MQSETMP' USING HCONN, HMSG, SETMSGOPTS, NAME, PROPDESC, TYPE,
                          VALUELENGTH, VALUE, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQSETMP (Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength,
              Value, CompCode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
         CALL MQSETMP,(HCONN,HMSG,SETMSGHOPTS,NAME,PROPDESC,TYPE,VALUELENGTH,
               VALUE,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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-Recuperar información de estado

Utilice la llamada MQSTAT para recuperar información de estado. El tipo de información de estado devuelta viene determinado por el valor de tipo especificado en la llamada.

Sintaxis

MQSTAT (*Hconn*, *Tipo*, *Stat* , *Compcode*, *Razón*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn* :

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

type
Tipo: MQLONG - entrada

Tipo de información de estado que se está solicitando. Los > valores válidos son:

MQSTAT_TYPE_ASYNC_ERROR
Devuelve información sobre operaciones de transferencia asíncronas anteriores.

MQSTAT_TYPE_RECONNECTION

Devuelve información sobre la reconexión. Si la conexión se está reconectando o no se ha podido volver a conectar, la información describe la anomalía que ha hecho que la conexión empezara a reconectarse.

Este valor sólo es válido para conexiones de cliente. Para otros tipos de conexión, la llamada falla con el código de razón **MQRC_ENVIRONMENT_ERROR**

MQSTAT_TYPE_RECONNECTION_ERROR

Devuelve información sobre una anomalía anterior relacionada con la reconexión. Si la conexión no se ha podido reconectar, la información describe la anomalía que ha hecho que la reconexión fallara.

Este valor sólo es válido para conexiones de cliente. Para otros tipos de conexión, la llamada falla con el código de razón **MQRC_ENVIRONMENT_ERROR** .

Estado

Tipo: MQSTS-entrada/salida

Estructura de información de estado. Para obtener más detalles, consulte "MQSTS-Estructura de informes de estado" en la página 568.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

El código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_API_EXIT_ERROR
(2374, X' 946 ') La salida de API ha fallado

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') No se ha podido cargar la salida de la API.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') La llamada MQI ha entrado antes de completar la llamada anterior.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Se ha perdido la conexión con el gestor de colas.

MQRC_CONNECTION_STOPPING
(2203, X'89B') La conexión está concluyendo.

MQRC_FUNCTION_NOT_SUPPORTED
(2298, X'8FA') La función solicitada no está disponible en el entorno actual.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_Q_MGR_STOPPING
(2162,X'872'-Se está deteniendo el gestor de colas

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_STAT_TYPE_ERROR
(2430, X'97E' Error con tipo MQSTAT

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_STS_ERROR
(2426, X'97A') Error con estructura MQSTS

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso

1. Una llamada a MQSTAT especificando un tipo de MQSTAT_TYPE_ASYNC_ERROR devuelve información sobre operaciones MQPUT y MQPUT1 asíncronas anteriores. La estructura MQSTS que se devuelve de la llamada MQSTAT contiene la primera información de error o aviso asíncrono registrada para dicha conexión. Si hay más errores o avisos a continuación del primero, normalmente no alteran estos valores. Sin embargo, si se produce un error con un código de terminación de MQCC_WARNING, en su lugar se devuelve una anomalía posterior con un código de terminación de MQCC_FAILED .
2. Si no se han producido errores desde que se estableció la conexión o desde la última llamada a MQSTAT , se devuelve un CompCode de MQCC_OK y una razón de MQRC_NONE en la estructura MQSTS .
3. Los recuentos del número de llamadas asíncronas que se han procesado bajo el descriptor de conexión se devuelven mediante tres campos de contador: PutSuccessCount, PutWarningCount y PutFailureCount. Estos contadores son incrementados por el gestor de colas cada vez que una operación asíncrona se procesa correctamente, tiene un aviso o falla (tenga en cuenta que, a efectos de contabilidad, una transferencia a una lista de distribución cuenta una vez por cola de destino en lugar de una vez por lista de distribución). Un contador no se incrementa más allá del valor positivo máximo AMQ_LONG_MAX.
4. Una llamada satisfactoria a MQSTAT da como resultado que se restablezca cualquier información de error o recuento anterior.
5. El comportamiento de MQSTAT depende del valor del parámetro MQSTAT Type que proporcione.
6. **MQSTAT_TYPE_ASYNC_ERROR**
    a. Una llamada a MQSTAT especificando un tipo de MQSTAT_TYPE_ASYNC_ERROR devuelve información sobre operaciones MQPUT y MQPUT1 asíncronas anteriores. La estructura MQSTS que se devuelve de la llamada MQSTAT contiene la primera información de error o aviso asíncrono registrada para dicha conexión. Si hay más errores o avisos a continuación del primero, normalmente no alteran estos valores. Sin embargo, si se produce un error con un código de terminación de MQCC_WARNING, en su lugar se devuelve una anomalía posterior con un código de terminación de MQCC_FAILED .
    b. Si no se han producido errores desde que se estableció la conexión o desde la última llamada a MQSTAT , se devuelve un CompCode de MQCC_OK y una razón de MQRC_NONE en la estructura MQSTS .
    c. Los recuentos del número de llamadas asíncronas que se han procesado bajo el descriptor de conexión se devuelven mediante tres campos de contador: PutSuccessCount, PutWarningCount y PutFailureCount. Estos contadores son incrementados por el gestor de colas cada vez que una operación asíncrona se procesa correctamente, tiene un aviso o falla (tenga en cuenta que, a efectos de contabilidad, una transferencia a una lista de distribución cuenta una vez por cola de destino en lugar de una vez por lista de distribución). Un contador no se incrementa más allá del valor positivo máximo AMQ_LONG_MAX.
    d. Una llamada satisfactoria a MQSTAT da como resultado que se restablezca cualquier información de error o recuento anterior.

MQSTAT_TYPE_RECONNECTION

Supongamos que llama a MQSTAT con Type establecido en MQSTAT_TYPE_RECONNECTION dentro de un manejador de sucesos durante la reconexión. Considere estos ejemplos.

El cliente está intentando la reconexión o no se ha podido volver a conectar.
CompCode en la estructura MQSTS es MQCC_FAILED y Reason puede ser MQRC_CONNECTION_BROKEN o MQRC_Q_MGR_QUIESCING . ObjectType es MQOT_Q_MGR, ObjectName es el nombre del gestor de colas y ObjectQMgrName está en blanco.

El cliente ha completado la reconexión correctamente o nunca se ha desconectado.
CompCode en la estructura MQSTS es MQCC_OK y Reason es MQRC_NONE

Las llamadas posteriores a MQSTAT devuelven los mismos resultados.

MQSTAT_TYPE_RECONNECTION_ERROR

Supongamos que llama a MQSTAT con Type establecido en MQSTAT_TYPE_RECONNECTION_ERROR en respuesta a la recepción de MQRC_RECONNECT_FAILED a una llamada MQI. Considere estos ejemplos.

Se ha producido un error de autorización cuando se estaba reabriendo una cola durante la reconexión con un gestor de colas diferente.
CompCode en la estructura MQSTS es MQCC_FAILED y Reason es la razón por la que la reconexión ha fallado, como por ejemplo MQRC_NOT_AUTHORIZED . ObjectType es el tipo de objeto que ha causado el problema, como por ejemplo MQOT_QUEUE, ObjectName es el nombre de la cola y ObjectQMgrName el nombre del gestor de colas propietario de la cola.

Se ha producido un error de conexión de socket durante la reconexión.
CompCode en la estructura MQSTS es MQCC_FAILED y Reason es la razón por la que la reconexión ha fallado, como por ejemplo MQRC_HOST_NOT_AVAILABLE . ObjectType es MQOT_Q_MGR, ObjectName es el nombre del gestor de colas y ObjectQMgrName está en blanco.

Las llamadas posteriores a MQSTAT devuelven los mismos resultados.

Invocación en C

```
MQSTAT (Hconn, StatType, &Stat, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN Hconn;        /* Connection Handle */
MQLONG StatType;    /* Status type */
MQSTS Stat;             /* Status information structure */
MQLONG CompCode;    /* Completion code */
MQLONG Reason;        /* Reason code qualifying CompCode */
```

Invocación en COBOL

```
CALL 'MQSTAT' USING HCONN, STATTYPE, STAT, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**    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.
```

Invocación en PL/I

```
call MQSTAT (Hconn, StatType, Stat, Compcode, Reason);
```

Declare los parámetros como se indica a continuación:

```
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 Invocación de Assembler

```
CALL MQSTAT,(HCONN,STATTYPE,STAT,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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 - Registrar suscripción

Utilice la llamada MQSUB para registrar la suscripción de aplicaciones a un tema determinado.

Sintaxis

MQSUB (*Hconn*, *SubDesc*, *Hobj*, *Hsub*, *CódigoComp* , *Razón*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* ha sido devuelto por una llamada MQCONN o MQCONNX anterior.

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn* :

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

SubDesc
Tipo: MQSD - entrada/salida

Esta es una estructura que identifica el objeto en uso que está registrando la aplicación. Consulte “MQSD - Descriptor de suscripción” en la página 542 para obtener más información.

Hobj
Tipo: MQHOBJ - entrada/salida

Este descriptor de contexto representa el acceso que se ha establecido para obtener los mensajes enviados a esta suscripción. Estos mensajes se pueden almacenar en una cola específica o el gestor de colas puede gestionar su almacenamiento sin utilizar ninguna cola específica.

Para utilizar una cola específica, debe asociarla con la suscripción al crear la suscripción. Puede hacerlo de dos maneras:

- Utilizando el mandato DEFINE SUB MQSC y proporcionando a ese mandato el nombre de un objeto de cola.
- Proporcionando este descriptor de contexto al llamar a MQSUB con MQSO_CREATE

Si se proporciona este descriptor de contexto como un parámetro de entrada en la llamada, debe ser un descriptor de contexto de objeto válido desvuelto de una llamada MQOPEN anterior de una cola utilizando al menos una de las opciones siguientes:

- MQOO_INPUT_*
- MQOO_BROWSE
- MQOO_OUTPUT (si la cola es una cola remota)

Si este no es el caso, la llamada falla con MQRC_HOBJ_ERROR. No puede ser un descriptor de contexto de objeto a una cola de alias que se resuelva en un objeto de tema. Si es así, la llamada falla con MQRC_HOBJ_ERROR.

Si el gestor de colas va a gestionar el almacenamiento de los mensajes enviados a esta suscripción, se debe establecer al crear la suscripción, utilizando la opción MQSO_MANAGED. A continuación, el gestor de colas devuelve este descriptor de contexto como un parámetro de salida en la llamada. El descriptor de contexto devuelto se conoce como descriptor de contexto gestionado. Si se especifica MQHO_NONE pero no se especifica MQSO_MANAGED, la llamada falla con MQRC_HOBJ_ERROR.

Cuando el gestor de colas le devuelve un descriptor de contexto gestionado, puede utilizarlo en una llamada MQGET o MQCB con o sin opciones browse, en una llamada MQINQ, o en MQCLOSE. No puede utilizarlo en MQPUT, MQSUB, MQSET; si se intenta, fallará con MQRC_NOT_OPEN_FOR_OUTPUT, MQRC_HOBJ_ERROR o MQRC_NOT_OPEN_FOR_SET.

Si esta suscripción se reanuda utilizando la opción MQSO_RESUME en la estructura MQSD, el descriptor de contexto se puede devolver a la aplicación en este parámetro estableciendo MQSO_MANAGED en MQHO_NONE. Puede hacer esto independientemente de si la suscripción utiliza o no un descriptor de contexto gestionado, y puede ser útil para proporcionar suscripciones creadas utilizando DEFINE SUB con el descriptor de contexto a la cola de suscripciones definida en ese mandato. En el caso donde se está reanudando una suscripción creada administrativamente, la cola se abre con MQOO_INPUT_AS_Q_DEF y MQOO_BROWSE. Si necesita especificar otras opciones, la aplicación debe abrir la cola de suscripciones explícitamente y proporcionar el descriptor de contexto de objeto en la llamada. Si hay un problema al abrir la cola, la llamada falla con MQRC_INVALID_DESTINATION. Si se proporciona *Hobj* , debe ser equivalente a *Hobj* en la llamada MQSUB original. Esto significa que si se proporciona un descriptor de contexto de objeto devuelto de una llamada MQOPEN, el descriptor de contexto debe ser a la misma cola que se ha utilizado anteriormente. Si no es la misma cola, la llamada falla con MQRC_HOBJ_ERROR.

Si esta suscripción se está alterando utilizando la opción MQSO_ALTER en la estructura MQSD, se puede proporcionar un *Hobj* diferente. Las publicaciones que se han entregado a la cola y que se han identificado anteriormente a través de este parámetro permanecen en dicha cola y es responsabilidad de la aplicación recuperar estos mensajes si el parámetro *Hobj* ahora representa una cola diferente.

La tabla resume el uso de este parámetro con varias opciones de suscripción:

Opciones	***Hobj***	**Descripción**
MQSO_CREATE + MQSO_MANAGED	Ignorado en la salida	Crea una suscripción con almacenamiento de mensajes gestionado por el gestor de colas
MQSO_CREATE	Un descriptor de contexto de objeto válido	Crea una suscripción que facilita una cola específica como destino de los mensajes.
MQSO_RESUME	MQHO_NONE	Reanuda una suscripción creada anteriormente independientemente de si está gestionada o no, y el gestor de colas devuelve el descriptor de contexto de objeto para que lo utilice la aplicación.

Opciones	*Hobj*	Descripción
MQSO_RESUME	Un descriptor de contexto de objeto coincidente válido	Reanuda una suscripción creada anteriormente que utiliza una cola específica como destino de los mensajes y utiliza un descriptor de contexto de objeto con opciones de apertura específicas.
MQSO_ALTER + MQSO_MANAGED	MQHO_NONE	Modifica una suscripción existente que utilizaba anteriormente una cola específica, por lo que ahora es una suscripción gestionada. La clase de destino (gestionado o no) no se puede cambiar.
MQSO_ALTER	Un descriptor de contexto de objeto válido	Modifica una suscripción existente, gestionada o no, para que ahora utilice una cola específica. Cuando no se utiliza la opción MQSO_MANAGED, la cola proporcionada se puede cambiar, pero no se puede cambiar la clase de destino (gestionada o no).

Tanto si se ha proporcionado como si se ha devuelto, *Hobj* debe especificarse en las llamadas MQGET o MQCB posteriores que deseen recibir los mensajes de publicación enviados a esta suscripción.

El descriptor de contexto *Hobj* ya no es válido cuando se emite la llamada MQCLOSE en él, o cuando la unidad de proceso que define el ámbito del descriptor de contexto termina (hasta que la aplicación se desconecta). El ámbito del descriptor de contexto de objeto devuelto es el mismo que el del descriptor de conexión especificado en la llamada. Consulte Hconn (MQHCONN) - salida para obtener información sobre el ámbito del descriptor de contexto. Un MQCLOSE del descriptor de contexto *Hobj* no afecta al descriptor de contexto *Hsub* .

Hsub

Tipo: MQHOBJ - salida

Este descriptor de contexto representa la suscripción que se ha realizado. Se puede utilizar para otras dos operaciones:

- Se puede utilizar en una llamada MQSUBRQ subsiguiente para solicitar que las publicaciones se envíen cuando se haya utilizado la opción MQSO_PUBLICATIONS_ON_REQUEST al realizar la suscripción.
- Se puede utilizar en una llamada MQCLOSE subsiguiente para eliminar la suscripción que se ha realizado. El descriptor de contexto *Hsub* deja de ser válido cuando se emite la llamada MQCLOSE o cuando termina la unidad de proceso que define el ámbito del descriptor de contexto. El ámbito del descriptor de contexto de objeto devuelto es el mismo que el del descriptor de conexión especificado en la llamada. Un MQCLOSE del descriptor de contexto *Hsub* no afecta al descriptor de contexto *Hobj* .

Este descriptor de contexto no se puede pasar a una llamada MQGET o MQCB. Debe utilizar el parámetro *Hobj* . No puede utilizar este descriptor de contexto en ninguna llamada WebSphere MQ que no sea MQCLOSE o MQSUBRQ. Si se pasa este descriptor de contexto a cualquier otra llamada WebSphere MQ , se genera MQRC_HOBJ_ERROR.

CompCode

Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria

MQCC_WARNING
Aviso (terminación parcial)

MQCC_FAILED
Llamada fallida

Razón
Tipo: MQLONG - salida

El código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK, el código de razón es el siguiente:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED, el código de razón es uno de los siguientes:

MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') La resolución de nombres de clúster ha fallado.

MQRC_DURABILITY_NOT_ALLOWED
2436 (X'0984') Ha fallado una llamada MQSUB utilizando la opción MQSO_DURABLE.

MQRC_FUNCTION_NOT_SUPPORTED
2298 (X'08FA') La función solicitada no está disponible en el entorno actual.

MQRC_HOBJ_ERROR
2019 (X'07E3') Descriptor de contexto de objeto Hobj no válido.

MQRC_IDENTITY_MISMATCH
2434 (X'0982') El nombre de suscripción coincide con una suscripción existente.

MQRC_NOT_AUTHORIZED
2035 (X'07F3') El usuario no está autorizado para realizar la operación.

MQRC_OBJECT_STRING_ERROR
2441 (X'0989') El campo Objectstring no es válido.

MQRC_OPTIONS_ERROR
2046 (X'07FE') El parámetro o campo Options contiene opciones que no son válidas, o una combinación de opciones que no es válida.

MQRC_Q_MGR_QUIESCING
2161 (X'0871') El gestor de colas se está poniendo en pausa.

MQRC_RECONNECT_Q_MGR_REQD
2555 (X'09FB'X) Se requiere la opción MQCNO_RECONNECT_Q_MGR.

MQRC_RETAINED_MSG_Q_ERROR
2525 (X'09DD') No se pueden recuperar las publicaciones retenidas que existen para la serie de tema suscrita.

MQRC_RETAINED_NOT_DELIVERED
2526 (X'09DE') Las publicaciones existentes para la serie de tema suscrita no se pueden entregar a la cola de destino de suscripción y no se pueden entregar a la cola de mensajes no entregados.

MQRC_SD_ERROR
2424 (X'0978') El descriptor de suscripción (MQSD) no es válido.

MQRC_SELECTION_NOT_AVAILABLE
2551 (X'09F7') La serie de selección no sigue la sintaxis del selector WebSphere MQ y no había ningún proveedor de selección de mensajes ampliado disponible.

MQRC_SELECTION_STRING_ERROR
2519 (X'09D7') La serie de selección se debe especificar tal como se describe en la documentación de la estructura MQCHARV.

MQRC_SELECTOR_SYNTAX_ERROR
2459 (X'099B') Se ha emitido una llamada MQOPEN, MQPUT1 o MQSUB pero la serie de selección que se ha especificado contenía un error de sintaxis.

MQRC_SUB_USER_DATA_ERROR
2431 (X'097F') El campo SubUserData no es válido.

MQRC_SUB_NAME_ERROR
2440 (X'0988') El campo SubName no es válido.

MQRC_SUB_ALREADY_EXISTS
2432 (X'0980') La suscripción ya existe.

MQRC_SUB_USER_DATA_ERROR
2431 (X'097F') El campo SubUserData no es válido.

MQRC_TOPIC_STRING_ERROR
2425 (X'0979') La serie de tema no es válida.

MQRC_UNKNOWN_OBJECT_NAME
2085 (X'0825 ') No se puede encontrar el objeto identificado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso

1. La suscripción se compone de un tema, nombrado utilizando el nombre abreviado de un objeto de tema predefinido, el nombre completo de la serie de tema o formado por la concatenación de dos partes. Consulte la descripción de *ObjectName* y *ObjectString* en “MQSD - Descriptor de suscripción” en la página 542.
2. El gestor de colas efectúa comprobaciones de seguridad cuando se emite una llamada MQSUB para verificar que el identificador de usuario con el que se está ejecutando la aplicación tiene el nivel de autorización adecuado antes de permitir el acceso. El objeto de tema adecuado se encuentra en la jerarquía de temas y se realiza una comprobación de autoridad en este objeto de tema para garantizar que se ha establecido la autoridad para suscripción. Si no se utiliza la opción MQSO_MANAGED, se realiza una comprobación de autoridad en la cola de destino para garantizar que se ha establecido la autoridad para salida. Si se utiliza la opción MQSO_MANAGED, no se realiza ninguna comprobación de autoridad en la cola gestionada para acceso de consulta o salida.
3. Si no proporciona un Hobj como entrada, la llamada MQSUB asigna dos descriptores de contexto, un descriptor de contexto de objeto (Hobj) y un descriptor de contexto de suscripción (Hsub).
4. El Hobj devuelto en la llamada MQSUB cuando se utiliza la opción MQSO_MANAGED se puede consultar para encontrar atributos como por ejemplo el umbral de retroceso o el nombre de recolocación excesiva en cola de retroceso. También puede consultar el nombre de la cola gestionada, pero no debe intentar abrir directamente esta cola.
5. Las suscripciones se pueden agrupar, lo que permite que se entregue una única publicación al grupo de suscripciones, incluso si más de una suscripción del grupo coincidía con la publicación. Las suscripciones se pueden agrupar utilizando la opción MQSO_GROUP_SUB, y para agrupar suscripciones estas deben
   - utilizar la misma cola con nombre (es decir, no utilizar la opción MQSO_MANAGED) en el mismo gestor de colas, representado por el parámetro Hobj en la llamada MQSUB
   - compartir el mismo SubCorrelId
   - estar en el mismo SubLevel

   Estos atributos definen el conjunto de suscripciones que se tienen en cuenta para formar parte de un grupo y que también son los atributos que no se pueden alterar si se agrupa una suscripción. Si se altera el SubLevel, se genera MQRC_SUBLEVEL_NOT_ALTERABLE, y si se altera cualquiera de los demás atributos (que se pueden cambiar si una suscripción no está agrupada) se genera MQRC_GROUPING_NOT_ALTERABLE.
6. Los campos del MQSD están rellenados cuando se devuelven de una llamada MQSUB que utiliza la opción MQSO_RESUME. El MQSD devuelto se puede pasar directamente en una llamada MQSUB que

utiliza la opción MQSO_ALTER con los cambios que es necesario realizar a la suscripción aplicada al MQSD. Algunos campos tienen algunas consideraciones especiales, tal como se indica en la tabla.

Salida MQSD de MQSUB	
Nombre de campo en MQSD	**Consideraciones especiales**
Opciones de acceso o creación	Algunas de las opciones se pueden restablecer cuando se devuelve la llamada MQSUB. Si, a continuación, reutiliza el MQSD en una llamada MQSUB, la opción necesaria se debe establecer explícitamente.
Opciones de durabilidad, Opciones de destino, Opciones de registro & Opciones de comodín	Estas opciones se establecen según corresponda
Opciones de publicación	Estas opciones se establecen según corresponda, excepto MQSO_NEW_PUBLICATIONS_ONLY, que sólo es aplicable a MQSO_CREATE.
Otras opciones	Estas opciones no se modifican en la devolución de una llamada MQSUB. Controlan cómo se emite la llamada de API y no se almacenan con la suscripción. Deben establecerse de la forma que sea necesaria en cualquier llamada MQSUB subsiguiente que vuelva a utilizar MQSD.
ObjectName	Este campo de sólo entrada no se modifica en la devolución de una llamada MQSUB.
ObjectString	Este campo de sólo entrada no se modifica en la devolución de una llamada MQSUB. El nombre de tema completo utilizado se devuelve en el campo *ResObjectString* , si se proporciona un almacenamiento intermedio.
AlternateUserId y AlternateSecurityId	Estos campos de sólo entrada no se modifican en la devolución de una llamada MQSUB. Controlan cómo se emite la llamada de API y no se almacenan con la suscripción. Deben establecerse de la forma que sea necesaria en cualquier llamada MQSUB subsiguiente que vuelva a utilizar MQSD.
SubExpiry	En la devolución de una llamada MQSUB mediante la opción MQSO_RESUME, este campo se establece en la caducidad original de la suscripción y no en el tiempo restante para la caducidad. Si a continuación reutiliza el MQSD en una llamada MQSUB utilizando la opción MQSO_ALTER, restablezca la caducidad de la suscripción para reiniciar la cuenta regresiva.
SubName	Este campo es un campo de entrada en una llamada MQSUB y no se modifica en la salida.

Salida MQSD de MQSUB *(continuación)*	
Nombre de campo en MQSD	**Consideraciones especiales**
SubUserData y SelectionString	Estos campos de longitud variable se devuelven en la salida de una llamada MQSUB utilizando la opción MQSO_RESUME, si se proporciona un almacenamiento intermedio, y también una longitud de almacenamiento intermedio positiva en *VSBufSize*. Si no se proporciona ningún almacenamiento intermedio, sólo se devuelve la longitud en el campo *VSLength* de MQCHARV. Si el almacenamiento intermedio proporcionado es menor que el espacio necesario para devolver el campo, sólo se devuelven *VSBufSize* bytes en el almacenamiento intermedio proporcionado. Si, a continuación, reutiliza el MQSD en una llamada MQSUB utilizando la opción MQSO_ALTER y no se proporciona un almacenamiento intermedio pero se proporciona un *VSLength* distinto de cero, si dicha longitud coincide con la longitud existente del campo, no se realiza ninguna modificación en el campo.
SubCorrelId y PubAccountingToken	Si no utiliza MQSO_SET_CORREL_ID, el gestor de colas genera *SubCorrelId* . Si no utiliza MQSO_SET_IDENTITY_CONTEXT, el gestor de colas genera *PubAccountingToken* . Estos campos se devuelven en el MQSD de una llamada MQSUB utilizando la opción MQSO_RESUME. Si el gestor de colas los genera, el valor generado se devuelve en una llamada MQSUB utilizando la opción MQSO_CREATE o MQSO_ALTER.
PubPriority, SubLevel & PubApplIdentityData	Estos campos se devuelven en el MQSD.
ResObjectString	Este campo de sólo salida se devuelve en el MQSD si se proporciona un almacenamiento intermedio.

Invocación en C

```
MQSUB (Hconn, &SubDesc, &Hobj, &Hsub, &CompCode, &Reason)
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
CALL 'MQSUB' USING HCONN, SUBDESC, HOBJ, HSUB, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

Invocación en PL/I

```
call MQSUB (Hconn, SubDesc, Hobj, Hsub, CompCode, Reason)
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
CALL MQSUB,(HCONN,SUBDESC,HOBJ,HSUB,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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-Solicitud de suscripción

Utilice la llamada MQSUBRQ para realizar una solicitud para la publicación retenida, cuando el suscriptor se haya registrado con MQSO_PUBLICATIONS_ON_REQUEST.

Sintaxis

MQSUBRQ (*Hconn*, *Hsub*, *Acción*, *SubRqOpts*, *Compcode*, *Reason*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas. El valor de *Hconn* lo ha devuelto una llamada MQCONN o MQCONNX anterior.

En z/OS para aplicaciones CICS y en IBM i para aplicaciones que se ejecutan en modalidad de compatibilidad, la llamada MQCONN se puede omitir y se puede especificar el valor siguiente para *Hconn*:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Hsub
Tipo: MQHOBJ - entrada

Este descriptor de contexto representa la suscripción para la que se va a solicitar una actualización. El valor de *Hsub* se ha devuelto de una llamada MQSUB anterior.

Action
Tipo: MQLONG - entrada

Este parámetro controla la acción concreta que se está solicitando en la suscripción. Se debe especificar el valor siguiente:

MQSR_ACTION_PUBLICATION

Esta acción solicita que se envíe una publicación de actualización para el tema especificado. Sólo se puede utilizar si el suscriptor ha especificado la opción MQSO_PUBLICATIONS_ON_REQUEST en la llamada MQSUB cuando ha realizado la suscripción. Si el gestor de colas tiene una publicación retenida para el tema, se envía al suscriptor. Si no es así, la llamada falla. Si a una aplicación se le envía una publicación que se ha retenido, esto se indica mediante la propiedad de mensaje MQIsRetained de dicha publicación.

Puesto que el tema de la suscripción existente representada por el parámetro Hsub puede contener comodines, el suscriptor puede recibir varias publicaciones retenidas.

SubRqOpts
Tipo: MQSRO-entrada/salida

Estas opciones controlan la acción de MQSUBRQ, consulte "MQSRO-Opciones de solicitud de suscripción" en la página 566 para obtener más detalles.

Si no se necesitan opciones, los programas escritos en C o S/390 assembler pueden especificar una dirección de parámetro nula en lugar de especificar la dirección de una estructura MQSRO.

CompCode
Tipo: MQLONG - salida

El código de terminación; es uno de los siguientes:

MQCC_OK
Realización satisfactoria

MQCC_WARNING
Aviso (terminación parcial)

MQCC_FAILED
Llamada fallida

Razón
Tipo: MQLONG - salida

El código de razón que califica *CompCode*.

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_FUNCTION_NOT_SUPPORTED
2298 (X'08FA') La función solicitada no está disponible en el entorno actual.

MQRC_NO_RETAINED_MSG
2437 (X'0985 ') No hay publicaciones retenidas almacenadas actualmente para este tema.

MQRC_OPTIONS_ERROR
2046 (X'07FE') El parámetro o campo Options contiene opciones que no son válidas, o una combinación de opciones que no es válida.

MQRC_Q_MGR_QUIESCING
2161 (X'0871') El gestor de colas se está poniendo en pausa.

MQRC_SRO_ERROR
2438 (X'0986 ') En la llamada MQSUBRQ, las opciones de solicitud de suscripción MQSRO no son válidas.

MQRC_RETAINED_MSG_Q_ERROR
2525 (X'09DD') No se pueden recuperar las publicaciones retenidas que existen para la serie de tema suscrita.

MQRC_RETAINED_NOT_DELIVERED
2526 (X'09DE') Las publicaciones existentes para la serie de tema suscrita no se pueden entregar a la cola de destino de suscripción y no se pueden entregar a la cola de mensajes no entregados.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Notas de uso

Las siguientes notas de uso se aplican al uso del código de acción MQSR_ACTION_PUBLICATION:

1. Si este verbo se completa correctamente, las publicaciones retenidas que coinciden con la suscripción especificada se han enviado a la suscripción y se pueden recibir utilizando MQGET o MQCB utilizando el Hobj devuelto en el verbo MQSUB original que ha creado la suscripción.
2. Si el tema suscrito por el verbo MQSUB original que ha creado la suscripción contenía un comodín, se puede enviar más de una publicación retenida. El número de publicaciones enviadas como resultado de esta llamada se registra en el campo NumPubs de la estructura SubRqOpts.
3. Si este verbo se completa con un código de razón de MQRC_NO_RETAINED_MSG, no había publicaciones retenidas actualmente para el tema especificado. #
4. Si este verbo se completa con un código de razón de MQRC_RETAINED_MSG_Q_ERROR o MQRC_RETAINED_NOT_DELIVERED, actualmente hay publicaciones retenidas para el tema especificado, pero se ha producido un error que significa que no se han podido entregar.
5. La aplicación debe tener una suscripción actual al tema para poder realizar esta llamada. Si la suscripción se ha realizado en una instancia anterior de la aplicación y no está disponible un descriptor de contexto válido para la suscripción, la aplicación debe llamar primero a MQSUB con la opción MQSO_RESUME para obtener un descriptor de contexto para utilizarla en esta llamada.
6. Las publicaciones se envían al destino que está registrado para su uso con la suscripción actual de esta aplicación. Si las publicaciones deben enviarse a otro lugar, la suscripción debe alterarse primero utilizando la llamada MQSUB con la opción MQSO_ALTER.

Invocación en C

```
MQSUB (Hconn, Hsub, Action, &SubRqOpts, &CompCode, &Reason)
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en COBOL

```
CALL 'MQSUBRQ' USING HCONN, HSUB, ACTION, SUBRQOPTS, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
** 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.
```

Invocación en PL/I

```
call MQSUBRQ (Hconn, Hsub, Action, SubRqOpts, CompCode, Reason)
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Invocación en ensamblador de alto nivel

```
CALL MQSUBRQ,(HCONN, HSUB, ACTION, SUBRQOPTS,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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
```

objeto, atributos de

Esta colección de temas lista sólo los objetos de WebSphere MQ que pueden ser objeto de una llamada a función MQINQ, y proporciona detalles de los atributos que se pueden consultar y los selectores que se van a utilizar.

Atributos para el gestor de colas

Algunos atributos de gestor de colas son fijos para implementaciones concretas; otros se pueden cambiar utilizando el mandato MQSC ALTER QMGR.

Los atributos también se pueden visualizar utilizando el mandato DISPLAY QMGR. La mayoría de los atributos de gestor de colas se pueden consultar abriendo un objeto MQOT_Q_MGR especial y utilizando la llamada MQINQ con el descriptor de contexto devuelto.

En la tabla siguiente se resumen los atributos que son específicos del gestor de colas. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos que se muestran en esta sección son nombres descriptivos utilizados con la llamada MQINQ; los nombres son los mismos que para los mandatos PCF. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte Mandatos de script (MQSC) para obtener más información.

Tabla 572. Atributos para el gestor de colas.

Lista de atributos de gestor de colas con enlaces y descripción breve

Atributo	Descripción
AccountingConnOverride	Alterar temporalmente los valores de contabilidad.
AccountingInterval	Frecuencia con la que se escriben registros de contabilidad intermedios.

Tabla 572. Atributos para el gestor de colas.

Lista de atributos de gestor de colas con enlaces y descripción breve

(continuación)

Atributo	**Descripción**
ActivityConnOverride	Alterar temporalmente los valores de actividad.
ActivityTrace	Controla la recopilación del rastreo de actividad de la aplicación MQI de WebSphere MQ .
AdoptNewMCACheck	Elementos seleccionados para determinar si se debe adoptar un nuevo MCA.
AdoptNewMCAType	Indica si se debe reiniciar automáticamente una instancia huérfana de un MCA de un tipo de canal determinado.
AlterationDate	Fecha en que se modificó por última vez la definición
AlterationTime	Hora a la que se modificó por última vez la definición
AuthorityEvent	Controla si se generan sucesos de autorización (no autorizados)
BridgeEvent	Atributo de control para sucesos de puente.
ChannelAutoDef	Controla si se permite la definición de canal automática
ChannelAutoDefEvent	Controla si se generan sucesos de definición automática de canal
ChannelAutoDefExit	Nombre de la salida de usuario para la definición automática de canal
ChannelEvent	Atributo de control para sucesos de canal.
ChannelInitiatorControl	Atributo de control para el iniciador de canal
ChannelMonitoring	Datos de supervisión en línea para canales
ChannelStatistics	Controla la recopilación de datos estadísticos para canales.
ChinitAdapters	Número de subtareas de adaptador para procesar llamadas de WebSphere MQ .
ChinitDispatchers	Número de asignadores a utilizar para el iniciador de canal.
	Reservado para uso de IBM .
ChinitTraceAutoStart	Indica si el rastreo de iniciador de canal debe iniciarse automáticamente.
ChinitTraceTableSize	Tamaño del espacio de datos de rastreo del iniciador de canal.
ClusterSenderMonitoringDefault	Datos de supervisión en línea predeterminados para canales emisores de clúster
Estadísticas deClusterSender	Controla la recopilación de información de supervisión de estadísticas para canales emisores de clúster.
ClusterWorkloadData	Datos de usuario para salida de carga de trabajo de clúster
ClusterWorkloadExit	Nombre de salida de usuario para la gestión de carga de trabajo de clúster
ClusterWorkloadLength	Longitud máxima de datos de mensaje pasados a salida de carga de trabajo de clúster
CLWLMRUChannels	Número de canales utilizados más recientemente para el equilibrio de carga de trabajo de clúster
CLWLUseQ	La carga de trabajo de clúster utiliza la cola remota.
CodedCharSetId	Identificador de juego de caracteres codificado
CommandEvent	Atributo de control para sucesos de mandato.
Atributo QName de CommandInput	Nombre de cola de entrada de mandatos
CommandLevel	Nivel de mandato
Atributo Control deCommandServer	Atributo de control para el servidor de mandatos.
Atributo Suceso de configuración	Atributo de control para sucesos de configuración.
DeadLetterQName	Nombre de la cola de mensajes no entregados
DEFCLXQ	Tipo de cola de transmisión de clúster predeterminado
DefXmitQName	Nombre de cola de transmisión predeterminado
DistLists	Soporte de lista de distribución
DNSGroup	Nombre del grupo para el escucha TCP cuando se utiliza el soporte de Workload Manager Dynamic Domain Name Services.
DNSWLM	Indica si el escucha TCP se registra en el Gestor de carga de trabajo para Dynamic Domain Name Services.
ExpiryInterval	Intervalo entre exploraciones de mensajes caducados
IGQPutAuthority	Autorización de transferencia a colas dentro del grupo

Tabla 572. Atributos para el gestor de colas.

Lista de atributos de gestor de colas con enlaces y descripción breve

(continuación)

Atributo	Descripción
IGQUserId	Identificador de usuario de transferencia a colas dentro del grupo
InhibitEvent	Controla si se generan sucesos de inhibición (inhibir obtención e inhibir colocación)
IPAddressVersion	Versión de la dirección Internet Protocol
IntraGroupQueuing	Soporte de transferencia a colas dentro del grupo
ListenerTimer	Intervalo de tiempo entre los intentos de reiniciar el escucha después de una anomalía de APPC o TCP/IP.
LocalEvent	Controla si se generan sucesos de error locales
LoggerEvent	Controla si se generan sucesos de registrador
LUGroupName	Nombre de LU genérico para el escucha de LU 6.2 que maneja las transmisiones de entrada para el grupo de compartición de colas.
LUName	Nombre de LU a utilizar para las transmisiones de LU de salida 6.2 .
LU62ARMSuffix	Sufijo de SYS1.PARMLIB miembro APPCPMxx, que nombra LUADD para este iniciador de canal.
LU62Channels	Número máximo de canales actuales o clientes conectados que utilizan LU 6.2.
MaxActiveChannels	Número máximo de canales que pueden estar activos en cualquier momento.
MaxChannels	Número máximo de canales actuales.
MaxHandles	Número máximo de descriptores de contexto
MaxMsgLength	Longitud máxima de mensaje en bytes
MaxPriority, atributo	Prioridad máxima
MaxPropertiesLength	Longitud máxima de datos de propiedad en bytes
MaxUncommittedMsgs	Número máximo de mensajes no confirmados dentro de una unidad de trabajo
MQIAccounting	Controla la recopilación de información de contabilidad para datos MQI.
MQIStatistics	Controla la recopilación de información de supervisión de estadísticas para el gestor de colas.
MsgMarkBrowseInterval	Intervalo tras el cual el gestor de colas puede eliminar la marca de los mensajes examinados.
OutboundPortMin	Con *OutboundPortMin*, define el rango de números de puerto a utilizar al enlazar canales de salida.
OutboundPortMin	Con *OutboundPortMax*, define el rango de números de puerto a utilizar al enlazar canales de salida.
PerformanceEvent	Controla si se generan sucesos relacionados con el rendimiento
Plataforma	Plataforma en la que se ejecuta el gestor de colas
PubSubNPInputMsg	Si se debe descartar (o conservar) un mensaje de entrada no entregado
PubSubNPResponse	Controla el comportamiento de los no entregados
PubSubMaxMsgRetryCount	Número de reintentos al procesar (bajo punto de sincronismo) un mensaje de mandato fallido
PubSubSyncPoint	Indica si sólo los mensajes persistentes (o todos) deben procesarse bajo punto de sincronismo
PubSubMode	Si la interfaz de publicación/suscripción en cola se está ejecutando
QMgrDesc	Descripción del gestor de colas
QMgrIdentifier	Identificador exclusivo generado internamente del gestor de colas
QMgrName	Nombre del gestor de colas
QSGName	Nombre del grupo de compartición de colas
QueueAccounting	Controla la recopilación de información de contabilidad para colas.
QueueMonitoring	Datos de supervisión en línea para colas
QueueStatistics	Controla la recopilación de datos de estadísticas para colas.
ReceiveTimeout	El tiempo que el canal TCP/IP espera los datos antes de volver al estado inactivo.
ReceiveTimeoutMin	Calificador para *ReceiveTimeout*.
ReceiveTimeoutType	Tiempo mínimo que el canal TCP/IP espera los datos antes de volver al estado inactivo.
RemoteEvent	Controla si se generan sucesos de error remotos

Tabla 572. Atributos para el gestor de colas.

Lista de atributos de gestor de colas con enlaces y descripción breve

(continuación)

Atributo	Descripción
RepositoryName	Nombre del clúster para el que este gestor de colas proporciona servicios de repositorio
RepositoryNamelist	Nombre del objeto de lista de nombres que contiene nombres de clústeres para los que este gestor de colas proporciona servicios de repositorio
ScyCase	Caso de perfiles de seguridad
SharedQMgrNombre	Nombre de gestor de colas de cola compartida
"SPLCAP" en la página 812	WebSphere MQ Advanced Protección de seguridad de mensajes para un gestor de colas activado o desactivado.
SSLCRLNamelist 1	Nombre del objeto de lista de nombres que contiene nombres de objetos de información de autenticación.
SSLCryptoHardware 1	Serie de configuración de hardware criptográfico.
SSLEvent	Atributo de control para sucesos SSL.
SSLFIPSRequired	Utilice sólo algoritmos certificados por FIPS para la criptografía.
SSLKeyRepository 1	Ubicación del repositorio de claves SSL.
Recuento deSSLKeyReset	Recuento de restablecimiento de clave SSL.
SSLTasks 1	Número de subtareas de servidor para procesar llamadas SSL.
StatisticsInterval	Frecuencia con la que se escriben datos de supervisión de estadísticas.
StartStopEvent	Controla si se generan sucesos de inicio y detención
SyncPoint	Disponibilidad de punto de sincronismo
TCPChannels	Número máximo de canales actuales o clientes conectados que utilizan TCP/IP.
TCPKeepAlive	Indica si se debe utilizar TCP KEEPALIVE para comprobar otro extremo de la conexión.
TCPName	Nombre del sistema TCP/IP que está utilizando.
TCPStackType	Cómo puede utilizar el iniciador de canal las direcciones TCP/IP.
Atributo Registro deTraceRoute	Controla el registro de la información de ruta de rastreo.
TriggerInterval	Desencadenante-intervalo de mensaje
Versión	Versión
XrCapability	Especifica si los mandatos de telemetría están soportados.

Notas:

1. Este atributo no se puede consultar utilizando la llamada MQINQ y no se describe en esta sección. Consulte Cambiar gestor de colas para obtener detalles de este atributo.

Tareas relacionadas

Especificación de que sólo se utilizan CipherSpecs certificadas por FIPS en el tiempo de ejecución del cliente MQI

Referencia relacionada

Federal Information Processing Standards (FIPS) para UNIX, Linux y Windows

Alteración temporal de AccountingConn(MQLONG)

Esto permite a las aplicaciones alterar temporalmente el valor de los valores ACCTMQI y ACCTQDATA en el atributo Qmgr.

El valor puede ser uno de los siguientes:

MQMON_INHABILITADO
Las aplicaciones no pueden alterar temporalmente el valor de los atributos ACCTMQI y ACCTQ Qmgr utilizando el campo Opciones de la estructura MQCNO en la llamada MQCONNX. Este es el valor predeterminado.

MQMON_HABILITADO
Las aplicaciones pueden alterar temporalmente los atributos ACCTQ y ACCTMQI Qmgr utilizando el campo Opciones de la estructura MQCNO.

Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas después del cambio en el atributo.

Este atributo sólo está soportado en sistemas IBM i, Unix y Windows.

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_CONN_OVERRIDE con la llamada MQINQ.

AccountingInterval (MQLONG)

Especifica cuánto tiempo debe transcurrir antes de que se graben los registros de contabilidad intermedios (en segundos).

El valor es un entero en el rango de 0 a 604800, con un valor predeterminado de 1800 (30 minutos). Especifique 0 para desactivar los registros intermedios.

Este atributo sólo está soportado en los sistemas IBM i, Windows, UNIXy Linux .

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_INTERVAL con la llamada MQINQ.

Alteración temporal de ActivityConn(MQLONG)

Esto permite a las aplicaciones alterar temporalmente el valor de ACTVTRC en el atributo del gestor de colas.

El valor puede ser uno de los siguientes:

MQMON_INHABILITADO
Las aplicaciones no pueden alterar temporalmente el valor del atributo de gestor de colas ACTVTRC utilizando el campo Opciones de la estructura MQCNO en la llamada MQCONNX. Este es el valor predeterminado.

MQMON_HABILITADO
Las aplicaciones pueden alterar temporalmente el atributo de gestor de colas ACTVTRC utilizando el campo Opciones de la estructura MQCNO.

Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas después del cambio en el atributo.

Este atributo sólo está soportado en sistemas IBM i, Unix y Windows.

Para determinar el valor de este atributo, utilice el selector MQIA_ACTIVITY_CONN_OVERRIDE con la llamada MQINQ .

ActivityTrace (MQLONG)

Esto controla la recopilación del rastreo de actividad de la aplicación MQI de WebSphere MQ .

El valor puede ser uno de los siguientes:

MQMON_ON
Recopile el rastreo de actividad de la aplicación MQI de WebSphere MQ .

MQMON_OFF
No recopile el rastreo de actividad de la aplicación MQI de WebSphere MQ . Este es el valor predeterminado.

Si establece el atributo de gestor de colas ACTVCONO en ENABLED, este valor puede alterarse temporalmente para conexiones individuales utilizando el campo Opciones de la estructura MQCNO.

Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas después del cambio en el atributo.

Este atributo sólo está soportado en sistemas IBM i, Unix y Windows.

Para determinar el valor de este atributo, utilice el selector MQIA_ACTIVITY_TRACE con la llamada MQINQ .

AdoptNewMCACheck (MQLONG)

Define los elementos que se deben comprobar para determinar si se debe adoptar un MCA cuando se detecta un nuevo canal de entrada que tiene el mismo nombre que un MCA que ya está activo

El valor puede ser uno de los siguientes:

MQADOPT_CHECK_Q_MGR_NAME
Compruebe el nombre del gestor de colas.

MQADOPT_CHECK_NET_ADDR
Compruebe la dirección de red.

MQADOPT_CHECK_ALL
Compruebe el nombre del gestor de colas y la dirección de red. Si es posible, realice esta comprobación para evitar que los canales se apaguen, de forma involuntaria o maliciosa. Este es el valor predeterminado.

MQADOPT_CHECK_NONE
No compruebe ningún elemento.

Los cambios en este atributo entrarán en vigor la próxima vez que un canal intente adoptar un canal.

Este atributo sólo está soportado en z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_ADOPTNEWMCA_CHECK con la llamada MQINQ.

AdoptNewMCAType (MQLONG)

Especifica si se debe reiniciar automáticamente una instancia huérfana de un MCA de un tipo de canal determinado cuando se detecta una nueva solicitud de canal de entrada que coincide con el atributo MCACheck AdoptNew

Es uno de los valores siguientes:

MQADOPT_TYPE_NO
La adopción de instancias de canal huérfanas no es necesaria. Este es el valor predeterminado.

MQADOPT_TYPE_ALL
Adopte todos los tipos de canal.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_ADOPTNEWMCA_TYPE con la llamada MQINQ.

AlterationDate (MQCHAR12)

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es YYYY-MM-DD, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Es la hora en que se modificó por última vez la definición. El formato de la hora es HH.MM.SS.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_TIME_LENGTH.

AuthorityEvent (MQLONG)

Esto controla si se generan sucesos de autorización (no autorizados). Es uno de los valores siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos .

Para determinar el valor de este atributo, utilice el selector MQIA_AUTORITY_EVENT con la llamada MQINQ.

BridgeEvent (MQLONG)
Especifica si se generan sucesos de puente IMS .

El valor puede ser uno de los siguientes:

MQEVR_ENABLED
Genere sucesos de puente IMS , como se indica a continuación:

MQRC_BRIDGE_STARTED
MQRC_BRIDGE_STOPPED

MQEVR_DISABLED
No genere sucesos de puente IMS ; este es el valor predeterminado.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_BRIDGE_EVENT con la llamada MQINQ.

Definición ChannelAuto(MQLONG)

Este atributo controla la definición automática de canales de tipo MQCHT_RECEIVER y MQCHT_SVRCONN. La definición automática de canales MQCHT_CLUSSDR siempre está habilitada. El valor puede ser uno de los siguientes:

MQCHAD_DISABLED
Definición automática de canal inhabilitada.

MQCHAD_ENABLED
Definición automática de canal habilitada.

Este atributo sólo está soportado en AIX, HP-UX, IBM i, Linux, Solaris y Windows.

Para determinar el valor de este atributo, utilice el selector MQIA_CHANNEL_AUTO_DEF con la llamada MQINQ.

ChannelAutoDefEvent (MQLONG)

Esto controla si se generan sucesos de definición automática de canal. Se aplica a canales de tipo MQCHT_RECEIVER, MQCHT_SVRCONN y MQCHT_CLUSSDR. El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos .

Este atributo sólo está soportado en AIX, HP-UX, IBM i, Linux, Solaris y Windows.

Para determinar el valor de este atributo, utilice el selector MQIA_CHANNEL_AUTO_DEF_EVENT con la llamada MQINQ.

ChannelAutoDefExit (MQCHARn)

Es el nombre de la salida de usuario para la definición automática de canal. Si este nombre no está en blanco y *ChannelAutoDef* tiene el valor MQCHAD_ENABLED, se llama a la salida cada vez que el gestor de colas está a punto de crear una definición de canal. Esto se aplica a los canales de tipo MQCHT_RECEIVER, MQCHT_SVRCONN y MQCHT_CLUSSDR. A continuación, la salida puede realizar una de las acciones siguientes:

- Cree la definición de canal sin cambios.
- Modifique los atributos de la definición de canal que se crea.
- Suprimir la creación del canal por completo.

Nota: Tanto la longitud como el valor de este atributo son específicos del entorno. Consulte la introducción a la estructura MQCD en "MQCD-Definición de canal" en la página 1035 para obtener detalles sobre el valor de este atributo en diversos entornos.

Este atributo sólo está soportado en AIX, HP-UX, IBM i, Linux, Solaris, Windowsy z/OS. En z/OS, sólo se aplica a los canales de clúster emisor y de clúster receptor.

Para determinar el valor de este atributo, utilice el selector MQCA_CHANNEL_AUTO_DEF_EXIT con la llamada MQINQ. La longitud de este atributo la proporciona MQ_EXIT_NAME_LENGTH.

ChannelEvent (MQLONG)

Especifica si se generan sucesos de canal.

Es uno de los valores siguientes:

MQEVR_EXCEPCIÓN
Solo genere los siguientes sucesos de canal:

- MQRC_CHANNEL_ACTIVATED
- MQRC_CHANNEL_CONV_ERROR
- MQRC_CHANNEL_NOT_ACTIVATED
- MQRC_CHANNEL_STOPPED con los ReasonQualifierssiguientes:

  MQRQ_CHANNEL_STOPPED_ERROR
  MQRQ_CHANNEL_STOPPED_RETRY
  MQRQ_CHANNEL_STOPPED_DISABLED

  MQRC_CHANNEL_STOPPED_BY_USER

MQEVR_ENABLED
Generar todos los sucesos de canal. Es decir, además de los generados por EXCEPTION, genera los siguientes sucesos de canal:

- MQRC_CHANNEL_STARTED
- MQRC_CHANNEL_STOPPED con el ReasonQualifiersiguiente:

  MQRQ_CHANNEL_STOPPED_OK

MQEVR_DISABLED
No genere sucesos de canal; este es el valor predeterminado.

Para determinar el valor de este atributo, utilice el selector MQIA_CHANNEL_EVENT con la llamada MQINQ.

Control de ChannelInitiator(MQLONG)

Especifica si el iniciador de canal debe iniciarse cuando se inicia el gestor de colas.

Es uno de los valores siguientes:

MQSVC_CONTROL_MANUAL
El iniciador de canal no se debe iniciar automáticamente.

MQSVC_CONTROL_Q_MGR
El iniciador de canal se iniciará automáticamente cuando se inicie el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_CONTROL con la llamada MQINQ.

ChannelMonitoring (MQLONG)
Especifica datos de supervisión en línea para canales.

El valor puede ser uno de los siguientes:

MQMON_NONE
Inhabilite la recopilación de datos para la supervisión de canales para todos los canales independientemente del valor del atributo de canal MONCHL. Este es el valor predeterminado.

MQMON_OFF
Desactive la recopilación de datos de supervisión para los canales que especifiquen QMGR en el atributo de canal MONCHL.

MQMON_LOW
Active la recopilación de datos de supervisión con una proporción baja de recopilación de datos para canales que especifican QMGR en el atributo de canal MONCHL.

MQMON_MEDIO
Active la recopilación de datos de supervisión con una proporción moderada de recopilación de datos para canales que especifican QMGR en el atributo de canal MONCHL.

MQMON_HIGH
Active la recopilación de datos de supervisión con una proporción alta de recopilación de datos para canales que especifican QMGR en el atributo de canal MONCHL.

Para determinar el valor de este atributo, utilice el selector MQIA_MONITORING_CHANNEL con la llamada MQINQ.

ChannelStatistics (MQLONG)

Esto controla la recopilación de datos estadísticos para los canales.

El valor puede ser uno de los siguientes:

MQMON_NONE
Inhabilite la recopilación de datos para las estadísticas de canal para todos los canales independientemente del valor del atributo de canal STATCHL. Este es el valor predeterminado.

MQMON_OFF
Desactive la recopilación de datos de estadísticas para los canales que especifiquen QMGR en el atributo de canal STATCHL.

MQMON_LOW
Active la recopilación de datos de estadísticas con una proporción baja de recopilación de datos para canales que especifican QMGR en el atributo de canal STATCHL.

MQMON_MEDIO
Active la recopilación de datos de estadísticas con una proporción moderada de recopilación de datos para canales que especifican QMGR en el atributo de canal STATCHL.

MQMON_HIGH
Active la recopilación de datos de estadísticas con una proporción alta de recopilación de datos para canales que especifican QMGR en el atributo de canal STATCHL.

Para la mayoría de los sistemas, se recomienda utilizar MEDIUM. Sin embargo, para un canal que procesa un gran volumen de mensajes cada segundo, es posible que desee reducir el nivel de muestreo seleccionando LOW. Además, para un canal que sólo procesa unos pocos mensajes, y para el que la información más actual es importante, es posible que desee seleccionar HIGH.

Este atributo sólo está soportado en sistemas IBM i, UNIX y Windows.

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_CHANNEL con la llamada MQINQ.

ChinitAdapters (MQLONG)

Es el número de subtareas de adaptador que se deben utilizar para procesar llamadas de WebSphere MQ . El valor debe ser de 0 a 9999, con un valor por omisión de 8.

La proporción de adaptadores con respecto a los asignadores (el atributo ChinitDispatchers) debe ser aproximadamente de 8 a 5. Sin embargo, si sólo tiene pocos canales, no tiene que disminuir el valor de este parámetro del valor predeterminado. Puede utilizar los valores siguientes: para un sistema de prueba, 8 (valor predeterminado); para un sistema de producción, 20. Idealmente, debería tener 20 adaptadores, lo que proporciona un mayor paralelismo de las llamadas de WebSphere MQ . Es importante para los mensajes persistentes. Es posible que sea mejor tener menos adaptadores para los mensajes no persistentes.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_ADAPTERS con la llamada MQINQ.

ChinitDispatchers (MQLONG)

Es el número de asignadores que se van a utilizar para el iniciador de canal. El valor debe ser de 0 a 9999, con un valor por omisión de 5.

Como directriz, permita un asignador para 50 canales actuales. Sin embargo, si sólo tiene unos pocos canales, no tiene que disminuir el valor de este atributo del valor predeterminado. Si está utilizando TCP/IP, el mayor número de asignadores que se utilizan para canales TCP/IP es 100, incluso si especifica un valor mayor aquí. Puede utilizar los valores siguientes: sistemas de prueba, 5 (el valor predeterminado); sistemas de producción, 20 (necesita 20 asignadores para manejar hasta 1000 canales activos).

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_DISPATCHERS con la llamada MQINQ.

ChinitTraceAutoStart (MQLONG)

Especifica si se debe iniciar automáticamente el rastreo de iniciador de canal.

El valor puede ser uno de los siguientes:

MQTRAXSTR_SÍ
Iniciar automáticamente el rastreo del iniciador de canal. Este es el valor predeterminado.

MQTRAXSTR_NO
No inicie automáticamente el rastreo del iniciador de canal.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_TRACE_AUTO_START con la llamada MQINQ.

ChinitTraceTableSize (MQLONG)

Es el tamaño del espacio de datos de rastreo del iniciador de canal (en MB).

El valor debe estar en el rango de 0 a 2048, con un valor predeterminado de 2.

Nota: Siempre que utilice espacios de datos z/OS de gran tamaño, asegúrese de que tiene suficiente almacenamiento auxiliar en el sistema para dar soporte a cualquier actividad de paginación z/OS relacionada. También tendrá que aumentar el tamaño de los conjuntos de datos SYS1.DUMP.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_CHINIT_TRACE_TABLE_SIZE con la llamada MQINQ.

ClusterSenderMonitoringDefault (MQLONG)

Especifica el valor que debe sustituirse por el atributo ChannelMonitoring de los canales emisores de clúster definidos automáticamente.

El valor puede ser uno de los siguientes:

MQMON_Q_MGR
La recopilación de datos de supervisión en línea se hereda del valor del atributo *ChannelMonitoring* del gestor de colas. Este es el valor predeterminado.

MQMON_OFF
La supervisión del canal está desactivada

MQMON_LOW
A menos que *ChannelMonitoring* sea MQMON_NONE, la supervisión se activa con una tasa baja de recopilación de datos con un efecto mínimo en el rendimiento del sistema. No es probable que los datos recopilados sean los más actuales.

MQMON_MEDIO
A menos que *ChannelMonitoring* sea MQMON_NONE, la supervisión se activa con una velocidad moderada de recopilación de datos con un efecto limitado en el rendimiento del sistema.

MQMON_HIGH
A menos que *ChannelMonitoring* sea MQMON_NONE, la supervisión se activa con una alta tasa de recopilación de datos con un efecto probable en el rendimiento del sistema. Los datos recopilados son los más actuales disponibles.

Para determinar el valor de este atributo, utilice el selector MQIA_MONITORING_AUTO_CLUSSDR con la llamada MQINQ.

Estadísticas de ClusterSender(MQLONG)

Puesto que los canales emisores de clúster se pueden definir automáticamente a partir de la definición de CLUSRCVR en el repositorio, no puede modificar el valor del atributo STATCHL para estos canales emisores de clúster definidos automáticamente utilizando el canal ALTER. Para estos canales, la decisión de recopilar datos de supervisión en línea se basa en el valor de este atributo de gestor de colas.

El valor puede ser uno de los siguientes:

MQMON_Q_MGR
La recopilación de datos de estadísticas para canales emisores de clúster definidos automáticamente se basa en el valor del atributo de gestor de colas STATCHL. Este es el valor predeterminado.

MQMON_OFF
Desactive la recopilación de datos de estadísticas para los canales emisores de clúster definidos automáticamente.

MQMON_LOW
Active la recopilación de datos de estadísticas para los canales emisores de clúster definidos automáticamente con una proporción baja de recopilación de datos.

MQMON_MEDIO
Active la recopilación de datos de estadísticas para canales emisores de clúster definidos automáticamente con una proporción moderada de recopilación de datos.

MQMON_HIGH
Active la recopilación de datos de estadísticas para los canales emisores de clúster definidos automáticamente con una proporción alta de recopilación de datos.

Para la mayoría de los sistemas recomendamos MEDIO. Sin embargo, para un canal emisor de clúster definido automáticamente que procesa un gran volumen de mensajes cada segundo, es posible que desee reducir el nivel de muestreo seleccionando LOW. Además, para un canal que sólo procesa unos

pocos mensajes, y para el que la información más actual es importante, es posible que desee seleccionar HIGH.

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_AUTO_CLUSSDR con la llamada MQINQ.

Datos de ClusterWorkload(MQCHAR32)

Es una serie de caracteres de 32 bytes definida por el usuario que se pasa a la salida de carga de trabajo del clúster cuando se llama. Si no hay datos para pasar a la salida, la serie está en blanco.

Este atributo sólo está soportado en AIX, HP-UX, IBM i, Linux, Solaris, Windows y z/OS.

Para determinar el valor de este atributo, utilice el selector MQCA_CLUSTER_WORKLOAD_DATA con la llamada MQINQ.

Salida de ClusterWorkload(MQCHARn)

Es el nombre de la salida de usuario para la gestión de carga de trabajo de clúster. Si este nombre no está en blanco, se llama a la salida cada vez que se transfiere un mensaje a una cola de clúster o se mueve de una cola de clúster emisor a otra. A continuación, la salida puede aceptar la instancia de cola seleccionada por el gestor de colas como destino del mensaje o seleccionar otra instancia de cola.

Nota: Tanto la longitud como el valor de este atributo son específicos del entorno.

Este atributo sólo está soportado en AIX, HP-UX, IBM i, Linux, Solaris, Windows y z/OS.

Para determinar el valor de este atributo, utilice el selector MQCA_CLUSTER_WORKLOAD_EXIT con la llamada MQINQ. La longitud de este atributo la proporciona MQ_EXIT_NAME_LENGTH.

Longitud de ClusterWorkload(MQLONG)

Es la longitud máxima de los datos de mensaje que se pasan a la salida de carga de trabajo del clúster. La longitud real de los datos pasados a la salida es el mínimo de lo siguiente:

- Longitud del mensaje.
- El atributo *MaxMsgLength* del gestor de colas.
- El atributo *ClusterWorkloadLength* .

Este atributo sólo está soportado en AIX, HP-UX, IBM i, Linux, Solaris, Windows y z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_CLUSTER_WORKLOAD_LENGTH con la llamada MQINQ.

CLWLMRUChannels (MQLONG)

Especifica el número máximo de canales de clúster utilizados más recientemente, que se deben tener en cuenta para que los utilice el algoritmo de elección de carga de trabajo de clúster.

Es un valor comprendido entre 1 y 999999999.

Para determinar el valor de este atributo, utilice el selector MQIA_CLWL_MRU_CHANNELS con la llamada MQINQ.

CLWLUseQ (MQLONG)

Especifica si se deben utilizar colas remotas para la carga de trabajo del clúster.

El valor puede ser uno de los siguientes:

MQCLWL_USEQ_ANY
Utilice colas locales y remotas.

MQCLWL_USEQ_LOCAL
No utilice colas remotas. Este es el valor predeterminado.

Para determinar el valor de este atributo, utilice el selector MQIA_CLWL_USEQ con la llamada MQINQ.

CodedCharSetId (MQLONG)

Define el juego de caracteres utilizado por el gestor de colas para todos los campos de serie de caracteres definidos en la MQI como, por ejemplo, los nombres de los objetos y la fecha y hora de creación de la cola. El juego de caracteres debe ser uno que tenga caracteres de un solo byte para los caracteres que son válidos en los nombres de objeto. No se aplica a los datos de aplicación transportados en el mensaje. El valor depende del entorno:

- En z/OS, el valor se establece a partir de los parámetros del sistema cuando se inicia el gestor de colas; el valor predeterminado es 500.
- En Windows, el valor es el CODEPAGE primario del usuario que crea el gestor de colas.
- En IBM i, el valor es el que se establece en el entorno cuando se crea por primera vez el gestor de colas.
- En sistemas UNIX , el valor es el valor predeterminado CODESET para el entorno local del usuario que crea el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_CODED_CHAR_SET_ID con la llamada MQINQ.

CommandEvent (MQLONG)

Especifica si se generan sucesos de mandato, como se indica a continuación:

MQEVR_DISABLED
No generar sucesos de mandato. Éste es el valor predeterminado.

MQEVR_ENABLED
Generar sucesos de mandato.

MQEVR_NO_DISPLAY
Los sucesos de mandato se generan para todos los mandatos satisfactorios distintos de MQINQ.

Para determinar el valor de este atributo, utilice el selector MQIA_COMMAND_EVENT con la llamada MQINQ.

CommandInputQName (MQCHAR48)

Es el nombre de la cola de entrada de mandatos definida en el gestor de colas local. Es una cola a la que los usuarios pueden enviar mandatos, si están autorizados a hacerlo. El nombre de la cola depende del entorno:

- En z/OS, el nombre de la cola es SYSTEM.COMMAND.INPUT; se le pueden enviar los mandatos MQSC y PCF. Consulte Mandatos MQSC para obtener detalles de los mandatos MQSC y Definiciones de los formatos de mandatos programables para obtener detalles de los mandatos PCF.
- En todos los demás entornos, el nombre de la cola es SYSTEM.ADMIN.COMMAND.QUEUEy solo se le pueden enviar mandatos PCF. Sin embargo, se puede enviar un mandato MQSC a esta cola si el mandato MQSC está entre un mandato PCF de tipo MQCMD_ESCAPE. Consulte Escape para obtener información sobre el mandato Escape.

Para determinar el valor de este atributo, utilice el selector MQCA_COMMAND_INPUT_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

CommandLevel (MQLONG)

Indica el nivel de mandatos de control del sistema soportados por el gestor de colas. Puede tener uno de los valores siguientes:

MQCMDL_LEVEL_1
Nivel 1 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes de IBM WebSphere MQ:

- MQSeries para AIX Versión 2 Release 2

- MQSeries para
  - Versión 1 Release 1.1
  - Versión 1 Release 1.2
  - Versión 1 Release 1.3
- MQSeries para OS/400
  - Versión 2 Release 3
  - Versión 3 Release 1
  - Versión 3 Release 6
- MQSeries para Windows Versión 2 Release 0

MQCMDL_LEVEL_101
MQSeries para Windows Versión 2 Release 0.1.

MQCMDL_LEVEL_110
MQSeries para Windows Versión 2 Release 1.

MQCMDL_LEVEL_114
MQSeries para Versión 1 Release 1.4.

MQCMDL_LEVEL_120
MQSeries para la Versión 1 Release 2.0.

MQCMDL_LEVEL_200
MQSeries para Windows NT Versión 2 Release 0.

MQCMDL_LEVEL_210
MQSeries para OS/390 Versión 2 Release 1.0.

MQCMDL_LEVEL_220
Nivel 220 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes de IBM WebSphere MQ:

- MQSeries para AT & T GIS UNIX Versión 2 Release 2
- MQSeries para SINIX y DC/OSx Versión 2 Release 2
- MQSeries para SunOS Versión 2 Release 2
- MQSeries para Tándem NonStop Kernel Versión 2 Release 2

MQCMDL_LEVEL_221
Nivel 221 de mandatos de control del sistema.

Este valor lo devuelve MQSeries para AIX Versión 2 Release 2.1

MQCMDL_LEVEL_320
Nivel 320 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes de IBM WebSphere MQ:

- MQSeries para OS/400
  - Versión 3 Release 2
  - Versión 3 Release 7

MQCMDL_LEVEL_420
Nivel 420 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes de IBM WebSphere MQ:

- MQSeries para IBM i
  - Versión 4 Release 2.0
  - Versión 4 Release 2.1

MQCMDL_LEVEL_500
Nivel 500 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes de IBM WebSphere MQ:

- IBM WebSphere MQ for AIX Versión 5 Release 0
- MQSeries para HP-UX Versión 5 Release 0
- MQSeries para Solaris Versión 5 Release 0
- MQSeries para Windows NT Versión 5 Release 0

MQCMDL_LEVEL_510
Nivel 510 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes de IBM WebSphere MQ:

- IBM WebSphere MQ for AIX Versión 5 Release 1
- MQSeries para AS/400 Versión 5 Release 1
- MQSeries para HP-UX Versión 5 Release 1
- IBM WebSphere MQ for HP Integrity NonStop Server Versión 5 Release 3
- MQSeries para Compaq Tru64 UNIX Versión 5 Release 1
- MQSeries para Solaris Versión 5 Release 1
- MQSeries para Windows NT Versión 5 Release 1

MQCMDL_LEVEL_520
Nivel 520 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes de IBM WebSphere MQ:

- MQSeries para AIX ersion 5 Release 2
- MQSeries para AS/400 Versión 5 Release 2
- MQSeries para HP-UX Versión 5 Release 2
- MQSeries para Linux Versión 5 Release 2
- MQSeries para OS/390 Versión 5 Release 2
- MQSeries para Sun Solaris Versión 5 Release 2
- MQSeries para Windows NT Versión 5 Release 2

MQCMDL_LEVEL_530
Nivel 530 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes de IBM WebSphere MQ:

- IBM WebSphere MQ for AIX Versión 5 Release 3
- IBM WebSphere MQ for HP-UX Versión 5 Release 3
- IBM WebSphere MQ for i/Series Versión 5 Release 3
- IBM WebSphere MQ for Linux for Intel Versión 5 Release 3
- IBM WebSphere MQ for Linux for zSeries Versión 5 Release 3
- IBM WebSphere MQ for Solaris Versión 5 Release 3
- IBM WebSphere MQ for Windows Versión 5 Release 3
- IBM WebSphere MQ for z/OS Versión 5 Release 3

MQCMDL_LEVEL_600
Nivel 600 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes de IBM WebSphere MQ:

- IBM WebSphere MQ for AIX Version 6.0
- IBM WebSphere MQ for HP-UX Version 6.0

- IBM WebSphere MQ para i/Series Version 6.0
- IBM WebSphere MQ for Linux Version 6.0
- IBM WebSphere MQ for Solaris Version 6.0
- IBM WebSphere MQ for Windows Version 6.0
- IBM WebSphere MQ for z/OS Version 6.0

MQCMDL_LEVEL_700
Nivel 700 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes de IBM WebSphere MQ:

- IBM WebSphere MQ for AIX Version 7.0
- IBM WebSphere MQ for HP-UX Version 7.0
- IBM WebSphere MQ for IBM i Version 7.0
- IBM WebSphere MQ for Linux Version 7.0
- IBM WebSphere MQ for Solaris Version 7.0
- IBM WebSphere MQ for Windows Version 7.0
- IBM WebSphere MQ for z/OS Version 7.0

MQCMDL_LEVEL_701
Nivel 701 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes de IBM WebSphere MQ:

- IBM WebSphere MQ for AIX Version 7.0.1
- IBM WebSphere MQ for HP-UX Version 7.0.1
- IBM WebSphere MQ for IBM i Version 7.0.1
- IBM WebSphere MQ for Linux Version 7.0.1
- IBM WebSphere MQ for Solaris Version 7.0.1
- IBM WebSphere MQ for Windows Version 7.0.1
- IBM WebSphere MQ for z/OS Version 7.0.1

MQCMDL_LEVEL_710
Nivel 710 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes de IBM WebSphere MQ:

- IBM WebSphere MQ for AIX Version 7.1
- IBM WebSphere MQ for HP-UX Version 7.1
- IBM WebSphere MQ for IBM i Version 7.1
- IBM WebSphere MQ for Linux Version 7.1
- IBM WebSphere MQ for Solaris Version 7.1
- IBM WebSphere MQ for Windows Version 7.1
- IBM WebSphere MQ for z/OS Version 7.1

MQCMDL_LEVEL_750
Nivel 750 de mandatos de control del sistema.

Este valor lo devuelven las versiones siguientes de IBM WebSphere MQ:

- IBM WebSphere MQ for AIX Version 7.5
- IBM WebSphere MQ for HP-UX Version 7.5
- IBM WebSphere MQ for IBM i Version 7.5
- IBM WebSphere MQ for Linux Version 7.5
- IBM WebSphere MQ for Solaris Version 7.5

- IBM WebSphere MQ for Windows Version 7.5

El conjunto de mandatos de control del sistema que corresponde a un valor determinado del atributo *CommandLevel* varía según el valor del atributo *Platform* ; ambos deben utilizarse para decidir qué mandatos de control del sistema están soportados.

Para determinar el valor de este atributo, utilice el selector MQIA_COMMAND_LEVEL con la llamada MQINQ .

CommandServerControl (MQLONG)

Especifica si el servidor de mandatos debe iniciarse cuando se inicia el gestor de colas.

El valor puede ser:

MQSVC_CONTROL_MANUAL
El servidor de mandatos no se debe iniciar automáticamente.

MQSVC_CONTROL_Q_MGR
El servidor de mandatos se iniciará automáticamente cuando se inicie el gestor de colas.

Este atributo no está soportado en z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_CMD_SERVER_CONTROL con la llamada MQINQ.

ConfigurationEvent (MQLONG)

Controla si se generan sucesos de configuración.

Para determinar el valor de este atributo, utilice el selector MQIA_CONFIGURATION_EVENT con la llamada MQINQ.

El valor puede ser:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

DeadLetterQName (MQCHAR48)

Es el nombre de una cola definida en el gestor de colas local como la cola de mensajes no entregados (undelivered-message). Los mensajes se envían a esta cola si no pueden direccionarse a su destino correcto.

Por ejemplo, los mensajes se colocan en esta cola cuando:

- Llega un mensaje a un gestor de colas, destinado a una cola que todavía no está definida en ese gestor de colas
- Un mensaje llega a un gestor de colas, pero la cola a la que está destinado no puede recibirlo porque, posiblemente:
  - La cola está llena
  - Las solicitudes de colocación están inhibidas
  - El nodo emisor no tiene autorización para colocar mensajes en la cola

Las aplicaciones también pueden colocar mensajes en la cola de mensajes no entregados.

Los mensajes de informe se tratan de la misma forma que los mensajes ordinarios; si el mensaje de informe no se puede entregar a su cola de destino (normalmente la cola especificada por el campo *ReplyToQ* en el descriptor de mensaje del mensaje original), el mensaje de informe se coloca en la cola de mensajes no entregados (mensaje no entregado).

Nota: Los mensajes que han pasado su tiempo de caducidad (consulte MQMD-Campo de caducidad) **no** se transfieren a esta cola cuando se descartan. Sin embargo, todavía se genera un mensaje de informe de caducidad (MQRO_EXPIRATION) y se envía a la cola *ReplyToQ* , si lo solicita la aplicación emisora.

Los mensajes no se colocan en la cola de mensajes no entregados cuando la aplicación que ha emitido la solicitud de colocación ha recibido una notificación síncrona del problema mediante el código de razón devuelto por la llamada MQPUT o MQPUT1 (por ejemplo, un mensaje colocado en una cola local para el que se inhiben las solicitudes de colocación).

Los mensajes de la cola de mensajes no entregados a veces tienen sus datos de mensaje de aplicación con el prefijo de una estructura MQDLH. Esta estructura contiene información adicional que indica por qué el mensaje se ha colocado en la cola de mensajes no entregados (undelivered-message). Consulte "MQDLH-Cabecera de mensaje no entregado" en la página 326 para obtener más detalles de esta estructura.

Esta cola debe ser una cola local, con un atributo *Usage* de MQUS_NORMAL.

Si un gestor de colas no da soporte a una cola de mensajes no entregados (undelivered-message), o no se ha definido una, el nombre estará en blanco. Todos los gestores de colas de WebSphere MQ dan soporte a una cola de mensajes no entregados (undelivered-message), pero de forma predeterminada no está definida.

Si la cola de mensajes no entregados (undelivered-message) no está definida, llena o inutilizable por alguna otra razón, un mensaje que un agente de canal de mensajes le habría transferido se conserva en la cola de transmisión.

Para determinar el valor de este atributo, utilice el selector MQCA_DEAD_LETTER_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

DefClusterXmitQueueTipo (MQLONG)

El atributo `DefClusterXmitQueueTipo` controla qué cola de transmisión seleccionan de forma predeterminada los canales de clúster emisor para obtener mensajes, para enviar los mensajes a los canales de clúster receptor.

Los valores de `DefClusterXmitQueueType` son `MQCLXQ_SCTQ` o `MQCLXQ_CHANNEL`.

`MQCLXQ_SCTQ`

Todos los canales de clúster emisor envían mensajes de `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. El `correlID` de los mensajes colocados en la cola de transmisión identifica el canal de clúster emisor al que va destinado el mensaje.

SCTQ se establece cuando se define un gestor de colas. Este comportamiento está implícito en las versiones de IBM WebSphere MQ, anteriores a Version 7.5. En versiones anteriores, el atributo de gestor de colas `DefClusterXmitQueueType` no estaba presente.

`MQCLXQ_CHANNEL`

Cada canal de clúster emisor envía mensajes desde una cola de transmisión diferente. Cada cola de transmisión se crea como una cola dinámica permanente de la cola modelo `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE`.

El atributo no está soportado en z/OS.

Si el atributo de gestor de colas, `DefClusterXmitQueueTipo`, se establece en `CHANNEL`, la configuración predeterminada se cambia a los canales de clúster emisor asociados a colas de transmisión de clúster individuales. Las colas de transmisión son colas dinámicas permanentes creadas a partir de la cola modelo `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE`. Cada cola de transmisión está asociada a un canal de clúster emisor. Cuando un canal de clúster emisor presta servicio a una cola de transmisión de clúster, la cola de transmisión contiene mensajes únicamente para un gestor de colas de un clúster. Puede configurar clústeres de modo que cada gestor de colas de un clúster sólo contenga una cola de clúster. En este caso, el tráfico de mensajes de un gestor de colas a cada cola de clúster se transfiere por separado de los mensajes a otras colas.

Para consultar el valor, llame a `MQINQ`o envíe un mandato Consultar gestor de colas (`MQCMD_INQUIRE_Q_MGR`) PCF, estableciendo el selector `MQIA_DEF_CLUSTER_XMIT_Q_TYPE` . Para

cambiar el valor, envíe un mandato PCF Cambiar gestor de colas (MQCMD_CHANGE_Q_MGR), estableciendo el selector MQIA_DEF_CLUSTER_XMIT_Q_TYPE .

Referencia relacionada

Cambiar gestor de colas

Consultar gestor de colas

"MQINQ-Consultar atributos de objeto" en la página 686

La llamada MQINQ devuelve una matriz de enteros y un conjunto de series de caracteres que contienen los atributos de un objeto.

DefXmitQName (MQCHAR48)

Es el nombre de la cola de transmisión que se utiliza para la transmisión de mensajes a gestores de colas remotos, si no hay ninguna otra indicación de qué cola de transmisión utilizar.

Si no hay ninguna cola de transmisión predeterminada, el nombre está totalmente en blanco. El valor inicial de este atributo está en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_DEF_XMIT_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

DistLists (MQLONG)

Esto indica si el gestor de colas local da soporte a listas de distribución en las llamadas MQPUT y MQPUT1 . Es uno de los valores siguientes:

MQDL_SUPPORTED
Listas de distribución soportadas.

MQDL_NOT_SUPPORTED
Listas de distribución no soportadas.

Para determinar el valor de este atributo, utilice el selector MQIA_DIST_LISTS con la llamada MQINQ.

DNSGroup (MQCHAR18)

Es el nombre del grupo para el escucha TCP que maneja las transmisiones de entrada para que se una el grupo de compartición de colas cuando se utiliza el soporte de servicios de nombres de dominio dinámicos del gestor de carga de trabajo. La longitud máxima es 18 caracteres. Si deja este nombre en blanco, se utiliza el nombre del grupo de compartición de colas.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQCA_DNS_GROUP con la llamada MQINQ. La longitud de este atributo la proporciona MQ_DNS_GROUP_NAME_LENGTH.

DNSWLM (MQLONG)

Especifica si el escucha TCP que maneja las transmisiones de entrada para el grupo de compartición de colas se registra con Workload Manager para Dynamic Domain Name Services

El valor puede ser uno de los siguientes:

MQDNSWLM_SÍ
El escucha se registra con el Gestor de carga de trabajo.

MQDNSWLM_NO
El escucha no se registra con el Gestor de carga de trabajo. Este es el valor predeterminado.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_DNS_WLM con la llamada MQINQ.

ExpiryInterval (MQLONG)

Esto indica la frecuencia con la que el gestor de colas explora las colas en busca de mensajes caducados. Es un intervalo de tiempo en segundos comprendido entre 1 y 99 999 999, o el siguiente valor especial:

MQEXPI_OFF
El gestor de colas no explora las colas en busca de mensajes caducados.

Para determinar el valor de este atributo, utilice el selector MQIA_EXPIRY_INTERVAL con la llamada MQINQ.

Este atributo sólo está soportado en z/OS.

IGQPutAuthority (MQLONG)

Este atributo sólo se aplica si el gestor de colas local es miembro de un grupo de compartición de colas. Indica el tipo de comprobación de autorización que se realiza cuando el agente de transferencia a colas dentro del grupo local (agente IGQ) elimina un mensaje de la cola de transmisión compartida y coloca el mensaje en una cola local. El valor puede ser uno de los siguientes:

MQIGQPA_PREDETERMINADO
El identificador de usuario que se comprueba para la autorización es el valor del campo *UserIdentifier* en el MQMD *separado* que está asociado con el mensaje cuando el mensaje está en la cola de transmisión compartida. Es el identificador de usuario del programa que ha colocado el mensaje en la cola de transmisión compartida, y normalmente es el mismo que el identificador de usuario bajo el que se ejecuta el gestor de colas remoto.

Si el perfil RESLEVEL indica que se debe comprobar más de un identificador de usuario, también se comprueba el identificador de usuario del agente de IGQ local (*IGQUserId*).

CONTEXTO_MQIGQPA
El identificador de usuario que se comprueba para la autorización es el valor del campo *UserIdentifier* en el MQMD *separado* que está asociado con el mensaje cuando el mensaje está en la cola de transmisión compartida. Es el identificador de usuario del programa que ha colocado el mensaje en la cola de transmisión compartida, y normalmente es el mismo que el identificador de usuario bajo el que se ejecuta el gestor de colas remoto.

Si el perfil RESLEVEL indica que se debe comprobar más de un identificador de usuario, también se comprueban el identificador de usuario del agente de IGQ local (*IGQUserId*) y el valor del campo *UserIdentifier* en el MQMD *incorporado* . Este último identificador de usuario suele ser el identificador de usuario de la aplicación que ha originado el mensaje.

MQIGQPA_ONLY_IGQ
El identificador de usuario que se comprueba para la autorización es el identificador de usuario del agente de IGQ local (*IGQUserId*).

Si el perfil RESLEVEL indica que se debe comprobar más de un identificador de usuario, este identificador de usuario se utiliza para todas las comprobaciones.

MQIGQPA_ALTERNATE_OR_IGQ
El identificador de usuario que se comprueba para la autorización es el identificador de usuario del agente de IGQ local (*IGQUserId*).

Si el perfil RESLEVEL indica que se debe comprobar más de un identificador de usuario, también se comprueba el valor del campo *UserIdentifier* en el MQMD *incorporado* . Este identificador de usuario suele ser el identificador de usuario de la aplicación que ha originado el mensaje.

Para determinar el valor de este atributo, utilice el selector MQIA_IGQ_PUT_AUTHORITY con la llamada MQINQ.

Este atributo sólo está soportado en z/OS.

IGQUserId (MQLONG)

Este atributo sólo se aplica si el gestor de colas local es miembro de un grupo de compartimiento de colas. Especifica el identificador de usuario que está asociado con el agente de transferencia a colas dentro del grupo local (agente IGQ). Este identificador es uno de los identificadores de usuario que se

pueden comprobar para la autorización cuando el agente de IGQ coloca mensajes en colas locales. Los identificadores de usuario reales que se comprueban dependen del valor del atributo *IGQPutAuthority* y de las opciones de seguridad externas.

Si *IGQUserId* está en blanco, no se asocia ningún identificador de usuario con el agente de IGQ y no se realiza la comprobación de autorización correspondiente (aunque es posible que se sigan comprobando otros identificadores de usuario para la autorización).

Para determinar el valor de este atributo, utilice el selector MQCA_IGQ_USER_ID con la llamada MQINQ. La longitud de este atributo la proporciona MQ_USER_ID_LENGTH.

Este atributo sólo está soportado en z/OS.

InhibitEvent (MQLONG)

Esto controla si se generan sucesos de inhibición (inhibir obtención e inhibir colocación). El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos .

Para determinar el valor de este atributo, utilice el selector MQIA_INHIBID_EVENT con la llamada MQINQ.

En z/OS, no puede utilizar la llamada MQINQ para determinar el valor de este atributo.

Cola IntraGroup(MQLONG)

Este atributo sólo se aplica si el gestor de colas local es miembro de un grupo de compartición de colas. Indica si la transferencia a colas dentro del grupo está habilitada para el grupo de compartición de colas. El valor puede ser uno de los siguientes:

MQIGQ_INHABILITADO
Todos los mensajes destinados a otros gestores de colas del grupo de compartición de colas se transmiten utilizando canales convencionales.

MQIGQ_HABILITADO
Los mensajes destinados a otros gestores de colas del grupo de compartición de colas se transmiten utilizando la cola de transmisión compartida si se cumple la condición siguiente:

- La longitud de los datos del mensaje más la cabecera de transmisión no supera los 63 KB (64 512 bytes).

  Se recomienda asignar algo más de espacio que el tamaño de MQXQH para la cabecera de transmisión; para ello se proporciona la constante MQ_MSG_HEADER_LENGTH.

Si esta condición no se satisface, el mensaje se transmite utilizando canales convencionales.

Nota: Cuando se habilita la transferencia a colas dentro del grupo, el orden de los mensajes transmitidos utilizando la cola de transmisión compartida no se conserva en relación con los transmitidos utilizando canales convencionales.

Para determinar el valor de este atributo, utilice el selector MQIA_INTRA_GROUP_QUEUING con la llamada MQINQ.

Este atributo sólo está soportado en z/OS.

IPAddressVersion (MQLONG)

Especifica qué versión de dirección IP, IPv4 o IPv6, se utiliza.

Este atributo sólo es relevante para los sistemas que ejecutan IPv4 e IPv6 y sólo afecta a los canales definidos como que tienen un *TransportType* de MQXPY_TCP cuando se cumple una de las condiciones siguientes:

- El *ConnectionName* del canal es un nombre de host que se resuelve en una dirección IPv4 y IPv6 y no se especifica su parámetro *LocalAddress* .
- Los *ConnectionName* y *LocalAddress* del canal son ambos nombres de host que se resuelven en las direcciones IPv4 y IPv6 .

El valor puede ser:

MQIPADDR_IPV4
Se utiliza IPv4 .

MQIPADDR_IPV6
Se utiliza IPv6 .

Para determinar el valor de este atributo, utilice el selector MQIA_IP_ADDRESS_VERSION con la llamada MQINQ.

ListenerTimer (MQLONG)

Este es el intervalo de tiempo (en segundos) entre los intentos de WebSphere MQ de reiniciar el escucha si se ha producido un error de APPC o TCP/IP. El valor debe estar entre 5 y 9999, con un valor predeterminado de 60.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_LISTENER_TIMER con la llamada MQINQ.

LocalEvent (MQLONG)

Esto controla si se generan sucesos de error locales. El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos .

Para determinar el valor de este atributo, utilice el selector MQIA_LOCAL_EVENT con la llamada MQINQ.

En z/OS, no puede utilizar la llamada MQINQ para determinar el valor de este atributo.

LoggerEvent (MQLONG)

Esto controla si se generan sucesos de registro de recuperación. El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos .

Para determinar el valor de este atributo, utilice el selector MQIA_LOGGER_EVENT con la llamada MQINQ.

Este atributo sólo está soportado en AIX, HP-UX, IBM i, Linux, Solaris y Windows.

LUGroupName (MQCHAR8)

Es el nombre de LU genérico para el escucha de LU 6.2 que maneja las transmisiones de entrada para el grupo de compartición de colas. Si deja este nombre en blanco, no puede utilizar este escucha.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQCA_LU_GROUP_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_LU_NAME_LENGTH.

LUName (MQCHAR8)

Es el nombre de la LU que se va a utilizar para las transmisiones de LU de salida 6.2 . Establézcalo en la misma LU que utiliza el escucha para las transmisiones de entrada. Si deja este nombre en blanco, se utiliza la LU predeterminada APPC/MVS; es variable, por lo tanto, establezca siempre LUName si utiliza LU6.2.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQCA_LU_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_LU_NAME_LENGTH.

LU62ARMSuffix (MQCHAR2)

Es el sufijo de SYS1.PARMLIB APPCPMxx, que nomina el LUADD para este iniciador de canal. El mandato z/OS SET APPC=xx se emite cuando ARM reinicia el iniciador de canal. Si deja este nombre en blanco, no se emite SET APPC=xx.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQCA_LU62_ARM_SUFFIX con la llamada MQINQ. La longitud de este atributo la proporciona MQ_ARM_SUFFIX_LENGTH.

LU62Channels (MQLONG)

Es el número máximo de canales que pueden ser actuales, o clientes que se pueden conectar, que utilizan el protocolo de transmisión LU 6.2 .

El valor debe estar en el rango de 0 a 9999, con un valor predeterminado de 200. Si lo establece en cero, no se utiliza el protocolo de transmisión LU 6.2 .

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_LU62_CHANNELS con la llamada MQINQ.

Canales MaxActive(MQLONG)

Este atributo es el número máximo de canales que pueden estar *activos* en cualquier momento.

El valor predeterminado es el especificado en el atributo `MaxChannels`. Para z/OS, el valor debe estar en el rango de 1 a 9 999. Para todas las demás plataformas, el valor debe estar en el rango de 1 a 65 535.

Para determinar el valor de este atributo, utilice el selector MQIA_ACTIVE_CHANNELS con la llamada **MQINQ** .

Conceptos relacionados

Estados de un canal

MaxChannels (MQLONG)

Este atributo es el número máximo de canales que pueden ser *actuales* (incluidos los canales de conexión de servidor con clientes conectados).

Para z/OS, el valor debe estar en el rango de 1 a 9 999, con un valor predeterminado de 200. Para todas las demás plataformas, el valor debe estar en el rango de 1 a 65 535, con un valor predeterminado de 100. Un sistema que está ocupado sirviendo conexiones desde la red puede necesitar un número mayor que el valor predeterminado. Determine el valor que es correcto para su entorno, idealmente observando el comportamiento del sistema durante las pruebas.

Para plataformas que no sean z/OS, el valor de MaxChannels se establece en el archivo qm.ini de los gestores de colas respectivos.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_CHANNELS con la llamada **MQINQ** .

Conceptos relacionados

Estados de un canal

MaxHandles (MQLONG)

Es el número máximo de descriptores de contexto abiertos que cualquier tarea puede utilizar simultáneamente. Cada llamada MQOPEN satisfactoria para una sola cola (o para un objeto que no es una cola) utiliza un descriptor de contexto. Ese descriptor de contexto pasa a estar disponible para su reutilización cuando se cierra el objeto. Sin embargo, cuando se abre una lista de distribución, a cada cola de la lista de distribución se le asigna un descriptor de contexto independiente, y de modo que la llamada MQOPEN utiliza tantos descriptores de contexto como colas hay en la lista de distribución. Esto debe tenerse en cuenta al decidir un valor adecuado para *MaxHandles*.

La llamada MQPUT1 realiza una llamada MQOPEN como parte de su proceso; como resultado, MQPUT1 utiliza tantos manejadores como MQOPEN lo haría, pero los manejadores sólo se utilizan durante la propia llamada MQPUT1 .

En z/OS, *tarea* significa una tarea CICS , una tarea MVS o una región dependiente IMS .

El valor está en el rango de 1 a 999.999.999. El valor predeterminado lo determina el entorno:

- En z/OS, el valor predeterminado es 100.
- En todos los demás entornos, el valor predeterminado es 256.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_MANEJAR con la llamada MQINQ.

MaxMsgLongitud (MQLONG)

Es la longitud del mensaje *físico* más largo que el gestor de colas puede manejar. Sin embargo, debido a que el atributo de gestor de colas *MaxMsgLength* se puede establecer independientemente del atributo de cola *MaxMsgLength* , el mensaje físico más largo que se puede colocar en una cola es el menor de estos dos valores.

Si el gestor de colas da soporte a la segmentación, una aplicación puede colocar un mensaje *lógico* que sea más largo que el menor de los dos atributos *MaxMsgLength* , pero sólo si la aplicación especifica el distintivo MQMF_SEGMENTATION_ALLOWED en MQMD. Si se especifica ese distintivo, el límite superior para la longitud de un mensaje lógico es de 999.999.999 bytes, pero normalmente las restricciones de recursos impuestas por el sistema operativo, o por el entorno en el que se ejecuta la aplicación, dan como resultado un límite inferior.

El límite inferior para el atributo *MaxMsgLength* es de 32 KB (32.768 bytes). El límite superior es de 100 MB (104 857 600 bytes).

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_MSG_LENGTH con la llamada MQINQ.

MaxPriority (MQLONG)

Esta es la prioridad de mensaje máxima soportada por el gestor de colas. Las prioridades van de cero (más bajo) a *MaxPriority* (más alto).

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_PRIORITY con la llamada MQINQ.

MaxPropertiesLongitud (MQLONG)

Se utiliza para controlar el tamaño de las propiedades que pueden fluir con un mensaje. Esto incluye tanto el nombre de propiedad en bytes como el tamaño del valor de propiedad también en bytes.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_PROPERTIES_LENGTH con la llamada MQINQ.

Mensajes MaxUncommitted(MQLONG)

Es el número máximo de mensajes no confirmados que pueden existir en una unidad de trabajo. El número de mensajes no confirmados es la suma de los siguientes mensajes desde el inicio de la unidad de trabajo actual:

- Mensajes transferidos por la aplicación con la opción MQPMO_SYNCPOINT
- Mensajes recuperados por la aplicación con la opción MQGMO_SYNCPOINT
- Mensajes de activación y mensajes de informe COA generados por el gestor de colas para mensajes transferidos con la opción MQPMO_SYNCPOINT
- Mensajes de informe COD generados por el gestor de colas para mensajes recuperados con la opción MQGMO_SYNCPOINT

Los siguientes *no* se cuentan como mensajes no confirmados:

- Mensajes colocados o recuperados por la aplicación fuera de una unidad de trabajo
- Mensajes desencadenantes o mensajes de informe COA/COD generados por el gestor de colas como resultado de mensajes colocados o recuperados fuera de una unidad de trabajo
- Mensajes de informe de caducidad generados por el gestor de colas (incluso si la llamada que provoca el mensaje de informe de caducidad ha especificado MQGMO_SYNCPOINT)
- Mensajes de suceso generados por el gestor de colas (incluso si la llamada que provoca el mensaje de suceso ha especificado MQPMO_SYNCPOINT o MQGMO_SYNCPOINT)

Nota:

1. Los mensajes de informe de excepción los genera el agente de canal de mensajes (MCA), o la aplicación, y se tratan de la misma forma que los mensajes ordinarios colocados o recuperados por la aplicación.
2. Cuando se coloca un mensaje o segmento con la opción MQPMO_SYNCPOINT, el número de mensajes no confirmados se incrementa en uno independientemente de cuántos mensajes físicos resulten realmente de la colocación. (Puede producirse más de un mensaje físico si el gestor de colas debe subdividir el mensaje o segmento.)
3. Cuando se coloca una lista de distribución con la opción MQPMO_SYNCPOINT, el número de mensajes no confirmados se incrementa en un *para cada mensaje físico que se genera*. Esto puede ser tan pequeño como uno, o tan grande como el número de destinos en la lista de distribución.

El límite inferior para este atributo es 1; el límite superior es 999 999 999. El valor predeterminado es 10000.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_UNCOMMITTED_MSGS con la llamada MQINQ.

MQIAccounting (MQLONG)

Esto controla la recopilación de información de contabilidad para datos MQI.

El valor puede ser uno de los siguientes:

MQMON_ON
Recopilar datos de contabilidad de API.

MQMON_OFF
No recopile datos de contabilidad de API. Este es el valor predeterminado.

Si establece el atributo de gestor de colas ACCTCONO en ENABLED, este valor puede alterarse temporalmente para conexiones individuales utilizando el campo Opciones de la estructura MQCNO. Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas que se producen después del cambio en el atributo.

Este atributo sólo está soportado en sistemas IBM i, UNIX y Windows.

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_MQI con la llamada MQINQ.

MQIStatistics (MQLONG)

Esto controla la recopilación de información de supervisión de estadísticas para el gestor de colas.

El valor puede ser uno de los siguientes:

MQMON_ON
Recopilar estadísticas de MQI.

MQMON_OFF
No recopile estadísticas MQI. Este es el valor predeterminado.

Este atributo sólo está soportado en sistemas IBM i, UNIX and Linux y Windows.

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_MQI con la llamada MQINQ.

MsgMarkBrowseInterval (MQLONG)

Intervalo de tiempo en milisegundos después del cual el gestor de colas puede eliminar automáticamente la marca de los mensajes de examen.

Se trata de un intervalo de tiempo (en milisegundos) después del cual el gestor de colas puede eliminar automáticamente la marca de los mensajes de examen.

Este atributo describe el intervalo de tiempo durante el cual se espera que los mensajes que se han marcado como examinados por una llamada a MQGET, utilizando la opción de obtención de mensaje MQGMO_MARK_BROWSE_CO_OP, permanezcan marcados como examinados.

El gestor de colas puede desmarcar automáticamente los mensajes examinados que se han marcado como examinados para el conjunto de descriptores de contexto cooperantes cuando se han marcado durante más de este intervalo aproximado.

Esto no afecta al estado de ningún mensaje marcado como examinar, que se ha obtenido mediante una llamada a MQGET, utilizando la opción de obtención de mensaje MQGMO_MARK_BROWSE_HANDLE.

El valor máximo es 999.999.999 y el valor predeterminado es 5000. Un valor especial de -1 para *MsgMarkBrowseInterval* representa un intervalo de tiempo ilimitado.

Atención: Este valor no debe estar por debajo del valor predeterminado de 5000.

Para determinar el valor de este atributo, utilice el selector MQIA_MSG_MARK_BROWSE_INTERVAL con la llamada MQINQ.

OutboundPortMáx (MQLONG)

Es el número de puerto más alto del rango, definido por OutboundPortMin y OutboundPortMax, de números de puerto que se utilizarán para enlazar canales de salida.

El valor es un entero en el rango de 0 a 65535, y debe ser igual o mayor que el valor mínimo de OutboundPort. El valor por omisión es 0.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_OUTBOUND_PORT_MAX con la llamada MQINQ.

OutboundPortmínimo (MQLONG)

Es el número de puerto más bajo del rango, definido por OutboundPortMin y OutboundPortMax, de números de puerto que se van a utilizar para enlazar canales de salida.

El valor es un entero en el rango de 0 a 65535, y debe ser igual o menor que el valor máximo de OutboundPort. El valor por omisión es 0.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_OUTBOUND_PORT_MIN con la llamada MQINQ.

PerformanceEvent (MQLONG)

Esto controla si se generan sucesos relacionados con el rendimiento. Es uno de los valores siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos .

Para determinar el valor de este atributo, utilice el selector MQIA_PERFORMANCE_EVENT con la llamada MQINQ.

Plataforma (MQLONG)

Esto indica el sistema operativo en el que se ejecuta el gestor de colas:

MQPL_AIX
AIX (el mismo valor que MQPL_UNIX).

MQPL_MVS
z/OS (el mismo valor que MQPL_ZOS).

MQPL_NSK
HP Integrity NonStop Server.

MQPL_OS390
z/OS (el mismo valor que MQPL_ZOS).

MQPL_OS400
IBM i.

MQPL_UNIX
Sistemas UNIX .

MQPL_WINDOWS_NT
Sistemas Windows .

MQPL_ZOS
z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_PLATFORM con la llamada MQINQ.

PubSubNPInputMsg (MQLONG)

Indica si se debe descartar o mantener un mensaje de entrada no entregado.

El valor puede ser uno de los siguientes:

MQUNDELIVERED_DISCARD
Los mensajes de entrada no persistentes se pueden desechar si no se pueden procesar.

Este es el valor predeterminado.

MQUNDELIVERED_KEEP
Los mensajes de entrada no persistentes no se desecharán si no se pueden procesar. En esta situación, la interfaz de publicación/suscripción en cola continuará reintentando el proceso a intervalos adecuados y no continúa procesando mensajes posteriores.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_NP_MSG con la llamada MQINQ.

PubSubNPResponse (MQLONG)

Controla el comportamiento de los mensajes de respuesta no entregados.

El valor puede ser uno de los siguientes:

MQUNDELIVERED_NORMAL
Las respuestas no persistentes que no se pueden colocar en la cola de respuestas se colocan en la cola de mensajes no entregados, si no se pueden colocar en la DLQ, se descartan.

MQUNDELIVERED_SAFE
Las respuestas no persistentes que no se pueden colocar en la cola de respuestas se transfieren a la cola de mensajes no entregados. Si la respuesta no se puede establecer y no se puede colocar en la DLQ, la interfaz de publicación/suscripción en cola retrotraerá la operación actual y, a continuación, volverá a intentarlo a intervalos adecuados y no continuará procesando los mensajes posteriores.

MQUNDELIVERED_DISCARD
Las respuestas no persistentes no se colocan en la cola de respuestas se descartan.

Este es el valor predeterminado para los nuevos gestores de colas.

MQUNDELIVERED_KEEP
Las respuestas no persistentes no se colocan en la cola de mensajes no entregados ni se descartan. En su lugar, la interfaz de publicación/suscripción en cola restituirá la operación actual y, a continuación, volverá a intentarlo a intervalos adecuados.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_NP_RESP con la llamada MQINQ.

Valor predeterminado para los gestores de colas migrados.

Si el gestor de colas se ha migrado desde WebSphere MQ V6.0, el valor inicial de este atributo depende de los valores de DiscardNonPersistentResponse y DLQNonPersistentResponse antes de la migración, tal como se muestra en la tabla siguiente.

		Respuesta DLQNonPersistent		
		Sí	**No**	**No establecido**
DiscardNonPersistentResponse	**Sí**	MQUNDELIVERED_NORMAL	MQUNDELIVERED_DISCARD	MQUNDELIVERED_NORMAL
	No	MQUNDELIVERED_SAFE	MQUNDELIVERED_KEEP	MQUNDELIVERED_SAFE
	No establecido	Si SyncPointPersistent = No, MQUNDELIVERED_SAFE de lo contrario MQUNDELIVERED_NORMAL	Si SyncPointPersistent = No, MQUNDELIVERED_KEEP de lo contrario MQUNDELIVERED_DISCARD	Si SyncPointPersistent = No, MQUNDELIVERED_SAFE de lo contrario MQUNDELIVERED_NORMAL

PubSubMaxMsgRetryCount (MQLONG)

Número de reintentos al procesar un mensaje de mandato fallido bajo punto de sincronismo.

El valor puede ser uno de los siguientes:

0 - 999 999 999
El valor predeterminado es 5.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_MAXMSG_RETRY_COUNT con la llamada MQINQ.

PubSubSyncPoint (MQLONG)

Indica si sólo los mensajes persistentes o todos los mensajes se procesan bajo punto de sincronismo.

El valor puede ser uno de los siguientes:

MQSYNCPOINT_IFPER
Esto hace que la interfaz de publicación/suscripción en cola reciba mensajes no persistentes fuera del punto de sincronización. Si el daemon recibe una publicación fuera del punto de sincronismo, el daemon reenvía la publicación a los suscriptores que conoce fuera del punto de sincronismo.

Este es el valor predeterminado.

MQSYNCPOINT_YES
Esto hace que la interfaz de publicación/suscripción en cola reciba todos los mensajes bajo punto de sincronismo.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_SYNC_PT con la llamada MQINQ.

Modalidad PubSub(MQLONG)

Si el motor de publicación/suscripción y la interfaz de publicación/suscripción en cola se están ejecutando, permitiendo por lo tanto que las aplicaciones publiquen/suscriban utilizando la interfaz de programación de aplicaciones y las colas que están siendo supervisadas por la interfaz de publicación/suscripción en cola.

El valor puede ser uno de los siguientes:

MQPSM_COMPAT
El motor de publicación/suscripción está ejecutándose. Por lo tanto, es posible publicar/suscribirse utilizando la interfaz de programación de aplicaciones. La interfaz de publicación/suscripción en cola no se está ejecutando, por lo tanto, no se actúa sobre ningún mensaje que se coloque en las colas supervisadas por la interfaz de publicación/suscripción en cola. Este valor se utiliza para la compatibilidad con WebSphere Message Broker V6 o versiones anteriores utilizando este gestor de colas, porque debe leer las mismas colas de las que normalmente lee la interfaz de publicación/suscripción en cola.

MQPSM_INHABILITADO
El motor de publicación/suscripción y la interfaz de publicación/suscripción en cola no están ejecutándose. Por lo tanto, no es posible publicar/suscribirse utilizando la interfaz de programación de aplicaciones. No se actúa sobre los mensajes de publicación/suscripción que se colocan en las colas supervisadas por la interfaz de publicación/suscripción en cola.

MQPSM_HABILITADO
El motor de publicación/suscripción y la interfaz de publicación/suscripción en cola están ejecutándose. Por lo tanto, es posible publicar/suscribirse utilizando la interfaz de programación de aplicaciones y las colas supervisadas por la interfaz de publicación/suscripción en cola. Este es el valor predeterminado inicial del gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_PUBSUB_MODE con la llamada MQINQ.

QMgrDesc (MQCHAR64)

Utilice este campo para un comentario que describa el gestor de colas. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, este campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas *CodedCharSetId*), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

- En z/OS, el valor predeterminado es el nombre de producto y el número de versión.
- En todos los demás entornos, el valor por omisión son espacios en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_MGR_DESC con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_MGR_DESC_LENGTH.

QMgrIdentifier (MQCHAR48)

Es un nombre exclusivo generado internamente para el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_MGR_IDENTIFIER con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_MGR_IDENTIFIER_LENGTH.

Este atributo está soportado en los entornos siguientes: AIX, HP-UX, z/OS, IBM i, Solaris, Linux, Windows, además de clientes WebSphere MQ conectados a estos sistemas.

QMgrName (MQCHAR48)

Es el nombre del gestor de colas local, es decir, el nombre del gestor de colas al que está conectada la aplicación.

Los primeros 12 caracteres del nombre se utilizan para construir un identificador de mensaje exclusivo (consulte MQMD-campo MsgId). Por lo tanto, los gestores de colas que pueden intercomunicarse deben tener nombres que difieran en los primeros 12 caracteres, para que los identificadores de mensajes sean exclusivos en la red de gestores de colas.

En z/OS, el nombre es el mismo que el nombre del subsistema, que está limitado a 4 caracteres que no están en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_MGR_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_MGR_NAME_LENGTH.

QSGName (MQCHAR4)

Es el nombre del grupo de compartición de colas al que pertenece el gestor de colas local. Si el gestor de colas local no pertenece a un grupo de compartición de colas, el nombre está en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_QSG_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_QSG_NAME_LENGTH.

Este atributo sólo está soportado en z/OS.

QueueAccounting (MQLONG)

Esto controla la recopilación de información de contabilidad para las colas.

El valor puede ser uno de los siguientes:

MQMON_NONE
No recopile datos de contabilidad para colas, independientemente del valor del atributo de contabilidad de cola ACCTQ. Este es el valor predeterminado.

MQMON_OFF
No recopile datos de contabilidad para colas que especifiquen QMGR en el atributo de cola ACCTQ.

MQMON_ON
Recopilar datos de contabilidad para colas que especifican QMGR en el atributo de cola ACCTQ.

Los cambios en este valor sólo son efectivos para las conexiones con el gestor de colas que se producen después del cambio en el atributo.

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_Q con la llamada MQINQ.

QueueMonitoring (MQLONG)

Especifica el valor predeterminado para la supervisión en línea de colas.

Si el atributo de cola *QueueMonitoring* se establece en MQMON_Q_MGR, este atributo especifica el valor que asume el canal. El valor puede ser:

MQMON_OFF
La recopilación de datos de supervisión en línea está desactivada. Este es el valor predeterminado inicial del gestor de colas.

MQMON_NONE
La recopilación de datos de supervisión en línea está desactivada para las colas independientemente del valor de su atributo *QueueMonitoring* .

MQMON_LOW
La recopilación de datos de supervisión en línea está activada, con una proporción baja de recopilación de datos.

MQMON_MEDIO
La recopilación de datos de supervisión en línea está activada, con una proporción moderada de recopilación de datos.

MQMON_HIGH
La recopilación de datos de supervisión en línea está activada, con una proporción alta de recopilación de datos.

Para determinar el valor de este atributo, utilice el selector MQIA_MONITORING_Q con la llamada MQINQ.

QueueStatistics (MQLONG)

Esto controla la recopilación de datos estadísticos para las colas.

Es uno de los valores siguientes:

MQMON_NONE
No recopile estadísticas de cola para colas, independientemente del valor del atributo de cola *QueueStatistics* . Este es el valor predeterminado.

MQMON_OFF
No recopile datos de estadísticas para colas que especifiquen el gestor de colas en el atributo de cola *QueueStatistics* .

MQMON_ON
Recopilar datos estadísticos para las colas que especifican el gestor de colas en el atributo de cola *QueueStatistics* .

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_Q con la llamada MQINQ.

ReceiveTimeout (MQLONG)

Especifica cuánto tiempo espera un canal TCP/IP para recibir datos, incluyendo pulsaciones, de su asociado antes de volver al estado inactivo. Sólo se aplica a los canales de mensajes y no a los canales MQI.

El significado exacto de ReceiveTimeout se modifica por el valor especificado en el tipo ReceiveTimeout. El tipo ReceiveTimeoutse puede establecer en uno de los siguientes:

- MQRCVTIME_EQUAL-este valor es el número en segundos que el canal debe esperar. Especifique un valor en el rango de 0 a 999999.
- MQRCVTIME_ADD-este valor es el número en segundos que se debe añadir al HBINT negociado y determina cuánto tiempo espera un canal. Especifique un valor en el rango de 1 a 999999.
- MQRCVTIME_MULTIPLY-este valor es un multiplicador que se aplica al HBINT negociado. Especifique un valor de 0 o un valor en el rango de 2 a 99.

El valor por omisión es 0.

Establezca el tipo ReceiveTimeouten MQRCVTIME_MULTIPLY o MQRCVTIME_EQUAL, y ReceiveTimeout en 0, para impedir que un canal exceda el tiempo de espera para recibir datos de su socio.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_RECEIVE_TIMEOUT con la llamada MQINQ.

ReceiveTimeoutMín (MQLONG)

Es el tiempo mínimo, en segundos, que un canal TCP/IP espera para recibir datos, incluidos los latidos, de su socio, antes de volver al estado inactivo.

Sólo se aplica a los canales de mensajes, no a los canales MQI. El valor debe estar en el rango de 0 a 999999, con un valor por omisión de 0.

Si utiliza el tipo ReceiveTimeoutpara especificar que el tiempo de espera del canal TCP/IP debe calcularse en relación con el valor negociado de HBINT, y el valor resultante es menor que el valor de este parámetro, este valor se utiliza en su lugar.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_RECEIVE_TIMEOUT_MIN con la llamada MQINQ.

Tipo ReceiveTimeout(MQLONG)

Este es el calificador, aplicado a ReceiveTimeout para definir cuánto tiempo espera un canal TCP/IP para recibir datos, incluidas las pulsaciones, de su socio, antes de volver al estado inactivo. Sólo se aplica a los canales de mensajes, no a los canales MQI.

El valor puede ser uno de los siguientes:

MQRCVTIME_MULTIPLY
ReceiveTimeout es un multiplicador que se aplica al valor de HBINT negociado para determinar cuánto tiempo espera un canal. Este es el valor predeterminado.

MQRCVTIME_ADD
ReceiveTimeout es un valor, en segundos, que se añade al valor de HBINT negociado para determinar cuánto tiempo espera un canal.

MQRCVTIME_EQUAL
ReceiveTimeout es un valor, en segundos, que el canal espera.

Para detener un canal que excede el tiempo de espera para recibir datos de su socio, establezca el tipo ReceiveTimeouten MQRCVTIME_MULTIPLY o MQRCVTIME_EQUAL, y ReceiveTimeout en 0.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_RECEIVE_TIMEOUT_TYPE con la llamada MQINQ.

RemoteEvent (MQLONG)

Esto controla si se generan sucesos de error remotos. Es uno de los valores siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos .

Para determinar el valor de este atributo, utilice el selector MQIA_REMOTE_EVENT con la llamada MQINQ.

RepositoryName (MQCHAR48)

Es el nombre de un clúster para el que este gestor de colas proporciona un servicio de gestor de repositorios. Si el gestor de colas proporciona este servicio para más de un clúster, *RepositoryNamelist* especifica el nombre de un objeto de lista de nombres que identifica los clústeres y *RepositoryName* está en blanco. Al menos uno de *RepositoryName* y *RepositoryNamelist* debe estar en blanco.

Este atributo sólo está soportado en AIX, HP-UX, IBM i, Linux, Solaris, Windowsy z/OS.

Para determinar el valor de este atributo, utilice el selector MQCA_REPOSITORY_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_MGR_NAME_LENGTH.

RepositoryNamelist (MQCHAR48)

Es el nombre de un objeto de lista de nombres que contiene los nombres de los clústeres para los que este gestor de colas proporciona un servicio de gestor de repositorios. Si el gestor de colas proporciona este servicio sólo para un clúster, el objeto de lista de nombres sólo contiene un nombre. De forma alternativa, se puede utilizar *RepositoryName* para especificar el nombre del clúster, en cuyo caso *RepositoryNamelist* está en blanco. Al menos uno de *RepositoryName* y *RepositoryNamelist* debe estar en blanco.

Este atributo sólo está soportado en AIX, HP-UX, IBM i, Linux, Solaris, Windowsy z/OS.

Para determinar el valor de este atributo, utilice el selector MQCA_REPOSITORY_NAMELIST con la llamada MQINQ. La longitud de este atributo la proporciona MQ_NAMELIST_NAME_LENGTH.

ScyCase(MQCHAR8)

Especifica si el gestor de colas da soporte a nombres de perfil de seguridad en mayúsculas y minúsculas, o sólo en mayúsculas.

El valor puede ser uno de los siguientes:

MQSCYC_UPPER
Los nombres de perfil de seguridad deben estar en mayúsculas.

MQSCYC_MIXED
Los nombres de perfil de seguridad pueden estar en mayúsculas o en mayúsculas y minúsculas.

Los cambios en este atributo entran en vigor cuando se ejecuta un mandato Renovar seguridad con *SecurityType(MQSECTYPE_CLASSES)* especificado.

Este atributo sólo está soportado en z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_SECURITY_CASE con la llamada MQINQ.

SharedQMgrNombre (MQLONG)

Especifica si el *ObjectQmgrName* debe utilizarse o tratarse como el gestor de colas local en una llamada MQOPEN, para una cola compartida, cuando el *ObjectQmgrName* es el de otro gestor de colas del grupo de compartición de colas.

El valor puede ser:

MQSQQM_USE
Se utiliza *ObjectQmgrName* y se abre la cola de transmisión adecuada.

MQSQQM_IGNORE
Si la cola de destino es compartida y el *ObjectQmgrName* es el de un gestor de colas en el mismo grupo de compartición de colas, la apertura se realiza localmente.

Este atributo sólo es válido en z/OS.

Para determinar el valor de este atributo, utilice el selector MQIA_SHARED_Q_Q_MGR_NAME con la llamada MQINQ.

SPLCAP

Indica si las prestaciones de seguridad de WebSphere MQ Advanced Message Security están disponibles para un gestor de colas.

MQCAP_SUPPORTED
Este es el valor predeterminado si el componente AMS de WebSphere MQ está instalado para la instalación bajo la que se ejecuta el gestor de colas.

MQCAP_NO_SOPORTADO

SSLEvent (MQLONG)

Especifica si se generan sucesos SSL.

Es uno de los valores siguientes:

MQEVR_ENABLED
Genere sucesos SSL, como se indica a continuación:

MQRC_CHANNEL_SSL_ERROR

MQEVR_DISABLED
No generar sucesos SSL; este es el valor predeterminado.

Para determinar el valor de este atributo, utilice el selector MQIA_SSL_EVENT con la llamada MQINQ.

SSLFIPSRequired (MQLONG)

Esto le permite especificar que sólo se utilizarán algoritmos certificados por FIPS si la criptografía se ejecuta en WebSphere MQ, en lugar de en hardware de cifrado. Si el hardware de cifrado está configurado, los módulos de cifrado utilizados son los módulos proporcionados por el producto de hardware; estos módulos pueden o no estar certificados por FIPS a un nivel determinado en función del producto de hardware en uso.

El valor es uno de los valores siguientes:

MQSSL_FIPS_NO
Utilice cualquier CipherSpec soportada en la plataforma en uso. Este es el valor predeterminado.

MQSSL_FIPS_YES
Utilice sólo algoritmos criptográficos certificados por FIPS en las CipherSpecs permitidas en todas las conexiones SSL desde y a este gestor de colas.

Este parámetro sólo es válido en plataformas UNIX, Linux, Windowsy z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_SSL_FIPS_REQUIRED con la llamada MQINQ.

Tareas relacionadas

Especificación de que sólo se utilizan CipherSpecs certificadas por FIPS en el tiempo de ejecución del cliente MQI

Referencia relacionada

Federal Information Processing Standards (FIPS) para UNIX, Linux y Windows

Recuento de SSLKeyReset(MQLONG)

Especifica cuándo los agentes de canal de mensajes (MCA) de canal SSL que inician la comunicación restablecen la clave secreta utilizada para el cifrado en el canal.

El valor representa el número total de bytes no cifrados que se envían y se reciben a través del canal antes de renegociar la clave secreta. El número de bytes incluye la información de control que envía el MCA.

El valor es un número comprendido entre 0 y 999 999 999, con un valor por omisión de 0. Si especifica un recuento de restablecimiento de clave secreta SSL/TLS en el rango de 1 byte a 32 KB, los canales SSL/TLS utilizarán un recuento de restablecimiento de clave secreta de 32 KB. Esto es para evitar el coste de proceso de restablecimientos de clave excesivos que se producirían para valores de restablecimiento de clave secreta SSL/TLS pequeños.

La clave secreta se renegocia cuando el número total de bytes no cifrados enviados y recibidos por el MCA de canal iniciador excede el valor especificado, o si las pulsaciones de canal están habilitadas antes de que se envíen o reciban los datos después de una pulsación de canal, lo que ocurra primero.

El recuento de bytes enviados y recibidos para renegociación incluye la información de control enviada y recibida por el canal MCA y se restablece cuando se produce una renegociación.

Utilice un valor de 0 para indicar que las claves secretas nunca se renegocian.

Para determinar el valor de este atributo, utilice el selector MQIA_SSL_RESET_COUNT con la llamada MQINQ.

Suceso StartStop(MQLONG)

Esto controla si se generan sucesos de inicio y detención. El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos .

Para determinar el valor de este atributo, utilice el selector MQIA_START_STOP_EVENT con la llamada MQINQ.

StatisticsInterval (MQLONG)

Especifica la frecuencia (en segundos) con la que se graban los datos de supervisión de estadísticas en la cola de supervisión.

El valor es un entero en el rango de 0 a 604800, con un valor predeterminado de 1800 (30 minutos).

Para determinar el valor de este atributo, utilice el selector MQIA_STATISTICS_INTERVAL con la llamada MQINQ.

SyncPoint (MQLONG)

Esto indica si el gestor de colas local soporta unidades de trabajo y sincronización con las llamadas MQGET, MQPUT y MQPUT1 .

MQSP_AVAILABLE
Unidades de trabajo y sincronización disponibles.

MQSP_NOT_AVAILABLE
Unidades de trabajo y sincronización no disponibles.

- En z/OS este valor nunca se devuelve.

Para determinar el valor de este atributo, utilice el selector MQIA_SYNCPOINT con la llamada MQINQ.

TCPChannels (MQLONG)

Es el número máximo de canales que pueden ser actuales, o clientes que se pueden conectar, que utilizan el protocolo de transmisión TCP/IP.

El valor debe estar en el rango de 0 a 9999, con un valor predeterminado de 200. Si especifica 0, no se utiliza TCP/IP.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_TCP_CHANNELS con la llamada MQINQ.

TCPKeepAlive (MQLONG)

Especifica si se debe utilizar TCP KEEPALIVE para comprobar que el otro extremo de la conexión sigue estando disponible. Si no está disponible, el canal se cierra.

El valor puede ser uno de los siguientes:

MQTCPKEEP_YES
Utilice TCP KEEPALIVE tal como se especifica en el conjunto de datos de configuración de perfil TCP. Si especifica el atributo de canal KeepAliveInterval (KAINT), se utiliza el valor en el que se establece.

MQTCPKEEP_NO
No utilice TCP KEEPALIVE. Este es el valor predeterminado.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_TCP_KEEP_ALIVE con la llamada MQINQ.

TCPName (MQCHAR8)

Este es el nombre del sistema TCP/IP único o predeterminado que está utilizando, en función del valor de TCPStackType. El valor predeterminado es TCPIP.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQCA_TCP_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_TCP_NAME_LENGTH.

TCPStackType (MQLONG)

Especifica si el iniciador de canal puede utilizar sólo el espacio de direcciones TCP/IP especificado en TCPName, o puede enlazar opcionalmente con cualquier dirección TCP/IP seleccionada

El valor puede ser uno de los siguientes:

MQTCPSTACK_SINGLE
El iniciador de canal sólo puede utilizar los espacios de direcciones TCP/IP especificados en TCPName. Este es el valor predeterminado.

MQTCPSTACK_MULTIPLE
El iniciador de canal puede utilizar cualquier espacio de direcciones TCP/IP disponible para él. El valor predeterminado es el especificado en TCPName si no se especifica ningún otro para un canal o escucha.

Este atributo solo está soportado en z/OS .

Para determinar el valor de este atributo, utilice el selector MQIA_TCP_STACK_TYPE con la llamada MQINQ.

Registro de TraceRoute(MQLONG)

Esto controla el registro de la información de ruta de rastreo.

El valor puede ser uno de los siguientes:

MQRECORDING_INHABILITADO
No se permite añadir mensajes de ruta de rastreo.

MQRECORDING_Q
Coloque los mensajes de ruta de rastreo en una cola con nombre fijo.

MQRECORDING_MSG
Transfiera los mensajes de ruta de rastreo a una cola determinada utilizando el propio mensaje. Este es el valor predeterminado

Para determinar el valor de este atributo, utilice el selector MQIA_TRACE_ROUTE_REGISTRAR con la llamada MQINQ.

TriggerInterval (MQLONG)

Es un intervalo de tiempo (en milisegundos) utilizado para restringir el número de mensajes desencadenantes. Esto sólo es relevante cuando *TriggerType* es MQTT_FIRST. En este caso, los mensajes desencadenantes normalmente se generan sólo cuando llega un mensaje adecuado a la cola y la cola estaba vacía anteriormente. En determinadas circunstancias, sin embargo, se puede generar un mensaje desencadenante adicional con el desencadenante MQTT_FIRST incluso si la cola no estaba vacía. Estos mensajes desencadenantes adicionales no se generan con más frecuencia que cada *TriggerInterval* milisegundos.

Para obtener más información sobre el desencadenamiento, consulte Desencadenamiento de canales .

El valor no es menor que 0 ni mayor que 999 999 999. El valor predeterminado es 999.999.999.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_INTERVAL con la llamada MQINQ.

TriggerInterval (MQLONG)

Es un intervalo de tiempo (en milisegundos) utilizado para restringir el número de mensajes desencadenantes. Esto sólo es relevante cuando *TriggerType* es MQTT_FIRST. En este caso, los mensajes desencadenantes normalmente se generan sólo cuando llega un mensaje adecuado a la cola y la cola estaba vacía anteriormente. En determinadas circunstancias, sin embargo, se puede generar un mensaje desencadenante adicional con el desencadenante MQTT_FIRST incluso si la cola

no estaba vacía. Estos mensajes desencadenantes adicionales no se generan con más frecuencia que cada *TriggerInterval* milisegundos.

Para obtener más información sobre el desencadenamiento, consulte Desencadenamiento de canales .

El valor no es menor que 0 ni mayor que 999 999 999. El valor predeterminado es 999.999.999.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_INTERVAL con la llamada MQINQ.

Versión (MQCFST)

Esta es la versión del código de WebSphere MQ como VVRRMMFF, donde:

VV-Versión

RR-Release

MM-Nivel de mantenimiento

FF-Nivel de arreglo

XrCapability(MQLONG)

Esto controla si el gestor de colas da soporte a los mandatos de WebSphere MQ Telemetry .

El valor puede ser uno de los siguientes:

MQCAP_SUPPORTED
El componente WebSphere MQ Telemetry instalado y los mandatos de telemetría están soportados.

MQCAP_NO_SOPORTADO
webSphere MQ El componente de telemetría no está instalado.

Este atributo sólo está soportado en IBM i, sistemas Unix y Windows.

Para determinar el valor de este atributo, utilice el selector MQIA_XR_ABILITY con la llamada MQINQ .

Atributos para colas

Hay cinco tipos de definición de cola. Algunos atributos de cola se aplican a todos los tipos de cola; otros atributos de cola se aplican sólo a determinados tipos de cola.

Tipos de cola

El gestor de colas da soporte a los siguientes tipos de definición de cola:

Cola local
Puede almacenar mensajes en una cola local. En z/OS puede convertirlo en una cola compartida o privada.

Una cola se conoce en un programa como *local* si es propiedad del gestor de colas al que está conectado el programa. Puede obtener mensajes y transferirlos en las colas locales.

El objeto de definición de cola contiene la información de definición de la cola, así como los mensajes físicos colocados en cola.

Cola del gestor de colas local
La cola existe en el gestor de colas local. La cola se conoce como cola privada en z/OS.

Cola compartida (soloz/OS)
La cola existe en un repositorio compartido al que pueden acceder todos los gestores de colas que pertenecen al grupo de compartición de colas propietario del repositorio compartido.

Las aplicaciones conectadas a cualquier gestor de colas del grupo de compartición de colas pueden colocar mensajes y eliminarlos de las colas de este tipo. Estas colas son efectivamente las mismas que las colas locales. El valor del atributo de cola *QType* es MQQT_LOCAL.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes y eliminar mensajes de colas de este tipo. El valor del atributo de cola *QType* es MQQT_LOCAL.

Cola de clúster

Puede almacenar mensajes en una cola de clúster en el gestor de colas donde está definido. Una cola de clúster es una cola que se aloja en un gestor de colas de clúster y que está disponible para otros gestores de colas del clúster. El valor del atributo de cola *QType* es MQQT_CLUSTER.

Una definición de cola de clúster se anuncia en otros gestores de colas del clúster. Los otros gestores de colas del clúster pueden transferir mensajes a una cola de clúster sin necesidad de que haya una definición de cola remota correspondiente. Una cola de clúster se puede anunciar en más de un clúster utilizando una lista de nombres de clúster.

Cuando se anuncia una cola, cualquier gestor de colas del clúster puede poner mensajes en ella. Para transferir un mensaje, el gestor de colas debe averiguar, en los repositorios completos, donde está alojada la cola. A continuación, añade información de direccionamiento al mensaje y pone el mensaje a una cola de transmisión de clúster.

Excepto en z/OS, un gestor de colas puede almacenar mensajes para otros gestores de colas de un clúster en varias colas de transmisión. Puede configurar un gestor de colas para almacenar mensajes en varias colas de transmisión de clúster de dos maneras diferentes. Si establece el atributo de gestor de colas DEFCLXQ en CHANNEL, se crea automáticamente una cola de transmisión de clúster de clúster diferente desde SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE para cada canal de clúster emisor. Si establece la opción de cola de transmisión CLCHNAME para que coincida con uno o varios canales de clúster emisor, el gestor de colas puede almacenar mensajes para los canales coincidentes en esa cola de transmisión.

Una cola de clúster puede ser una cola que se comparte entre miembros de un grupo de compartimiento de colas en IBM WebSphere MQ for z/OS.

Cola remota

Una cola remota no es una cola física; es la definición local de una cola que existe en un gestor de colas remoto. La definición local de la cola remota contiene información que indica al gestor de colas local cómo direccionar los mensajes al gestor de colas remoto.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes en colas de este tipo; los mensajes se colocan en la cola de transmisión local que se utiliza para direccionar los mensajes al gestor de colas remoto. Las aplicaciones no pueden eliminar mensajes de colas remotas. El valor del atributo de cola *QType* es MQQT_REMOTE.

También puede utilizar una definición de cola remota para:

- Alias de cola de respuestas

  En este caso, el nombre de la definición es el nombre de una cola de respuestas. Para obtener más información, consulte Alias de cola de respuesta y clústeres.

- Alias de gestor de colas

  En este caso, el nombre de la definición es un alias para un gestor de colas, no el nombre de una cola. Para obtener más información, consulte Alias y clústeres de gestores de colas.

Cola alias

No es una cola física; es un nombre alternativo para una cola local, una cola compartida, una cola de clúster o una cola remota. El nombre de la cola en la que se resuelve el alias forma parte de la definición de la cola alias.

Las aplicaciones conectadas al gestor de colas local pueden colocar mensajes en colas de este tipo; los mensajes se colocan en la cola en la que se resuelve el alias. Las aplicaciones pueden eliminar mensajes de colas de este tipo si el alias se resuelve en una cola local, una cola compartida o una cola de clúster que tiene una instancia local. El valor del atributo de cola *QType* es MQQT_ALIAS.

Cola modelo

No es una cola física; es un conjunto de atributos de cola a partir de los cuales se puede crear una cola local.

Los mensajes no se pueden almacenar en colas de este tipo.

Atributos de colas

Algunos atributos de cola se aplican a todos los tipos de cola; otros atributos de cola se aplican sólo a determinados tipos de cola. Los tipos de cola a los que se aplica un atributo se muestran en las tablas Tabla 573 en la página 818 y posteriores.

La Tabla 573 en la página 818 resume los atributos específicos de las colas. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos que se muestran en esta sección son nombres descriptivos utilizados con las llamadas MQINQ y MQSET ; los nombres son los mismos que para los mandatos PCF. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte Mandatos de script (MQSC) para obtener más detalles.

Tabla 573. Atributos para colas. Las columnas se aplican de la siguiente manera:

- La columna para colas locales también se aplica a colas compartidas.
- La columna para colas modelo indica qué atributos hereda la cola local creada a partir de la cola modelo.
- La columna para colas de clúster indica los atributos que se pueden consultar cuando la cola de clúster se abre solo para consulta, o para consulta y salida. Si se consulta cualquier otro atributo, la llamada devuelve el código de terminación MQCC_WARNING y el código de razón MQRC_SELECTOR_NOT_FOR_TYPE (2068).

  Si la cola de clúster se abre para realizar consultas más una o más entradas, examinar o establecer, en su lugar se aplica la columna correspondiente a las colas locales.

  Si la cola de clúster se abre solo para consultas, o para consultas y salidas, además de especificar el nombre del gestor de colas base, en su lugar se aplica la columna para colas locales.

Atributo	Descripción	Local	Modelo	Alias	Remotos	Clúster
AlterationDate	Fecha en que se modificó por última vez la definición	✓		✓	✓	
AlterationTime	Hora a la que se modificó por última vez la definición	✓		✓	✓	
BackoutRequeueQName	Nombre de cola de reposición en cola de restitución excesivo	✓	✓			
BackoutThreshold	Umbral de restituciones	✓	✓			
BaseQName	Nombre de cola al que se resuelve el alias			✓		
CFStrucName	Nombre de estructura de recurso de acoplamiento	✓	✓			
CLCHNAME	Nombres de canal de clúster emisor	✓	✓			
ClusterName	Nombre del clúster al que pertenece la cola	✓		✓	✓	✓
ClusterNamelist	Nombre del objeto de lista de nombres que contiene nombres de clústeres a los que pertenece la cola	✓		✓	✓	
CLWLQueuePriority	Prioridad de cola de carga de trabajo de clúster	✓		✓	✓	✓
CLWLQueueRank	Rango de cola de carga de trabajo de clúster	✓		✓	✓	✓
CLWLUseQ	Utilizar cola remota	✓				
CreationDate	Fecha de creación de la cola	✓				
CreationTime	Hora a la que se ha creado la cola	✓				
CurrentQDepth	Profundidad de cola actual	✓				
DefaultPutResponse	Resp predet de transferencia	✓	✓	✓	✓	

Tabla 573. Atributos para colas. Las columnas se aplican de la siguiente manera:

- La columna para colas locales también se aplica a colas compartidas.
- La columna para colas modelo indica qué atributos hereda la cola local creada a partir de la cola modelo.
- La columna para colas de clúster indica los atributos que se pueden consultar cuando la cola de clúster se abre solo para consulta, o para consulta y salida. Si se consulta cualquier otro atributo, la llamada devuelve el código de terminación MQCC_WARNING y el código de razón MQRC_SELECTOR_NOT_FOR_TYPE (2068).

  Si la cola de clúster se abre para realizar consultas más una o más entradas, examinar o establecer, en su lugar se aplica la columna correspondiente a las colas locales.

  Si la cola de clúster se abre solo para consultas, o para consultas y salidas, además de especificar el nombre del gestor de colas base, en su lugar se aplica la columna para colas locales.

(continuación)

Atributo	Descripción	Local	Modelo	Alias	Remotos	Clúster
DefBind	Enlace predeterminado	✓		✓	✓	✓
DefinitionType attribute	Tipo de definición de cola	✓	✓			
DefInputOpenOption	Opción abierta de entrada predeterminada	✓	✓			
DefPersistence	Persistencia de mensajes predeterminada	✓	✓	✓	✓	✓
DefPriority	Prioridad de mensajes predeterminada	✓	✓	✓	✓	✓
DefReadAhead	Lectura anticipada predeterminada	✓	✓	✓		
DistLists	Soporte de lista de distribución	✓	✓			
HardenGetBackout	Si se debe mantener un recuento de restituciones preciso	✓	✓			
IndexType	Tipo de índice	✓	✓			
InhibitGet	Si se permiten operaciones get para la cola	✓	✓	✓		
InhibitPut	Si se permiten operaciones de colocación para la cola	✓	✓	✓	✓	✓
InitiationQName	Nombre de cola de inicio	✓	✓			
MaxMsgLength	Longitud máxima de mensaje en bytes	✓	✓			
MaxQDepth	Profundidad máxima de la cola	✓	✓			
MsgDeliverySequence attribute	Secuencia de entrega de mensajes	✓	✓			
NonPersistentMessage Class	Objetivo de fiabilidad para mensajes no persistentes	✓	✓			
OpenInputCount	Número de aperturas para entrada	✓				
OpenOutputCount	Número de aperturas para salida	✓				
PropertyControl	Control de propiedad	✓	✓	✓		
ProcessName	Nombre de proceso	✓	✓			
QDepthHighEvent attribute	Si se generan sucesos de profundidad de cola alta	✓	✓			
QDepthHighLimit	Límite alto para profundidad de cola	✓	✓			
QDepthLowEvent attribute	Si se generan sucesos de profundidad de cola baja	✓	✓			
QDepthLowLimit attribute	Límite bajo para profundidad de cola	✓	✓			

Tabla 573. Atributos para colas. Las columnas se aplican de la siguiente manera:

- La columna para colas locales también se aplica a colas compartidas.
- La columna para colas modelo indica qué atributos hereda la cola local creada a partir de la cola modelo.
- La columna para colas de clúster indica los atributos que se pueden consultar cuando la cola de clúster se abre solo para consulta, o para consulta y salida. Si se consulta cualquier otro atributo, la llamada devuelve el código de terminación MQCC_WARNING y el código de razón MQRC_SELECTOR_NOT_FOR_TYPE (2068).

  Si la cola de clúster se abre para realizar consultas más una o más entradas, examinar o establecer, en su lugar se aplica la columna correspondiente a las colas locales.

  Si la cola de clúster se abre solo para consultas, o para consultas y salidas, además de especificar el nombre del gestor de colas base, en su lugar se aplica la columna para colas locales.

(continuación)

Atributo	Descripción	Local	Modelo	Alias	Remotos	Clúster
QDepthMaxEvent	Si se generan sucesos de cola llena	✓	✓			
QDesc	Descripción de la cola	✓	✓	✓	✓	✓
QName	Nombre de cola	✓		✓	✓	✓
QServiceInterval	Destino para intervalo de servicio de cola	✓	✓			
QServiceIntervalEvent attribute	Si se generan sucesos de intervalo de servicio alto o de intervalo de servicio correcto	✓	✓			
QSGDisp attribute	Disposición del grupo de compartición de colas	✓		✓	✓	
QueueAccounting	Recopilación de datos de contabilidad de cola	✓	✓	✓	✓	✓
QueueMonitoring	Datos de supervisión en línea para colas	✓	✓			
QueueStatistics	recopilación de datos de estadísticas de cola	✓	✓	✓	✓	✓
QType	Tipo de cola	✓		✓	✓	✓
RemoteQMgrName	Nombre del gestor de colas remoto				✓	
RemoteQName	Nombre de cola remota				✓	
RetentionInterval	Intervalo de retención	✓	✓			
Scope	Si también existe una entrada para la cola en un directorio de célula	✓		✓	✓	
Shareability	Compartibilidad de cola	✓	✓			
StorageClass	Clase de almacenamiento para cola	✓	✓			
TriggerControl	Activar control	✓	✓			
TriggerData	Datos desencadenantes	✓	✓			
TriggerDepth	Profundidad de desencadenante	✓	✓			
TriggerMsgPriority	Prioridad de mensaje de umbral para desencadenantes	✓	✓			
TriggerType	Tipo de desencadenante	✓	✓			
Usage attribute	Uso de la cola	✓	✓			
XmitQName	Nombre de cola de transmisión				✓	

Conceptos relacionados

Colas de clúster

AlterationDate (MQCHAR12)

Fecha en la que se cambió por última vez la definición.

Local	Modelo	Alias	Remotos	Clúster
X		X	X	

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es YYYY-MM-DD, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes (por ejemplo, 1992-09-23¬¬, donde ¬¬ representa dos caracteres en blanco).

Los valores de determinados atributos (por ejemplo, *CurrentQDepth*) cambian a medida que opera el gestor de colas. Los cambios en estos atributos no afectan a *AlterationDate*.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Hora a la que se modificó por última vez la definición.

Local	Modelo	Alias	Remotos	Clúster
X		X	X	

Es la hora en que se modificó por última vez la definición. El formato de la hora es HH.MM.SS utilizando el reloj de 24 horas, con un cero inicial si la hora es inferior a 10 (por ejemplo, 09.10.20).

- En z/OS, la hora es Greenwich Mean Time (GMT), sujeta a que el reloj del sistema se establezca correctamente en GMT.
- En otros entornos, la hora es local.

Los valores de determinados atributos (por ejemplo, *CurrentQDepth*) cambian a medida que opera el gestor de colas. Los cambios en estos atributos no afectan a *AlterationTime*.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_TIME_LENGTH.

BackoutRequeueQName (MQCHAR48)

Este es el nombre de cola de reposición en cola de restitución excesivo. Aparte de permitir que se consulte su valor, el gestor de colas no realiza ninguna acción basada en el valor de este atributo.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Las aplicaciones que se ejecutan en WebSphere Application Server y las que utilizan WebSphere MQ Application Server Facilities utilizan este atributo para determinar dónde deben ir los mensajes que se han restituido. Para todas las demás aplicaciones, el gestor de colas no realiza ninguna acción basada en el valor del atributo.

WebSphere MQ classes for JMS utiliza este atributo para determinar dónde transferir un mensaje que ya se ha restituido el número máximo de veces especificado por el atributo *BackoutThreshold* .

Para determinar el valor de este atributo, utilice el selector MQCA_BACKOUT_REQ_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

BackoutThreshold (MQLONG)
Este es el umbral de restitución. Aparte de permitir que se consulte su valor, el gestor de colas no realiza ninguna acción basada en el valor de este atributo.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Las aplicaciones que se ejecutan dentro de WebSphere Application Server y las que utilizan WebSphere MQ Application Server Facilities utilizarán este atributo para determinar si se debe restituir un mensaje. Para todas las demás aplicaciones, el gestor de colas no realiza ninguna acción basada en el valor del atributo.

WebSphere MQ classes for JMS utiliza este atributo para determinar cuántas veces se debe permitir la restitución de un mensaje antes de transferir el mensaje a la cola especificada por el atributo *BackoutRequeueQName* .

Para determinar el valor de este atributo, utilice el selector MQIA_BACKOUT_THRESHOLD con la llamada MQINQ.

BaseQName (MQCHAR48)
Es el nombre de una cola definida en el gestor de colas local.

Local	Modelo	Alias	Remotos	Clúster
		X		

(Para obtener más información sobre los nombres de cola, consulte MQOD-Campo ObjectName.) La cola es de uno de los tipos siguientes:

MQQT_LOCAL
Cola local.

MQQT_REMOTE
Definición local de una cola remota.

MQQT_CLUSTER
Cola de clúster.

Para determinar el valor de este atributo, utilice el selector MQCA_BASE_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

BaseType (MQCFIN)
El tipo de objeto en el que se resuelve el alias.

Local	Modelo	Alias	Remotos	Clúster
		X		

Es uno de los valores siguientes:

MQOT_Q
El tipo de objeto base es una cola

MQOT_TOPIC
El tipo de objeto base es un tema

CFStrucName (MQCHAR12)
Es el nombre de la estructura del recurso de acoplamiento donde se almacenan los mensajes de la cola. El primer carácter del nombre está en el rango de A a Z, y los caracteres restantes están en el rango de A a Z, de 0 a 9 o en blanco.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Para obtener el nombre completo de la estructura en el recurso de acoplamiento, sufijo el valor del atributo de gestor de colas *QSGName* con el valor del atributo de cola *CFStrucName* .

Este atributo sólo se aplica a las colas compartidas; se ignora si *QSGDisp* no tiene el valor MQQSGD_SHARED.

Para determinar el valor de este atributo, utilice el selector MQCA_CF_STRUC_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CF_STRUC_NAME_LENGTH.

Este atributo sólo está soportado en z/OS.

ClusterChannelNombre (MQCHAR20)

`ClusterChannelNombre` es el nombre genérico de los canales de clúster emisor que utilizan esta cola como cola de transmisión. El atributo especifica los canales de clúster emisor han enviado mensajes a un canal de clúster receptor desde esta cola de transmisión de clúster. `ClusterChannelNombre` no está soportado en z/OS.

Local	Modelo	Alias	Remotos	Clúster
✓	✓			

La configuración predeterminada del gestor de colas es que todos los canales de clúster emisor envíen mensajes desde una sola cola de transmisión, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. La configuración predeterminada se puede cambiar modificando el atributo de gestor de colas, `DefClusterXmitQueueType`. El valor predeterminado del atributo es `SCTQ`. Puede cambiar el valor a `CHANNEL`. Si establece el atributo `DefClusterXmitQueueTipo` en `CHANNEL`, cada canal de clúster emisor utiliza de forma predeterminada una cola de transmisión de clúster específica, `SYSTEM.CLUSTER.TRANSMIT.`*`ChannelName`*.

También puede establecer el atributo de cola de transmisión `ClusterChannelName` en un canal de clúster emisor manualmente. Los mensajes destinados al gestor de colas conectado por el canal de clúster emisor se almacenan en la cola de transmisión que identifica el canal de clúster emisor. No se almacenan en la cola de transmisión de clúster predeterminada. Si establece el atributo `ClusterChannelName` en blancos, el canal conmuta a la cola de transmisión de clúster predeterminada cuando se reinicia el canal. La cola predeterminada es `SYSTEM.CLUSTER.TRANSMIT.`*`ChannelName`* o `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, en función del valor del atributo `DefClusterXmitQueueType` del gestor de colas.

Especificando asteriscos, ("`*`"), en `ClusterChannelName` para asociar una cola de transmisión con un conjunto de canales de clúster emisor. Los asteriscos pueden estar al principio, al final o en cualquier posición intermedia de la serie de nombre de canal. `ClusterChannelName` está limitado a una longitud de 20 caracteres: `MQ_CHANNEL_NAME_LENGTH`.

ClusterName (MQCHAR48)

Es el nombre del clúster al que pertenece la cola.

Local	Modelo	Alias	Remotos	Clúster
X		X	X	X

Si la cola pertenece a más de un clúster, *`ClusterNamelist`* especifica el nombre de un objeto de lista de nombres que identifica los clústeres y *`ClusterName`* está en blanco. Al menos uno de *`ClusterName`* y *`ClusterNamelist`* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_CLUSTER_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CLUSTER_NAME_LENGTH.

ClusterNamelist (MQCHAR48)
Es el nombre de un objeto de lista de nombres que contiene los nombres de los clústeres a los que pertenece esta cola.

Local	Modelo	Alias	Remotos	Clúster
X		X	X	

Si la cola sólo pertenece a un clúster, el objeto de lista de nombres sólo contiene un nombre. De forma alternativa, se puede utilizar *ClusterName* para especificar el nombre del clúster, en cuyo caso *ClusterNamelist* está en blanco. Al menos uno de *ClusterName* y *ClusterNamelist* debe estar en blanco.

Para determinar el valor de este atributo, utilice el selector MQCA_CLUSTER_NAMELIST con la llamada MQINQ. La longitud de este atributo la proporciona MQ_NAMELIST_NAME_LENGTH.

CLWLQueuePriority (MQLONG)
Es la prioridad de cola de carga de trabajo de clúster, un valor comprendido entre 0 y 9 que representa la prioridad de la cola.

Local	Modelo	Alias	Remotos	Clúster
X		X	X	X

Para obtener más información, consulte Colas de clúster.

Para determinar el valor de este atributo, utilice el selector MQIA_CLWL_Q_PRIORITY con la llamada MQINQ.

CLWLQueueRank (MQLONG)
Este es el rango de cola de carga de trabajo de clúster, un valor en el rango de 0 a 9 que representa el rango de la cola.

Local	Modelo	Alias	Remotos	Clúster
X		X	X	X

Para obtener más información, consulte Colas de clúster.

Para determinar el valor de este atributo, utilice el selector MQIA_CLWL_Q_RANK con la llamada MQINQ.

CLWLUseQ (MQLONG)
Esto define el comportamiento de un MQPUT cuando la cola de destino tiene una instancia local y al menos una instancia de clúster remoto. Si la transferencia se origina en un canal de clúster, este atributo no es aplicable.

Local	Modelo	Alias	Remotos	Clúster
X				

El valor puede ser uno de los siguientes:

MQCLWL_USEQ_ANY
Utilice colas remotas y locales.

MQCLWL_USEQ_LOCAL
No utilice colas remotas.

MQCLWL_USEQ_AS_Q_MGR
Heredar definición de MQIA_CLWL_USEQ del gestor de colas.

Para obtener más información, consulte Colas de clúster.

Para determinar el valor de este atributo, utilice el selector MQCA_CLWL_USEQ con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CLWL_USEQ_LENGTH.

CreationDate (MQCHAR12)

Es la fecha en la que se creó la cola.

Local	Modelo	Alias	Remotos	Clúster
X				

El formato de la fecha es YYYY-MM-DD, rellenado con dos espacios en blanco finales para hacer que la longitud sea de 12 bytes (por ejemplo, 2013-09-23¬¬, donde ¬¬ representa 2 caracteres en blanco).

- En IBM i, la fecha de creación de una cola puede diferir de la de la entidad del sistema operativo subyacente (archivo o espacio de usuario) que representa la cola.

Para determinar el valor de este atributo, utilice el selector MQCA_CREATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CREATION_DATE_LENGTH.

CreationTime (MQCHAR8)

Es la hora en que se ha creado la cola.

Local	Modelo	Alias	Remotos	Clúster
X				

El formato de la hora es HH.MM.SS utilizando el reloj de 24 horas, con un cero inicial si la hora es inferior a 10 (por ejemplo, 09.10.20).

- En z/OS, la hora es Greenwich Mean Time (GMT), sujeta a que el reloj del sistema se establezca correctamente en GMT.
- En otros entornos, la hora es local.
- En IBM i, la hora de creación de una cola puede diferir de la de la entidad del sistema operativo subyacente (archivo o espacio de usuario) que representa la cola.

Para determinar el valor de este atributo, utilice el selector MQCA_CREATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_CREATION_TIME_LENGTH.

CurrentQDepth (MQLONG)

Es el número de mensajes que hay en la cola actualmente.

Local	Modelo	Alias	Remotos	Clúster
X				

Se incrementa durante una llamada MQPUT y durante la restitución de una llamada MQGET. Se reduce durante una llamada MQGET no examinada y durante la restitución de una llamada MQPUT. El efecto de esto es que el recuento incluye los mensajes que se han colocado en la cola dentro de una unidad de trabajo, pero que todavía no se han confirmado, aunque no sean elegibles para ser recuperados por la llamada MQGET. De forma similar, excluye los mensajes que se han recuperado dentro de una unidad de trabajo utilizando la llamada MQGET, pero que todavía no se han confirmado.

El recuento también incluye los mensajes que han pasado su hora de caducidad pero que todavía no se han descartado, aunque estos mensajes no son aptos para ser recuperados. Consulte Campo MQMD-Caducidad para obtener más información.

El proceso de unidad de trabajo y la segmentación de mensajes pueden hacer que *CurrentQDepth* supere *MaxQDepth*. Sin embargo, esto no afecta a la capacidad de recuperación de los mensajes; *todos* los mensajes de la cola se pueden recuperar utilizando la llamada MQGET de la forma normal.

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_CURRENT_Q_DEPTH con la llamada MQINQ.

Respuesta DefaultPut(MQLONG)

Especifica el tipo de respuesta que debe utilizarse para las operaciones de colocación en la cola cuando una aplicación especifica MQPMO_RESPONSE_AS_Q_DEF.

Local	Modelo	Alias	Remotos	Clúster
X	X	X	X	

Es uno de los valores siguientes:

MQPRT_SYNC_RESPONSE
La operación de colocación se emite de forma síncrona, devolviendo una respuesta.

MQPRT_ASYNC_RESPONSE
La operación de transferencia se emite de forma asíncrona, devolviendo un subconjunto de campos MQMD.

DefBind (MQLONG)

Este es el enlace predeterminado que se utiliza cuando se especifica MQOO_BIND_AS_Q_DEF en la llamada MQOPEN y la cola es una cola de clúster.

Local	Modelo	Alias	Remotos	Clúster
X		X	X	X

El valor puede ser uno de los siguientes:

MQBND_BIND_ON_OPEN
Enlace arreglado por la llamada MQOPEN.

MQBND_BIND_NOT_FIXED
Enlace no arreglado.

MQBND_BIND_ON_GROUP
Permite a una aplicación solicitar que un grupo de mensajes se asigne a la misma instancia de destino. Puesto que este valor es nuevo en IBM WebSphere MQ Version 7.1, no se debe utilizar si alguna de las aplicaciones que abren esta cola se conecta a IBM WebSphere MQ Version 7.0.1 o a gestores de colas anteriores.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_BIND con la llamada MQINQ.

DefinitionType (MQLONG)

Indica cómo se ha definido la cola.

Local	Modelo	Alias	Remotos	Clúster
X	X			

El valor puede ser uno de los siguientes:

MQQDT_PREDEFINED
La cola es una cola permanente creada por el administrador del sistema; sólo el administrador del sistema puede suprimirla.

Las colas predefinidas se crean utilizando el mandato MQSC de DEFINE y solo se pueden suprimir utilizando el mandato MQSC de DELETE . Las colas predefinidas no se pueden crear a partir de colas modelo.

Los mandatos pueden ser emitidos por un operador o por un usuario autorizado que envía un mensaje de mandato a la cola de entrada de mandatos (consulte CommandInputQName para obtener más información).

MQQDT_PERMANENT_DYNAMIC
La cola es una cola permanente creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor MQQDT_PERMANENT_DYNAMIC para el atributo *DefinitionType* .

Este tipo de cola se puede suprimir utilizando la llamada MQCLOSE. Consulte "MQCLOSE-Cerrar objeto" en la página 628 para obtener más detalles.

El valor del atributo *QSGDisp* para una cola dinámica permanente es MQQSGD_Q_MGR.

MQQDT_TEMPORARY_DYNAMIC
La cola es una cola temporal creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor MQQDT_TEMPORARY_DYNAMIC para el atributo *DefinitionType* .

Este tipo de cola se suprime automáticamente mediante la llamada MQCLOSE cuando la aplicación que la ha creado la cierra.

El valor del atributo *QSGDisp* para una cola dinámica temporal es MQQSGD_Q_MGR.

MQQDT_SHARED_DYNAMIC
La cola es una cola permanente compartida creada por una aplicación que emite una llamada MQOPEN con el nombre de una cola modelo especificada en el descriptor de objeto MQOD. La definición de cola modelo tenía el valor MQQDT_SHARED_DYNAMIC para el atributo *DefinitionType* .

Este tipo de cola se puede suprimir utilizando la llamada MQCLOSE. Consulte "MQCLOSE-Cerrar objeto" en la página 628 para obtener más detalles.

El valor del atributo *QSGDisp* para una cola dinámica compartida es MQQSGD_SHARED.

Este atributo en una definición de cola modelo no indica cómo se ha definido la cola modelo, porque las colas modelo siempre están predefinidas. En su lugar, el valor de este atributo en la cola modelo se utiliza para determinar el *DefinitionType* de cada una de las colas dinámicas creadas a partir de la definición de cola modelo utilizando la llamada MQOPEN.

Para determinar el valor de este atributo, utilice el selector MQIA_DEFINITION_TYPE con la llamada MQINQ.

DefInputOpenOption (MQLONG)
Esta es la forma predeterminada en la que se abre la cola para la entrada.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Se aplica si se especifica la opción MQOO_INPUT_AS_Q_DEF en la llamada MQOPEN cuando se abre la cola. El valor puede ser uno de los siguientes:

MQOO_INPUT_EXCLUSIVE
Abra la cola para obtener mensajes con acceso exclusivo.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada falla con el código de razón MQRC_OBJECT_IN_USE si la cola está abierta actualmente por esta u otra aplicación para cualquier tipo de entrada (MQOO_INPUT_SHARED o MQOO_INPUT_EXCLUSIVE).

MQOO_INPUT_SHARED
Abra la cola para obtener mensajes con acceso compartido.

La cola se abre para su uso con las llamadas MQGET posteriores. La llamada puede realizarse correctamente si la cola está abierta actualmente por esta u otra aplicación con MQOO_INPUT_SHARED, pero falla con el código de razón MQRC_OBJECT_IN_USE si la cola está abierta actualmente con MQOO_INPUT_EXCLUSIVE.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_INPUT_OPEN_OPTION con la llamada MQINQ.

DefPersistence (MQLONG)

Es la persistencia predeterminada de los mensajes en la cola. Se aplica si se especifica MQPER_PERSISTENCE_AS_Q_DEF en el descriptor de mensaje cuando se coloca el mensaje.

Local	Modelo	Alias	Remotos	Clúster
X	X	X	X	X

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la persistencia predeterminada se toma del valor de este atributo en la *primera* definición de la vía de acceso en el momento de la llamada MQPUT o MQPUT1 . Puede ser lo siguiente:

- una cola alias
- Una cola local
- Una definición local de una cola remota
- Un alias de gestor de colas
- Una cola de transmisión (por ejemplo, la cola *DefXmitQName*)

El valor puede ser uno de los siguientes:

MQPER_PERSISTENT
El mensaje sobrevive a las anomalías del sistema y se reinicia el gestor de colas. Los mensajes persistentes no se pueden colocar en:

- Colas dinámicas temporales
- Colas compartidas que se correlacionan con un objeto CFSTRUCT en CFLEVEL (2) o inferior, o donde el objeto CFSTRUCT se define como RECOVER (NO).

Los mensajes persistentes se pueden colocar en colas dinámicas permanentes y colas predefinidas.

MQPER_NOT_PERSISTENT
Normalmente, el mensaje no sobrevive a las anomalías del sistema o a los reinicios del gestor de colas. Esto se aplica incluso si se encuentra una copia intacta del mensaje en el almacenamiento auxiliar durante un reinicio del gestor de colas.

En el caso de las colas compartidas, los mensajes no persistentes *sí* sobreviven a los reinicios de los gestores de colas en el grupo de compartición de colas, pero no sobreviven a los errores del recurso de acoplamiento utilizado para almacenar mensajes en las colas compartidas.

Los mensajes persistentes y no persistentes pueden existir en la misma cola.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_PERSISTENCE con la llamada MQINQ.

DefPriority (MQLONG)

Esta es la prioridad predeterminada para los mensajes de la cola. Esto se aplica si se especifica MQPRI_PRIORITY_AS_Q_DEF en el descriptor de mensaje cuando el mensaje se coloca en la cola.

Local	Modelo	Alias	Remotos	Clúster
X	X	X	X	X

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, la prioridad predeterminada para el mensaje se toma del valor de este atributo en la *primera* definición de la vía de acceso en el momento de la operación de colocación. Puede ser lo siguiente:

- una cola alias
- Una cola local
- Una definición local de una cola remota

- Un alias de gestor de colas
- Una cola de transmisión (por ejemplo, la cola *DefXmitQName*)

La forma en que se coloca un mensaje en una cola depende del valor del atributo *MsgDeliverySequence* de la cola:

- Si el atributo *MsgDeliverySequence* es MQMDS_PRIORITY, la posición lógica en la que se coloca un mensaje en la cola depende del valor del campo *Priority* en el descriptor de mensaje.
- Si el atributo *MsgDeliverySequence* es MQMDS_FIFO, los mensajes se colocan en la cola como si tuvieran una prioridad igual a la *DefPriority* de la cola resuelta, independientemente del valor del campo *Priority* en el descriptor de mensaje. Sin embargo, el campo *Priority* conserva el valor especificado por la aplicación que ha colocado el mensaje. Consulte Atributo de secuenciaMsgDelivery para obtener más información.

Las prioridades están en el rango de cero (menor) a *MaxPriority* (mayor); consulte el atributo MaxPriority.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_PRIORITY con la llamada MQINQ.

DefReadpor omisión (MQLONG)

Especifica el comportamiento de lectura anticipada predeterminado para los mensajes no persistentes entregados al cliente.

Local	Modelo	Alias	Remotos	Clúster
X	X	X		

DefReadSe puede establecer en uno de los valores siguientes:

MQREADA_NO
Los mensajes no persistentes no se envían al cliente antes de que las aplicaciones los soliciten. Puede perderse un mensaje no persistente como máximo, si el cliente finaliza de forma anómala.

MQREADA_SÍ
Los mensajes no persistentes se envían al cliente antes de que una aplicación los solicite. Los mensajes no persistentes se pueden perder si el cliente finaliza de forma anómala o si el cliente no consume todos los mensajes que se envían.

MQREADA_DISABLED
Lectura anticipada de mensajes no persistentes en no habilitados para esta cola. Los mensajes no se envían al cliente independientemente de si la aplicación cliente solicita la lectura anticipada.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_READ_AHEAD con la llamada MQINQ.

DefPResp (MQLONG)

El atributo de tipo de respuesta de colocación predeterminada (DEFPRESP) define el valor utilizado por las aplicaciones cuando el tipo PutResponsedentro de MQPMO se ha establecido en MQPMO_RESPONSE_AS_Q_DEF. Este atributo es válido para todos los tipos de cola.

Local	Modelo	Alias	Remotos	Clúster
Sí	Sí	Sí	Sí	Sí

El valor puede ser uno de los siguientes:

SYNC
La operación de colocación se emite de forma síncrona devolviendo una respuesta.

ASYNC
La operación de transferencia se emite de forma asíncrona, devolviendo un subconjunto de campos MQMD.

Para determinar el valor de este atributo, utilice el selector MQIA_DEF_PUT_RESPONSE_TYPE con la llamada MQINQ.

DistLists (MQLONG)

Indica si los mensajes de lista de distribución se pueden colocar en la cola.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Un agente de canal de mensajes (MCA) establece el atributo para informar al gestor de colas local si el gestor de colas del otro extremo del canal soporta listas de distribución. Este último gestor de colas (denominado gestor de colas de *asociación*) es el que recibe a continuación el mensaje, después de que un MCA emisor lo haya eliminado de la cola de transmisión local.

El MCA emisor establece el atributo siempre que establece una conexión con el MCA receptor en el gestor de colas asociado. De este modo, el MCA emisor puede hacer que el gestor de colas local coloque en la cola de transmisión sólo los mensajes que el gestor de colas asociado puede procesar correctamente.

Este atributo se utiliza principalmente con colas de transmisión, pero el proceso descrito se realiza independientemente del uso definido para la cola (consulte Atributo de uso).

El valor puede ser uno de los siguientes:

MQDL_SUPPORTED
Los mensajes de lista de distribución pueden almacenarse en la cola y transmitirse al gestor de colas asociado en ese formato. Esto reduce la cantidad de proceso necesaria para enviar el mensaje a varios destinos.

MQDL_NOT_SUPPORTED
Los mensajes de lista de distribución no se pueden almacenar en la cola porque el gestor de colas asociado no da soporte a las listas de distribución. Si una aplicación coloca un mensaje de lista de distribución y ese mensaje se va a colocar en esta cola, el gestor de colas divide el mensaje de lista de distribución y coloca los mensajes individuales en la cola. Esto aumenta la cantidad de proceso necesario para enviar el mensaje a varios destinos, pero garantiza que el gestor de colas asociado procese correctamente los mensajes.

Para determinar el valor de este atributo, utilice el selector MQIA_DIST_LISTS con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

Este atributo no está soportado en z/OS.

Restitución de HardenGet(MQLONG)

Para cada mensaje, se mantiene un recuento del número de veces que una llamada MQGET recupera el mensaje dentro de una unidad de trabajo, y dicha unidad de trabajo se restituye posteriormente.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Este recuento está disponible en el campo *BackoutCount* del descriptor de mensaje después de que se haya completado la llamada MQGET.

El recuento de restituciones de mensajes sobrevive a los reinicios del gestor de colas. Sin embargo, para asegurarse de que el recuento es preciso, la información debe *guardarse* (grabarse en disco u otro dispositivo de almacenamiento permanente) cada vez que una llamada MQGET recupera un mensaje dentro de una unidad de trabajo para esta cola. Si esto no se hace, el gestor de colas falla y la llamada MQGET se restituye, el recuento puede o no incrementarse.

Sin embargo, el endurecimiento de la información para cada llamada MQGET dentro de una unidad de trabajo impone un coste de proceso adicional, por lo tanto, establezca el atributo *HardenGetBackout* en MQQA_BACKOUT_HARAPARTADO sólo si es esencial que el recuento sea preciso.

En los sistemas IBM i, UNIX y Windows, el recuento de restituciones de mensajes siempre está protegido, independientemente del valor de este atributo.

Son posibles los siguientes valores:

MQQA_BACKOUT_HARDENED
El refuerzo se utiliza para asegurarse de que el recuento de restituciones para los mensajes de esta cola es preciso.

MQQA_BACKOUT_NOT_HARTIZADO
El refuerzo no se utiliza para asegurarse de que el recuento de restituciones para los mensajes de esta cola es preciso. Por lo tanto, el recuento puede ser menor de lo que debería ser.

Para determinar el valor de este atributo, utilice el selector MQIA_HARDEN_GET_BACKOUT con la llamada MQINQ.

IndexType (MQLONG)

Especifica el tipo de índice que el gestor de colas mantiene para los mensajes de la cola.

Local	Modelo	Alias	Remotos	Clúster
X	X			

El tipo de índice necesario depende de cómo la aplicación recupera los mensajes y de si la cola es una cola compartida o no compartida (consulte Atributo QSGDisp). Los valores siguientes son posibles para *IndexType*:

MQIT_NONE
El gestor de colas no mantiene ningún índice para esta cola. Utilice este valor para las colas que normalmente se procesan secuencialmente, es decir, sin utilizar ningún criterio de selección en la llamada MQGET.

ID_MSG_MQIT
El gestor de colas mantiene un índice que utiliza los identificadores de mensaje de los mensajes de la cola. Utilice este valor en las colas en las que la aplicación normalmente recupera mensajes utilizando el identificador de mensaje como criterio de selección en la llamada MQGET.

ID_CORREL_MQIT
El gestor de colas mantiene un índice que utiliza los identificadores de correlación de los mensajes de la cola. Utilice este valor para las colas en las que la aplicación normalmente recupera mensajes utilizando el identificador de correlación como criterio de selección en la llamada MQGET.

MQIT_MSG_TOKEN
El gestor de colas mantiene un índice que utiliza las señales de mensaje de los mensajes de la cola para su uso con las funciones del gestor de carga de trabajo (WLM) de z/OS.

Debe especificar esta opción para colas gestionadas por WLM; no la especifique para ningún otro tipo de cola. Además, no utilice este valor para una cola en la que una aplicación no utiliza las funciones del gestor de carga de trabajo de z/OS , sino que está recuperando mensajes utilizando la señal de mensaje como criterio de selección en la llamada MQGET.

ID_grupo_MQIT
El gestor de colas mantiene un índice que utiliza los identificadores de grupo de los mensajes de la cola. Este valor *debe* utilizarse para las colas en las que la aplicación recupera mensajes utilizando la opción MQGMO_LOGICAL_ORDER en la llamada MQGET.

Una cola con este tipo de índice no puede ser una cola de transmisión. Debe definirse una cola compartida con este tipo de índice para correlacionar con un objeto CFSTRUCT en CFLEVEL (3) o CFLEVEL (4).

Nota:

1. El orden físico de los mensajes en una cola con el tipo de índice MQIT_GROUP_ID no está definido, ya que la cola está optimizada para la recuperación eficaz de mensajes utilizando la opción

MQGMO_LOGICAL_ORDER en la llamada MQGET. Esto significa que el orden físico de los mensajes no suele ser el orden en el que los mensajes han llegado a la cola.

2. Si una cola MQIT_GROUP_ID tiene un *MsgDeliverySequence* de MQMDS_PRIORITY, el gestor de colas utiliza las prioridades de mensajes 0 y 1 para optimizar la recuperación de mensajes en orden lógico. Como resultado, el primer mensaje de un grupo no debe tener una prioridad de cero o uno; si lo tiene, el mensaje se procesa como si tuviera una prioridad de dos. El campo *Priority* de la estructura MQMD no se modifica.

Para obtener más información sobre los grupos de mensajes, consulte la descripción de las opciones de grupo y segmento en el campo MQGMO-Opciones.

El tipo de índice que se debe utilizar en varios casos se muestra en Tabla 574 en la página 832 y Tabla 575 en la página 833.

Tabla 574. Valores sugeridos o necesarios del tipo de índice de cola cuando no se especifica MQGMO_LOGICAL_ORDER

Criterios de selección en la llamada MQGET	**Tipo de índice para cola no compartida**	**Tipo de índice para cola compartida**
Ninguna	Cualquiera	Cualquiera
Selección utilizando un identificador:		
Identificador del mensaje	MQIT_MSG_ID sugerido	MQIT_NONE o MQIT_MSG_ID necesarios; MQIT_MSG_ID sugerido
Identificador de correlación	MQIT_CORREL_ID sugerido	MQIT_CORREL_ID necesario
Identificador de grupo	MQIT_GROUP_ID sugerido	MQIT_GROUP_ID necesario
Selección utilizando dos identificadores:		
Identificador de mensaje más identificador de correlación	MQIT_MSG_ID o MQIT_CORREL_ID sugeridos	MQIT_NONE o MQIT_MSG_ID o MQIT_CORREL_ID necesarios (Para mayor eficacia, se sugiere que el tipo de índice se elija para que coincida con el campo MQMD que tendrá las claves más diferenciadas)
Identificador de mensaje más identificador de grupo	MQIT_MSG_ID o MQIT_GROUP_ID sugeridos	No soportado
Identificador de correlación más identificador de grupo	MQIT_CORREL_ID o MQIT_GROUP_ID sugeridos	No soportado
Selección utilizando tres identificadores:		
Identificador de mensaje más identificador de correlación más identificador de grupo	MQIT_MSG_ID o MQIT_CORREL_ID o MQIT_GROUP_ID sugeridos	No soportado
Selección utilizando criterios relacionados con grupos:		
Identificador de grupo más número de secuencia de mensaje	MQIT_GROUP_ID necesario	MQIT_GROUP_ID necesario
Número de secuencia de mensaje (debe ser 1)	MQIT_GROUP_ID necesario	MQIT_GROUP_ID necesario

Tabla 574. Valores sugeridos o necesarios del tipo de índice de cola cuando no se especifica MQGMO_LOGICAL_ORDER (continuación)

Criterios de selección en la llamada MQGET	Tipo de índice para cola no compartida	Tipo de índice para cola compartida
Selección utilizando señal de mensaje:		
Señal de mensaje para uso de la aplicación	No utilizar MQIT_MSG_TOKEN	
Señal de mensaje para uso de WLM	MQIT_MSG_TOKEN necesario	No soportado

Tabla 575. Valores sugeridos o necesarios del tipo de índice de cola cuando se especifica MQGMO_LOGICAL_ORDER

Criterios de selección en la llamada MQGET	Tipo de índice para cola no compartida	Tipo de índice para cola compartida
Ninguna	MQIT_GROUP_ID necesario	MQIT_GROUP_ID necesario
Selección utilizando un identificador:		
Identificador del mensaje	MQIT_GROUP_ID necesario	No soportado
Identificador de correlación	MQIT_GROUP_ID necesario	No soportado
Identificador de grupo	MQIT_GROUP_ID necesario	MQIT_GROUP_ID necesario
Selección utilizando dos identificadores:		
Identificador de mensaje más identificador de correlación	MQIT_GROUP_ID necesario	No soportado
Identificador de mensaje más identificador de grupo	MQIT_GROUP_ID necesario	No soportado
Identificador de correlación más identificador de grupo	MQIT_GROUP_ID necesario	No soportado
Selección utilizando tres identificadores:		
Identificador de mensaje más identificador de correlación más identificador de grupo	MQIT_GROUP_ID necesario	No soportado

Para determinar el valor de este atributo, utilice el selector MQIA_INDEX_TYPE con la llamada MQINQ.

Este atributo sólo está soportado en z/OS.

InhibitGet (MQLONG)

Esto controla si se permiten operaciones get para esta cola.

Local	Modelo	Alias	Remotos	Clúster
X	X	X		

Si la cola es una cola alias, las operaciones get deben estar permitidas tanto para el alias como para la cola base en el momento de la operación get, para que la llamada MQGET sea satisfactoria. El valor puede ser uno de los siguientes:

MQQA_GET_INHIBITED
Las operaciones de obtención están inhibidas.

Las llamadas MQGET fallan con el código de razón MQRC_GET_inhibiITED. Esto incluye las llamadas MQGET que especifican MQGMO_BROWSE_FIRST o MQGMO_BROWSE_NEXT.

Nota: Si una llamada MQGET que opera dentro de una unidad de trabajo se completa correctamente, el cambio del valor del atributo *InhibitGet* posteriormente a MQQA_GET_inhibiITED no impide que se confirme la unidad de trabajo.

MQQA_GET_ALLOWED
Las operaciones de obtención están permitidas.

Para determinar el valor de este atributo, utilice el selector MQIA_INHIBID_GET con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

InhibitPut (MQLONG)
Esto controla si se permiten las operaciones de colocación para esta cola.

Local	Modelo	Alias	Remotos	Clúster
X	X	X	X	X

Si hay más de una definición en la vía de acceso de resolución de nombre de cola, se deben permitir operaciones de colocación para *cada* definición de la vía de acceso (incluidas las definiciones de alias de gestor de colas) en el momento de la operación de colocación, para que la llamada MQPUT o MQPUT1 sea satisfactoria. El valor puede ser uno de los siguientes:

MQQA_PUT_INHIBITED
Las operaciones de colocación están inhibidas.

Las llamadas MQPUT y MQPUT1 fallan con el código de razón MQRC_PUT_inhibiITED.

Nota: Si una llamada MQPUT que opera dentro de una unidad de trabajo se completa correctamente, el cambio del valor del atributo *InhibitPut* posteriormente a MQQA_PUT_inhibiITED no impide que se confirme la unidad de trabajo.

MQQA_PUT_ALLOWED
Las operaciones de colocación están permitidas.

Para determinar el valor de este atributo, utilice el selector MQIA_INHIBID_PUT con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

InitiationQName (MQCHAR48)
Es el nombre de una cola definida en el gestor de colas local; la cola debe ser de tipo MQQT_LOCAL.

Local	Modelo	Alias	Remotos	Clúster
X				

El gestor de colas envía un mensaje desencadenante a la cola de inicio cuando es necesario iniciar la aplicación como resultado de un mensaje que llega a la cola a la que pertenece este atributo. La cola de inicio debe ser supervisada por una aplicación de supervisor desencadenante que inicie la aplicación adecuada después de recibir el mensaje desencadenante.

Para determinar el valor de este atributo, utilice el selector MQCA_INITIATION_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

MaxMsgLongitud (MQLONG)
Este es un límite superior para la longitud del mensaje *físico* más largo que se puede colocar en la cola.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Sin embargo, debido a que el atributo de cola *MaxMsgLength* se puede establecer independientemente del atributo de gestor de colas *MaxMsgLength* , el límite superior real para la longitud del mensaje físico más largo que se puede colocar en la cola es el menor de estos dos valores.

Si el gestor de colas da soporte a la segmentación, es posible que una aplicación coloque un mensaje *lógico* que sea más largo que el menor de los dos atributos *MaxMsgLength* , pero sólo si la aplicación especifica el distintivo MQMF_SEGMENTATION_ALLOWED en MQMD. Si se especifica ese distintivo, el límite superior para la longitud de un mensaje lógico es de 999.999.999 bytes, pero normalmente las restricciones de recursos impuestas por el sistema operativo, o por el entorno en el que se ejecuta la aplicación, dan como resultado un límite inferior.

Un intento de colocar en la cola un mensaje demasiado largo falla con uno de los siguientes códigos de razón:

- MQRC_MSG_TOO_BIG_FOR_Q si el mensaje es demasiado grande para la cola
- MQRC_MSG_TOO_BIG_FOR_Q_MGR si el mensaje es demasiado grande para el gestor de colas, pero no demasiado grande para la cola

El límite inferior para el atributo *MaxMsgLength* es cero; el límite superior es de 100 MB (104 857 600 bytes).

Para obtener más información, consulte MQPUT-Parámetro BufferLength.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_MSG_LENGTH con la llamada MQINQ.

MaxQDepth (MQLONG)

Es el límite superior definido para el número de mensajes físicos que pueden existir en la cola en cualquier momento.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Un intento de colocar un mensaje en una cola que ya contiene mensajes *MaxQDepth* falla con el código de razón MQRC_Q_FULL.

El proceso de unidad de trabajo y la segmentación de mensajes pueden hacer que el número real de mensajes físicos en la cola supere *MaxQDepth*. Sin embargo, esto no afecta a la capacidad de recuperación de los mensajes; *todos* los mensajes de la cola se pueden recuperar utilizando la llamada MQGET.

El valor de este atributo es cero o mayor. El límite superior lo determina el entorno:

- En AIX, HP-UX, z/OS, Solaris, Linuxy Windows, el valor no puede exceder de 999 999 999.
- En IBM i, el valor no puede superar los 640 000.

Nota: El espacio de almacenamiento disponible para la cola puede agotarse incluso si hay menos de *MaxQDepth* mensajes en la cola.

Para determinar el valor de este atributo, utilice el selector MQIA_MAX_Q_DEPTH con la llamada MQINQ.

Secuencia MsgDelivery(MQLONG)

Local	Modelo	Alias	Remotos	Clúster
X	X			

Esto determina el orden en el que la llamada MQGET devuelve mensajes a la aplicación:

MQMDS_FIFO

Los mensajes se devuelven en orden FIFO (primero en entrar, primero en salir).

Una llamada MQGET devuelve el *primer* mensaje que cumple los criterios de selección especificados en la llamada, independientemente de la prioridad del mensaje.

MQMDS_PRIORITY
Los mensajes se devuelven en orden de prioridad.

Una llamada MQGET devuelve el mensaje de *prioridad más alta* que cumple los criterios de selección especificados en la llamada. Dentro de cada nivel de prioridad, los mensajes se devuelven en orden FIFO (primero en entrar, primero en salir).

- En z/OS, si la cola tiene un *IndexType* de MQIT_GROUP_ID, el atributo *MsgDeliverySequence* especifica el orden en el que se devuelven los grupos de mensajes a la aplicación. La secuencia particular en la que se devuelven los grupos viene determinada por la posición o prioridad del primer mensaje de cada grupo. El orden físico de los mensajes en la cola no está definido, ya que la cola está optimizada para una recuperación eficaz de mensajes utilizando la opción MQGMO_LOGICAL_ORDER en la llamada MQGET.
- En z/OS, si *IndexType* es MQIT_GROUP_ID y *MsgDeliverySequence* es MQMDS_PRIORITY, el gestor de colas utiliza las prioridades de mensajes cero y una para optimizar la recuperación de mensajes en orden lógico. Como resultado, el primer mensaje de un grupo no debe tener una prioridad de cero o uno; si lo tiene, el mensaje se procesa como si tuviera una prioridad de dos. El campo *Priority* de la estructura MQMD no se modifica.

Si los atributos relevantes se cambian mientras hay mensajes en la cola, la secuencia de entrega es la siguiente:

- El orden en el que la llamada MQGET devuelve los mensajes viene determinado por los valores de los atributos *MsgDeliverySequence* y *DefPriority* en vigor para la cola en el momento en que el mensaje llega a la cola:
  - Si *MsgDeliverySequence* es MQMDS_FIFO cuando llega el mensaje, el mensaje se coloca en la cola como si su prioridad fuera *DefPriority*. Esto no afecta al valor del campo *Priority* en el descriptor de mensaje del mensaje; dicho campo conserva el valor que tenía cuando se colocó el mensaje por primera vez.
  - Si *MsgDeliverySequence* es MQMDS_PRIORITY cuando llega el mensaje, el mensaje se coloca en la cola en el lugar adecuado a la prioridad proporcionada por el campo *Priority* en el descriptor de mensaje.

  Si el valor del atributo *MsgDeliverySequence* cambia mientras hay mensajes en la cola, el orden de los mensajes en la cola no cambia.

  Si el valor del atributo *DefPriority* se cambia mientras hay mensajes en la cola, los mensajes no se entregan necesariamente en orden FIFO, aunque el atributo *MsgDeliverySequence* se establezca en MQMDS_FIFO; los que se colocaron en la cola con la prioridad más alta se entregan primero.

Para determinar el valor de este atributo, utilice el selector MQIA_MSG_DELIVERY_SEQUENCE con la llamada MQINQ.

NonPersistentMessageClass (MQLONG)
El objetivo de fiabilidad para los mensajes no persistentes.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Especifica las circunstancias en las que se descartan los mensajes no persistentes colocados en esta cola:

MQNPM_CLASS_NORMAL
Los mensajes no persistentes están limitados al tiempo de vida de la sesión del gestor de colas; los mensajes se descartan en el caso de un reinicio del gestor de colas. Sólo es válido para colas no compartidas y es el valor predeterminado.

MQNPM_CLASS_HIGH
El gestor de colas intenta retener los mensajes no persistentes durante el tiempo de vida de la cola. Es posible que los mensajes no persistentes se pierdan si se produce una anomalía. Este valor se aplica a las colas compartidas.

Para determinar el valor de este atributo, utilice el selector MQIA_NPM_CLASS con la llamada MQINQ.

Recuento de OpenInput(MQLONG)
Es el número de descriptores de contexto que son válidos actualmente para eliminar mensajes de la cola mediante la llamada MQGET.

Local	Modelo	Alias	Remotos	Clúster
X				

Es el número total de manejadores de este tipo conocidos por el gestor de colas *local* . Si la cola es una cola compartida, el recuento no incluye las aperturas para la entrada que se han realizado para la cola en otros gestores de colas del grupo de compartición de colas al que pertenece el gestor de colas local.

El recuento incluye los descriptores de contexto en los que se ha abierto una cola alias que se resuelve en esta cola para entrada. El recuento no incluye los descriptores de contexto en los que se ha abierto la cola para acciones que no han incluido la entrada (por ejemplo, una cola abierta sólo para examinar).

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_OPEN_INPUT_COUNT con la llamada MQINQ.

Recuento de OpenOutput(MQLONG)
Es el número de descriptores de contexto que son válidos actualmente para añadir mensajes a la cola mediante la llamada MQPUT.

Local	Modelo	Alias	Remotos	Clúster
X				

Es el número total de manejadores de este tipo conocidos por el gestor de colas *local* ; no incluye las aperturas para salida que se han realizado para esta cola en los gestores de colas remotos. Si la cola es una cola compartida, el recuento no incluye las aperturas para la salida que se han realizado para la cola en otros gestores de colas del grupo de compartición de colas al que pertenece el gestor de colas local.

El recuento incluye los descriptores de contexto en los que se ha abierto una cola alias que se resuelve en esta cola para salida. El recuento no incluye los descriptores de contexto en los que se ha abierto la cola para acciones que no han incluido la salida (por ejemplo, una cola abierta sólo para consulta).

El valor de este atributo fluctúa a medida que opera el gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQIA_OPEN_OUTPUT_COUNT con la llamada MQINQ.

ProcessName (MQCHAR48)
Es el nombre de un objeto de proceso definido en el gestor de colas local. El objeto de proceso identifica un programa que puede dar servicio a la cola.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Para determinar el valor de este atributo, utilice el selector MQCA_PROCESS_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_PROCESS_NAME_LENGTH.

PropertyControl (MQLONG)

Especifica cómo se manejan las propiedades de mensaje para los mensajes que se recuperan de las colas utilizando la llamada MQGET con la opción MQGMO_PROPERTIES_AS_Q_DEF.

Local	Modelo	Alias	Remotos	Clúster
X	X	X		

El valor puede ser uno de los siguientes:

MQPROP_ALL
Todas las propiedades del mensaje se incluyen con el mensaje cuando se entrega a la aplicación. Las propiedades, excepto las que se encuentran en el descriptor de mensaje (o extensión), se colocan en una o más cabeceras MQRFH2 en los datos del mensaje. Si se proporciona un descriptor de mensaje, el comportamiento es devolver las propiedades en el descriptor de mensaje.

COMPATIBILIDAD de MQPROP_COMPATIBILITY
Si el mensaje contiene una propiedad con el prefijo mcd., jms., usr. o mqext., todas las propiedades de mensaje se entregan a la aplicación en una cabecera MQRFH2 . De lo contrario, todas las propiedades del mensaje, excepto las que se encuentran en el descriptor de mensaje (o extensión), se descartan y dejan de estar accesibles para la aplicación. Éste es el valor predeterminado; permite que las aplicaciones que esperan que las propiedades relacionadas con JMS estén en una cabecera MQRFH2 en los datos del mensaje, sigan funcionando sin ningún cambio. Si se proporciona un descriptor de mensaje, el comportamiento es devolver las propiedades en el descriptor de mensaje.

MQPROP_FORCE_MQRFH2
Las propiedades siempre se devuelven en los datos de mensaje en una cabecera MQRFH2 independientemente de si la aplicación especifica un manejador de mensajes. Se ignora un manejador de mensajes válido proporcionado en el campo MsgHandle de la estructura MQGMO en la llamada MQGET. Las propiedades del mensaje no son accesibles a través del manejador de mensajes.

MQPROP_NONE
Todas las propiedades del mensaje, excepto las del descriptor de mensaje (o extensión), se eliminan del mensaje antes de que el mensaje se entregue a la aplicación. Si se proporciona un descriptor de mensaje, el comportamiento es devolver las propiedades en el descriptor de mensaje.

Este parámetro es aplicable a las colas Local, Alias y Modelo. Para determinar su valor, utilice el selector MQIA_PROPERTY_CONTROL con la llamada MQINQ.

Suceso QDepthHigh(MQLONG)

Esto controla si se generan sucesos de profundidad de cola alta.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Un suceso de profundidad de cola alta indica que una aplicación ha colocado un mensaje en una cola y esto ha hecho que el número de mensajes de la cola sea mayor o igual que el umbral de profundidad de cola alta (consulte el atributo *QDepthHighLimit*).

Nota: El valor de este atributo puede cambiar dinámicamente.

El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos .

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_HIGH_EVENT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

Límite QDepthHigh(MQLONG)

Es el umbral con el que se compara la profundidad de cola para generar un suceso Profundidad de cola alta.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Este suceso indica que una aplicación ha colocado un mensaje en una cola y que esto ha hecho que el número de mensajes en la cola sea mayor o igual que el umbral alto de profundidad de cola. Consulte Atributo de sucesoQDepthHigh.

El valor se expresa como un porcentaje de la profundidad de cola máxima (atributo*MaxQDepth*), y es mayor o igual que 0 y menor o igual que 100. El valor predeterminado es 80.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_HIGH_LIMIT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

Suceso QDepthLow(MQLONG)

Esto controla si se generan sucesos de profundidad de cola baja.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Un suceso Profundidad de cola baja indica que una aplicación ha recuperado un mensaje de una cola y que esto ha hecho que el número de mensajes de la cola sea menor o igual que el umbral de profundidad de cola baja (consulte atributo LímiteQDepthLow).

Nota: El valor de este atributo puede cambiar dinámicamente.

El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos .

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_LOW_EVENT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

Límite QDepthLow(MQLONG)

Es el umbral con el que se compara la profundidad de cola para generar un suceso Profundidad de cola baja.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Este suceso indica que una aplicación ha recuperado un mensaje de una cola y que esto ha hecho que el número de mensajes de la cola sea menor o igual que el umbral bajo de profundidad de cola. Consulte Atributo de sucesoQDepthLow.

El valor se expresa como un porcentaje de la profundidad de cola máxima (atributo*MaxQDepth*), y es mayor o igual que 0 y menor o igual que 100. El valor predeterminado es de 20.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_LOW_LIMIT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

Suceso QDepthMax(MQLONG)

Esto controla si se generan sucesos de cola llena. Un suceso Cola llena indica que una colocación en una cola se ha rechazado porque la cola está llena, es decir, la profundidad de cola ya ha alcanzado su valor máximo.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Nota: El valor de este atributo puede cambiar dinámicamente.

El valor puede ser uno de los siguientes:

MQEVR_DISABLED
Informes de sucesos inhabilitados.

MQEVR_ENABLED
Informes de sucesos habilitados.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos .

Para determinar el valor de este atributo, utilice el selector MQIA_Q_DEPTH_MAX_EVENT con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

QDesc (MQCHAR64)

Utilice este campo para comentarios descriptivos.

Local	Modelo	Alias	Remotos	Clúster
X	X	X	X	X

El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, el campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas *CodedCharSetId*), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_DESC con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_DESC_LENGTH.

QName (MQCHAR48)

Es el nombre de una cola definida en el gestor de colas local.

Local	Modelo	Alias	Remotos	Clúster
X		X	X	X

Todas las colas definidas en un gestor de colas comparten el mismo espacio de nombres de cola. Por lo tanto, una cola MQQT_LOCAL y una cola MQQT_ALIAS no pueden tener el mismo nombre.

Para determinar el valor de este atributo, utilice el selector MQCA_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

QServiceInterval (MQLONG)

Este es el intervalo de servicio utilizado para la comparación para generar sucesos de intervalo de servicio alto y de intervalo de servicio correcto.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Consulte Atributo de sucesoQServiceInterval.

El valor está en unidades de milisegundos, y es mayor o igual que cero, y menor o igual que 999 999 999 999.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_SERVICE_INTERVAL con la llamada MQINQ.

Este atributo está soportado en z/OS, pero la llamada MQINQ no se puede utilizar para determinar su valor.

QServiceIntervalEvent (MQLONG)

Esto controla si se generan sucesos de intervalo de servicio alto o de intervalo de servicio correcto.

Local	Modelo	Alias	Remotos	Clúster
X	X			

- Se genera un suceso de intervalo de servicio alto cuando una comprobación indica que no se ha recuperado ningún mensaje de la cola durante al menos el tiempo indicado por el atributo *QServiceInterval* .
- Se genera un suceso de intervalo de servicio correcto cuando una comprobación indica que los mensajes se han recuperado de la cola dentro del tiempo indicado por el atributo *QServiceInterval* .

Nota: El valor de este atributo puede cambiar dinámicamente.

El valor puede ser uno de los siguientes:

MQQSIE_HIGH
Sucesos de intervalo de servicio de cola alto habilitados.

- Los sucesos de intervalo de servicio de cola alto están **habilitados** y
- Los sucesos de intervalo de servicio de cola correcto están **inhabilitados**.

MQQSIE_OK
Sucesos de intervalo de servicio de cola correcto habilitados.

- Los sucesos de intervalo de servicio de cola alto están **inhabilitados** y
- Los sucesos de intervalo de servicio de cola correcto están **habilitados**.

MQQSIE_NONE
No hay sucesos de intervalo de servicio de cola habilitados.

- Los sucesos de intervalo de servicio de cola alto están **inhabilitados** y
- Los sucesos de intervalo de servicio de cola correcto también están **inhabilitados**.

Para las colas compartidas, el valor de este atributo se ignora; se asume el valor MQQSIE_NONE.

Para obtener más información sobre los sucesos, consulte Supervisión de sucesos .

Para determinar el valor de este atributo, utilice el selector MQIA_Q_SERVICE_INTERVAL_EVENT con la llamada MQINQ.

En z/OS, no puede utilizar la llamada MQINQ para determinar el valor de este atributo.

QSGDisp (MQLONG)

Especifica la disposición de la cola.

Local	Modelo	Alias	Remotos	Clúster
X		X	X	

El valor puede ser uno de los siguientes:

MQQSGD_Q_MGR
El objeto tiene una disposición de gestor de colas. Esto significa que la definición de objeto sólo es conocida por el gestor de colas local; la definición no es conocida por otros gestores de colas del grupo de compartición de colas.

Cada gestor de colas del grupo de compartición de colas puede tener un objeto con el mismo nombre y tipo que el objeto actual, pero estos son objetos separados y no hay ninguna correlación entre ellos. Sus atributos no están restringidos a ser iguales entre sí.

MQQSGD_COPY
El objeto es una copia local de una definición de objeto maestro que existe en el repositorio compartido. Cada gestor de colas del grupo de compartición de colas puede tener su propia copia del objeto. Inicialmente, todas las copias tienen los mismos atributos, pero utilizando mandatos MQSC, puede modificar cada copia para que sus atributos difieran de los de las otras copias. Los atributos de las copias se resincronizan cuando se modifica la definición maestra en el repositorio compartido.

MQQSGD_SHARED
El objeto tiene una disposición compartida. Esto significa que existe en el repositorio compartido una única instancia del objeto que es conocida por todos los gestores de colas del grupo de compartición de colas. Cuando un gestor de colas del grupo accede al objeto, accede a la única instancia compartida del objeto.

Para determinar el valor de este atributo, utilice el selector MQIA_QSG_DISP con la llamada MQINQ.

Este atributo sólo está soportado en z/OS.

QueueAccounting (MQLONG)

Local	Modelo	Alias	Remotos	Clúster
X	X	X	X	

Esto controla la recopilación de datos de contabilidad para la cola. Para que los datos de contabilidad se recopilen para esta cola, los datos de contabilidad para esta conexión también deben estar habilitados, utilizando el atributo ACCTQ de QMGR o el campo Opciones de la estructura MQCNO en la llamada MQCONNX.

Este atributo tiene uno de los siguientes valores:

MQMON_Q_MGR
Los datos de contabilidad para esta cola se recopilan basándose en el valor del atributo ACCTQ de QMGR. Este es el valor predeterminado.

MQMON_OFF
No recopile datos de contabilidad para esta cola.

MQMON_ON
Recopilar datos de contabilidad para esta cola.

Para determinar el valor de este atributo, utilice el selector MQIA_ACCOUNTING_Q con la llamada MQINQ.

QueueMonitoring (MQLONG)

Controla la recopilación de los datos de supervisión para las colas.

Local	Modelo	Alias	Remotos	Clúster
X	X			

El valor puede ser uno de los siguientes:

MQMON_Q_MGR
Recopile datos de supervisión según el valor del atributo de gestor de colas *QueueMonitoring* . Este es el valor predeterminado.

MQMON_OFF
La recopilación de datos de supervisión en línea está desactivada para esta cola.

MQMON_LOW
Si el valor del atributo de gestor de colas *QueueMonitoring* no es MQMON_NONE, se activa la recopilación de datos de supervisión en línea, con una tasa baja de recopilación de datos para esta cola.

MQMON_MEDIO
Si el valor del atributo de gestor de colas *QueueMonitoring* no es MQMON_NONE, la recopilación de datos de supervisión en línea está activada, con una tasa moderada de recopilación de datos para esta cola.

MQMON_HIGH
Si el valor del atributo de gestor de colas *QueueMonitoring* no es MQMON_NONE, se activa la recopilación de datos de supervisión en línea, con una alta tasa de recopilación de datos para esta cola.

Para determinar el valor de este atributo, utilice el selector MQIA_MONITORING_Q con la llamada MQINQ.

QueueStatistics (MQCHAR12)

Local	Modelo	Alias	Remotos	Clúster
X	X	X	X	

Esto controla la recopilación de datos estadísticos para la cola.

Este atributo tiene uno de los siguientes valores:

MQMON_Q_MGR
Los datos de contabilidad para esta cola se recopilan basándose en el valor del atributo STATQ de QMGR. Este es el valor predeterminado.

MQMON_OFF
Desactive la recopilación de datos de estadísticas para esta cola.

MQMON_ON
Active la recopilación de datos de estadísticas para esta cola.

Tipo de cola (MQLONG)

Local	Modelo	Alias	Remotos	Clúster
X		X	X	X

Este es el tipo de cola; tiene uno de los valores siguientes:

MQQT_ALIAS
Definición de cola alias.

MQQT_CLUSTER
Cola de clúster.

MQQT_LOCAL
Cola local.

MQQT_REMOTE
Definición local de una cola remota.

Para determinar el valor de este atributo, utilice el selector MQIA_Q_TYPE con la llamada MQINQ.

RemoteQMgrNombre (MQCHAR48)

Local	Modelo	Alias	Remotos	Clúster
			X	

Es el nombre del gestor de colas remoto en el que se define la cola *RemoteQName* . Si la cola *RemoteQName* tiene un valor *QSGDisp* de MQQSGD_COPY o MQQSGD_SHARED, *RemoteQMgrName* puede ser el nombre del grupo de compartición de colas que es propietario de *RemoteQName*.

Si una aplicación abre la definición local de una cola remota, *RemoteQMgrName* no debe estar en blanco y no debe ser el nombre del gestor de colas local. Si *XmitQName* está en blanco, se utiliza la cola local con el mismo nombre que *RemoteQMgrName* como cola de transmisión. Si no hay ninguna cola con el nombre *RemoteQMgrName*, se utiliza la cola identificada por el atributo de gestor de colas *DefXmitQName* .

Si esta definición se utiliza para un alias de gestor de colas, *RemoteQMgrName* es el nombre del gestor de colas que se está alias. Puede ser el nombre del gestor de colas local. De lo contrario, si *XmitQName* está en blanco cuando se produce la apertura, debe haber una cola local con un nombre que sea el mismo que *RemoteQMgrName*; esta cola se utiliza como cola de transmisión.

Si esta definición se utiliza para un alias de respuesta, este nombre es el nombre del gestor de colas que debe ser *ReplyToQMgr*.

Nota: No se realiza ninguna validación en el valor especificado para este atributo cuando se crea o modifica la definición de cola.

Para determinar el valor de este atributo, utilice el selector MQCA_REMOTE_Q_MGR_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_MGR_NAME_LENGTH.

RemoteQName (MQCHAR48)

Local	Modelo	Alias	Remotos	Clúster
			X	

Es el nombre de la cola tal como se conoce en el gestor de colas remoto *RemoteQMgrName*.

Si una aplicación abre la definición local de una cola remota, cuando se produce la apertura, *RemoteQName* no debe estar en blanco.

Si esta definición se utiliza para una definición de alias de gestor de colas, cuando se produzca la apertura, *RemoteQName* debe estar en blanco.

Si la definición se utiliza para un alias de respuesta, este nombre es el nombre de la cola que va a ser *ReplyToQ*.

Nota: No se realiza ninguna validación en el valor especificado para este atributo cuando se crea o modifica la definición de cola.

Para determinar el valor de este atributo, utilice el selector MQCA_REMOTE_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

RetentionInterval (MQLONG)

Es el periodo de tiempo durante el cual se retiene la cola. Una vez transcurrido este tiempo, la cola es apta para su supresión.

Local	Modelo	Alias	Remotos	Clúster
X	X			

El tiempo se mide en horas, contando desde la fecha y hora en que se creó la cola. La fecha y hora de creación de la cola se registran en los atributos *CreationDate* y *CreationTime* .

Esta información se proporciona para permitir que una aplicación de mantenimiento o el operador identifique y suprima colas que ya no son necesarias.

Nota: El gestor de colas nunca realiza ninguna acción para suprimir colas basándose en este atributo, o para impedir la supresión de colas con un intervalo de retención que no ha caducado; es responsabilidad del usuario realizar cualquier acción necesaria.

Utilice un intervalo de retención realista para evitar la acumulación de colas dinámicas permanentes (consulte el atributo DefinitionType). Sin embargo, este atributo también se puede utilizar con colas predefinidas.

Para determinar el valor de este atributo, utilice el selector MQIA_RETENTION_INTERVAL con la llamada MQINQ.

Ámbito (MQLONG)

Esto controla si también existe una entrada para esta cola en un directorio de célula.

Local	Modelo	Alias	Remotos	Clúster
X		X	X	

Un servicio de nombres instalable proporciona un directorio de célula. El valor puede ser uno de los siguientes:

MQSCO_Q_MGR
La definición de cola tiene ámbito de gestor de colas: la definición de la cola no se extiende más allá del gestor de colas que la posee. Para abrir la cola para salida de algún otro gestor de colas, debe especificarse el nombre del gestor de colas propietario o el otro gestor de colas debe tener una definición local de la cola.

MQSCO_CELL
La definición de cola tiene ámbito de célula: la definición de cola también se coloca en un directorio de célula disponible para todos los gestores de colas de la célula. La cola se puede abrir para salida de cualquiera de los gestores de colas de la célula especificando el nombre de la cola; no es necesario especificar el nombre del gestor de colas propietario de la cola. Sin embargo, la definición de cola no está disponible para ningún gestor de colas de la célula que también tenga una definición local de una cola con ese nombre, ya que la definición local tiene prioridad.

Un servicio de nombres instalable proporciona un directorio de célula.

El modelo y las colas dinámicas no pueden tener ámbito de célula.

Este valor sólo es válido si se ha configurado un servicio de nombres que da soporte a un directorio de célula.

Para determinar el valor de este atributo, utilice el selector MQIA_SCOPE con la llamada MQINQ.

El soporte para este atributo está sujeto a las restricciones siguientes:

- En IBM i, el atributo está soportado, pero sólo MQSCO_Q_MGR es válido.
- En z/OS, el atributo no está soportado.

Compartibilidad (MQLONG)
Esto indica si la cola se puede abrir para entrada varias veces simultáneamente.

Local	Modelo	Alias	Remotos	Clúster
X	X			

El valor puede ser uno de los siguientes:

MQQA_SHAREABLE
La cola es compartible.

Se permiten varias aperturas con la opción MQOO_INPUT_SHARED.

MQQA_NOT_SHAREABLE
La cola no se puede compartir.

Una llamada MQOPEN con la opción MQOO_INPUT_SHARED se trata como MQOO_INPUT_EXCLUSIVE.

Para determinar el valor de este atributo, utilice el selector MQIA_SHAREABILITY con la llamada MQINQ.

StorageClass (MQCHAR8)
Es un nombre definido por el usuario que define el almacenamiento físico utilizado para contener la cola. En la práctica, un mensaje se graba en disco sólo si necesita paginarse fuera de su almacenamiento intermedio de memoria.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Para determinar el valor de este atributo, utilice el selector MQCA_STORAGE_CLASS con la llamada MQINQ. La longitud de este atributo la proporciona MQ_STORAGE_CLASS_LENGTH.

Este atributo sólo está soportado en z/OS.

TriggerControl (MQLONG)
Esto controla si los mensajes desencadenantes se graban en una cola de inicio para iniciar una aplicación para dar servicio a la cola.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Es uno de los siguientes:

MQTC_OFF
No se deben escribir mensajes desencadenantes para esta cola. El valor de *TriggerType* es irrelevante en este caso.

MQTC_ON
Los mensajes desencadenantes se escribirán para esta cola cuando se produzcan los sucesos desencadenantes adecuados.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_CONTROL con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

TriggerData (MQCHAR64)
Son datos de formato libre que el gestor de colas inserta en el mensaje desencadenante cuando un mensaje que llega a esta cola hace que se grabe un mensaje desencadenante en la cola de inicio.

Local	Modelo	Alias	Remotos	Clúster
X	X			

El contenido de estos datos no es significativo para el gestor de colas. Es significativo para la aplicación de supervisor desencadenante que procesa la cola de inicio o para la aplicación que inicia el supervisor desencadenante.

La serie de caracteres no debe contener ningún valor nulo. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector MQCA_TRIGGER_DATA con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET. La longitud de este atributo la proporciona MQ_TRIGGER_DATA_LENGTH.

TriggerDepth (MQLONG)

Local	Modelo	Alias	Remotos	Clúster
X	X			

Es el número de mensajes de prioridad *TriggerMsgPriority* o superior que deben estar en la cola antes de que se escriba un mensaje desencadenante. Esto se aplica cuando *TriggerType* se establece en MQTT_DEPTH. El valor de *TriggerDepth* es uno o mayor. De lo contrario, este atributo no se utiliza.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_DEPTH con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

TriggerMsgPriority (MQLONG)

Esta es la prioridad de mensaje por debajo de la cual los mensajes no contribuyen a la generación de mensajes desencadenantes (es decir, el gestor de colas ignora estos mensajes al decidir si se genera un mensaje desencadenante).

Local	Modelo	Alias	Remotos	Clúster
X	X			

TriggerMsgPriority puede estar en el rango de cero (más bajo) a *MaxPriority* (más alto; consulte el atributo MaxPriority); un valor de cero hace que todos los mensajes contribuyan a la generación de mensajes desencadenantes.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_MSG_PRIORITY con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

TriggerType (MQLONG)

Esto controla las condiciones en las que se graban los mensajes desencadenantes como resultado de los mensajes que llegan a esta cola.

Local	Modelo	Alias	Remotos	Clúster
X	X			

Tiene uno de los siguientes valores:

MQTT_NONE
No se graban mensajes desencadenantes como resultado de los mensajes de esta cola. Esto tiene el mismo efecto que establecer *TriggerControl* en MQTC_OFF.

MQTT_FIRST
Se graba un mensaje desencadenante siempre que el número de mensajes de prioridad *TriggerMsgPriority* o superior en la cola cambia de 0 a 1.

MQTT_EVERY
Se graba un mensaje desencadenante siempre que llega a la cola un mensaje de prioridad *TriggerMsgPriority* o superior.

MQTT_DEPTH
Un mensaje desencadenante se graba siempre que el número de mensajes de prioridad *TriggerMsgPriority* o superior en la cola es igual o superior a *TriggerDepth*. Después de que se haya grabado el mensaje desencadenante, *TriggerControl* se establece en MQTC_OFF para evitar que se desencadene más hasta que se vuelva a activar explícitamente.

Para determinar el valor de este atributo, utilice el selector MQIA_TRIGGER_TYPE con la llamada MQINQ. Para cambiar el valor de este atributo, utilice la llamada MQSET.

Uso (MQLONG)
Indica para qué se utiliza la cola.

Local	Modelo	Alias	Remotos	Clúster
X	X			

El valor puede ser uno de los siguientes:

MQUS_NORMAL
Se trata de una cola que las aplicaciones utilizan al transferir y obtener mensajes; la cola no es una cola de transmisión.

MQUS_TRANSMISSION
Es una cola utilizada para contener mensajes destinados a gestores de colas remotos. Cuando una aplicación envía un mensaje a una cola remota, el gestor de colas local almacena el mensaje temporalmente en la cola de transmisión adecuada en un formato especial. A continuación, un agente de canal de mensajes lee el mensaje de la cola de transmisión y lo transporta al gestor de colas remoto. Para obtener más información sobre las colas de transmisión, consulte Definición de una cola de transmisión.

Sólo las aplicaciones con privilegios pueden abrir una cola de transmisión para que MQOO_OUTPUT coloque mensajes directamente en ella. Normalmente, sólo las aplicaciones de programa de utilidad lo hacen. Asegúrese de que el formato de datos del mensaje sea correcto (consulte "MQXQH-Cabecera de cola de transmisión" en la página 596) o que se puedan producir errores durante el proceso de transmisión. El contexto no se pasa ni se establece a menos que se especifique una de las opciones de contexto MQPMO_ * _CONTEXT.

Para determinar el valor de este atributo, utilice el selector MQIA_USAGE con la llamada MQINQ.

XmitQName (MQCHAR48)
Es el nombre de la cola de transmisión. Si este atributo no está en blanco cuando se produce una apertura, ya sea para una cola remota o para una definición de alias de gestor de colas, especifica el nombre de la cola de transmisión local que se utilizará para reenviar el mensaje.

Local	Modelo	Alias	Remotos	Clúster
			X	

Si *XmitQName* está en blanco, la cola local con un nombre que es el mismo que *RemoteQMgrName* se utiliza como cola de transmisión. Si no hay ninguna cola con el nombre *RemoteQMgrName*, se utiliza la cola identificada por el atributo de gestor de colas *DefXmitQName* .

Este atributo se ignora si la definición se está utilizando como alias de gestor de colas y *RemoteQMgrName* es el nombre del gestor de colas local. Este atributo también se ignora si la definición se utiliza como definición de alias de cola de respuestas.

Para determinar el valor de este atributo, utilice el selector MQCA_XMIT_Q_NAME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_Q_NAME_LENGTH.

Atributos de las listas de nombres

En la tabla siguiente se resumen los atributos específicos de las listas de nombres. Los atributos se describen en orden alfabético.

Las listas de nombres están soportadas en todos los sistemas WebSphere MQ , además de los clientes MQI de WebSphere MQ conectados a estos sistemas.

Nota: Los nombres de los atributos que se muestran en esta sección son nombres descriptivos utilizados con las llamadas MQINQ y MQSET; los nombres son los mismos que para los mandatos PCF. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte Mandatos de script (MQSC) para obtener más información.

Tabla 576. Atributos de las listas de nombres

Atributo	Descripción
AlterationDate	Fecha en que se modificó por última vez la definición
AlterationTime	Hora a la que se modificó por última vez la definición
NameCount	Número de nombres en la lista de nombres
NamelistDesc	Descripción de lista de nombres
NamelistName	Nombre de lista de nombres
Nombres	Una lista de nombres de *NameCount*
NamelistType	Tipo de lista de nombres
QSGDisp	Disposición del grupo de compartición de colas

AlterationDate (MQCHAR12)

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es YYYY-MM-DD, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Es la hora en que se modificó por última vez la definición. El formato de la hora es HH.MM.SS.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_TIME_LENGTH.

NameCount (MQLONG)

Es el número de nombres de la lista de nombres. Es mayor o igual que cero. Se define el valor siguiente:

MQNC_MAX_NAMELIST_NAME_COUNT
Número máximo de nombres en una lista de nombres.

Para determinar el valor de este atributo, utilice el selector MQIA_NAME_COUNT con la llamada MQINQ.

NamelistDesc (MQCHAR64)

Utilice este campo para comentarios descriptivos; su valor se establece mediante el proceso de definición. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, este campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas *CodedCharSetId*), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQCA_NAMELIST_DESC con la llamada MQINQ.

La longitud de este atributo la proporciona MQ_NAMELIST_DESC_LENGTH.

NamelistName (MQCHAR48)

Es el nombre de una lista de nombres definida en el gestor de colas local. Para obtener más información sobre los nombres de lista de nombres, consulte la sección Otros nombres de objeto .

Cada lista de nombres tiene un nombre que es diferente de los nombres de otras listas de nombres que pertenecen al gestor de colas, pero puede duplicar los nombres de otros objetos del gestor de colas de tipos diferentes (por ejemplo, colas).

Para determinar el valor de este atributo, utilice el selector MQCA_NAMELIST_NAME con la llamada MQINQ.

La longitud de este atributo la proporciona MQ_NAMELIST_NAME_LENGTH.

NamelistType (MQLONG)

Especifica la naturaleza de los nombres de la lista de nombres e indica cómo se utiliza la lista de nombres. Es uno de los valores siguientes:

MQNT_NONE
Lista de nombres sin tipo asignado.

MQNT_Q
Lista de nombres que contiene los nombres de las colas.

MQNT_CLUSTER
Lista de nombres que contiene los nombres de los clústeres.

MQNT_AUTH_INFO
Lista de nombres que contiene los nombres de los objetos de información de autenticación.

Para determinar el valor de este atributo, utilice el selector MQIA_NAMELIST_TYPE con la llamada MQINQ.

Este atributo sólo está soportado en z/OS.

Nombres (MQCHAR48xNameCount)

Esta es una lista de nombres de *NameCount* , donde cada nombre es el nombre de un objeto definido en el gestor de colas local. Para obtener más información sobre los nombres de objeto, consulte Reglas para la denominación de objetos de IBM WebSphere MQ .

Para determinar el valor de este atributo, utilice el selector MQCA_NAMES con la llamada MQINQ.

La longitud de cada nombre de la lista la proporciona MQ_OBJECT_NAME_LENGTH.

QSGDisp (MQLONG)

Especifica la disposición de la lista de nombres. El valor puede ser uno de los siguientes:

MQQSGD_Q_MGR
El objeto tiene disposición de gestor de colas: la definición de objeto sólo es conocida por el gestor de colas local; la definición no es conocida por otros gestores de colas del grupo de compartición de colas.

Cada gestor de colas del grupo de compartición de colas puede tener un objeto con el mismo nombre y tipo que el objeto actual, pero estos son objetos separados y no hay ninguna correlación entre ellos. Sus atributos no están restringidos a ser iguales entre sí.

MQQSGD_COPY
El objeto es una copia local de una definición de objeto maestro que existe en el repositorio compartido. Cada gestor de colas del grupo de compartición de colas puede tener su propia copia del objeto. Inicialmente, todas las copias tienen los mismos atributos, pero puede modificar cada copia,

utilizando mandatos MQSC, para que sus atributos difieran de los de las otras copias. Los atributos de las copias se resincronizan cuando se modifica la definición maestra en el repositorio compartido.

Para determinar el valor de este atributo, utilice el selector MQIA_QSG_DISP con la llamada MQINQ.

Este atributo sólo está soportado en z/OS.

Atributos de las definiciones de proceso

En la tabla siguiente se resumen los atributos que son específicos de las definiciones de proceso. Los atributos se describen en orden alfabético.

Nota: Los nombres de los atributos de esta sección son nombres descriptivos utilizados con las llamadas MQINQ y MQSET; los nombres son los mismos que para los mandatos PCF. Cuando se utilizan mandatos MQSC para definir, modificar o visualizar atributos, se utilizan nombres abreviados alternativos; consulte Mandatos de script (MQSC) para obtener más información.

Tabla 577. Atributos de las definiciones de proceso

Atributo	Descripción
AlterationDate	Fecha en que se modificó por última vez la definición
AlterationTime	Hora a la que se modificó por última vez la definición
ApplId	Identificador de aplicación
ApplType	Tipo de aplicación
EnvData	Datos de entorno
ProcessDesc	Descripción del proceso
ProcessName	Nombre de proceso
QSGDisp	Disposición del grupo de compartición de colas
UserData	Datos de usuario

AlterationDate (MQCHAR12)

Es la fecha en la que se modificó por última vez la definición. El formato de la fecha es YYYY-MM-DD, rellenado con dos espacios en blanco finales para que la longitud sea de 12 bytes.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_DATE con la llamada MQINQ. La longitud de este atributo la proporciona MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Es la hora en que se modificó por última vez la definición. El formato de la hora es HH.MM.SS.

Para determinar el valor de este atributo, utilice el selector MQCA_ALTERATION_TIME con la llamada MQINQ. La longitud de este atributo la proporciona MQ_TIME_LENGTH.

ApplId (MQCHAR256)

Es una serie de caracteres que identifica la aplicación que se va a iniciar. Esta información la utiliza una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *ApplId* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por WebSphere MQ requiere que *ApplId* sea el nombre de un programa ejecutable. Las notas siguientes se aplican a los entornos indicados:

- En z/OS, *ApplId* debe ser:
  - Un identificador de transacción CICS , para aplicaciones iniciadas utilizando la CKTI de transacción de supervisor desencadenante CICS
  - Un identificador de transacción de IMS , para aplicaciones iniciadas utilizando el supervisor desencadenante de IMS CSQQTRMN

- En sistemas Windows , el nombre de programa puede tener como prefijo una unidad y una vía de acceso de directorio.
- En sistemas UNIX , el nombre de programa puede tener como prefijo una vía de acceso de directorio.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector MQCA_APPL_ID con la llamada MQINQ. La longitud de este atributo la proporciona MQ_PROCESS_APPL_ID_LENGTH.

ApplType (MQLONG)

Identifica la naturaleza del programa que se va a iniciar en respuesta a la recepción de un mensaje desencadenante. Esta información la utiliza una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje desencadenante.

ApplType puede tener cualquier valor, pero se recomiendan los siguientes valores para los tipos estándar; restrinja los tipos de aplicación definidos por el usuario a los valores del rango MQAT_USER_FIRST a MQAT_USER_LAST:

MQAT_AIX
Aplicación AIX (el mismo valor que MQAT_UNIX).

MQAT_LOTE
aplicación por lotes

INTERMEDIARIO
Aplicación de intermediario

MQAT_CICS
Transacción CICS .

MQAT_CICS_BRIDGE
Aplicación puente CICS .

MQAT_CICS_VSE
Transacción CICS/VSE .

MQAT_DOS
WebSphere MQ Aplicación cliente MQI en PC DOS.

MQAT_IMS
Aplicación IMS .

MQAT_IMS_BRIDGE
IMS .

MQAT_JAVA
Aplicación Java.

MQAT_MVS
MVS o aplicación TSO (el mismo valor que MQAT_ZOS).

MQAT_NOTES_AGENT
Lotus Notes .

MQAT_NSK
Aplicación HP Integrity NonStop Server .

MQAT_OS390
OS/390 (mismo valor que MQAT_ZOS).

MQAT_OS400
Aplicación IBM i .

MQAT_RRS_BATCH
Aplicación por lotes RRS.

MQAT_UNIX
Aplicación UNIX .

MQAT_DESCONOCIDO
Aplicación de tipo desconocido.

USUARIO_MQ
Aplicación de usuario.

MQAT_VOS
Aplicación Stratus VOS.

MQAT_WINDOWS
Aplicación Windows de 16 bits.

MQAT_WINDOWS_NT
Aplicación Windows de 32 bits.

MQAT_WLM
Aplicación del gestor de carga de trabajo de z/OS .

MQAT_XCF
XCF.

MQAT_ZOS
Aplicación z/OS .

MQAT_USER_FIRST
Valor más bajo para el tipo de aplicación definido por el usuario.

MQAT_USER_LAST
Valor más alto para el tipo de aplicación definido por el usuario.

Para determinar el valor de este atributo, utilice el selector MQIA_APPL_TYPE con la llamada MQINQ.

EnvData (MQCHAR128)

Es una serie de caracteres que contiene información relacionada con el entorno perteneciente a la aplicación que se va a iniciar. Esta información la utiliza una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio; la información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *EnvData* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por WebSphere MQ añade *EnvData* a la lista de parámetros pasada a la aplicación iniciada. La lista de parámetros consta de la estructura MQTMC2 , seguida de un espacio en blanco, seguida de *EnvData* con los espacios en blanco finales eliminados. Las notas siguientes se aplican a los entornos indicados:

- En z/OS:
  - *EnvData* no lo utilizan las aplicaciones de supervisor desencadenante proporcionadas por WebSphere MQ.
  - Si ApplType es MQAT_WLM, puede proporcionar valores predeterminados en EnvData para los campos ServiceName y ServiceStep en la cabecera de información de trabajo (MQWIH).
- En sistemas UNIX , *EnvData* se puede establecer en el carácter & para ejecutar la aplicación iniciada en segundo plano.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario.

Para determinar el valor de este atributo, utilice el selector MQCA_ENV_DATA con la llamada MQINQ. La longitud de este atributo la proporciona MQ_PROCESS_ENV_DATA_LENGTH.

ProcessDesc (MQCHAR64)

Utilice este campo para comentarios descriptivos. El contenido del campo no es significativo para el gestor de colas, pero es posible que el gestor de colas requiera que el campo contenga sólo caracteres

que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, el campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como lo define el atributo de gestor de colas *CodedCharSetId*), estos caracteres podrían traducirse incorrectamente si este campo se envía a otro gestor de colas.

Para determinar el valor de este atributo, utilice el selector MQCA_PROCESS_DESC con la llamada MQINQ.

La longitud de este atributo la proporciona MQ_PROCESS_DESC_LENGTH.

ProcessName (MQCHAR48)

Es el nombre de una definición de proceso definida en el gestor de colas local.

Cada definición de proceso tiene un nombre que es diferente de los nombres de otras definiciones de proceso que pertenecen al gestor de colas. Pero el nombre de la definición de proceso puede ser el mismo que los nombres de otros objetos de gestor de colas de tipos diferentes (por ejemplo, colas).

Para determinar el valor de este atributo, utilice el selector MQCA_PROCESS_NAME con la llamada MQINQ.

La longitud de este atributo la proporciona MQ_PROCESS_NAME_LENGTH.

QSGDisp (MQLONG)

Especifica la disposición de la definición de proceso. El valor puede ser uno de los siguientes:

MQQSGD_Q_MGR
El objeto tiene disposición de gestor de colas: la definición de objeto sólo es conocida por el gestor de colas local; la definición no es conocida por otros gestores de colas del grupo de compartición de colas.

Cada gestor de colas del grupo de compartición de colas puede tener un objeto con el mismo nombre y tipo que el objeto actual, pero estos son objetos separados y no hay ninguna correlación entre ellos. Sus atributos no están restringidos a ser iguales entre sí.

MQQSGD_COPY
El objeto es una copia local de una definición de objeto maestro que existe en el repositorio compartido. Cada gestor de colas del grupo de compartición de colas puede tener su propia copia del objeto. Inicialmente, todas las copias tienen los mismos atributos, pero puede modificar cada copia, utilizando mandatos MQSC, para que sus atributos difieran de los de las otras copias. Los atributos de las copias se resincronizan cuando se modifica la definición maestra en el repositorio compartido.

Para determinar el valor de este atributo, utilice el selector MQIA_QSG_DISP con la llamada MQINQ.

Este atributo sólo está soportado en z/OS.

UserData (MQCHAR128)

UserData es una serie de caracteres que contiene información de usuario perteneciente a la aplicación que se va a iniciar. Esta información es para que la utilice una aplicación de supervisor desencadenante que procesa mensajes en la cola de inicio, o la aplicación iniciada por el supervisor desencadenante. La información se envía a la cola de inicio como parte del mensaje desencadenante.

El significado de *UserData* lo determina la aplicación de supervisor desencadenante. El supervisor desencadenante proporcionado por WebSphere MQ pasa *UserData* a la aplicación iniciada como parte de la lista de parámetros. La lista de parámetros consta de la estructura MQTMC2 (que contiene *UserData*), seguida de un espacio en blanco, seguida de *EnvData* con los espacios en blanco finales eliminados.

La serie de caracteres no puede contener nulos. Se rellena a la derecha con espacios en blanco si es necesario. Para Microsoft Windows, la serie de caracteres no debe contener comillas dobles si la definición de proceso se va a pasar a **runmqtrm**.

Para determinar el valor de este atributo, utilice el selector MQCA_USER_DATA con la llamada MQINQ. La longitud de este atributo la proporciona MQ_PROCESS_USER_DATA_LENGTH.

Códigos de retorno

Para cada llamada a la Interfaz de cola de mensajes (MQI) de WebSphere MQ y a la Interfaz de administración (MQAI) de WebSphere MQ , el gestor de colas o una rutina de salida devuelven un código de **terminación** y un código de **razón** para indicar el éxito o el error de la llamada.

Las aplicaciones no deben depender de los errores que se comprueban en un orden específico, excepto cuando se indique específicamente. Si puede surgir más de un código de terminación o código de razón de una llamada, el error concreto notificado depende de la implementación.

Las aplicaciones que comprueban la finalización satisfactoria después de una llamada de API de WebSphere MQ siempre deben comprobar el código de finalización. No asuma el valor del código de terminación, basándose en el valor del código de razón.

Códigos de finalización

El parámetro de código de terminación (*CompCode*) permite al interlocutor ver rápidamente si la llamada se ha completado correctamente, se ha completado parcialmente o ha fallado.A continuación se muestra una lista de códigos de terminación, con más detalles de los que se proporcionan en las descripciones de las llamadas:

MQCC_OK
La llamada se ha completado del todo; se han establecido todos los parámetros de salida. El parámetro *Reason* siempre tiene el valor MQRC_NONE en este caso.

MQCC_WARNING
La llamada se ha completado parcialmente. Es posible que algunos parámetros de salida se hayan establecido además de los parámetros de salida *CompCode* y *Reason* . El parámetro *Reason* proporciona información adicional sobre la finalización parcial.

MQCC_FAILED
El proceso de la llamada no se ha completado. El estado del gestor de colas no se modifica, excepto cuando se indique específicamente. Se han establecido los parámetros de salida *CompCode* y *Reason* ; otros parámetros no se modifican, excepto cuando se indique lo contrario.

La razón puede ser un error en el programa de aplicación, o puede ser el resultado de alguna situación externa al programa, por ejemplo, es posible que se haya revocado la autorización del usuario. El parámetro *Reason* proporciona información adicional sobre el error.

códigos de razón

El parámetro de código de razón (*Reason*) califica el parámetro de código de terminación (*CompCode*).

Si no hay que notificar ninguna razón especial, se devuelve MQRC_NONE. Una llamada que ha finalizado correctamente devuelve MQCC_OK y MQRC_NONE.

Si el código de terminación es MQCC_WARNING o MQCC_FAILED, el gestor de colas siempre informa de una razón calificadora; se proporcionan detalles en la descripción de cada llamada.

Cuando las rutinas de salida de usuario establecen códigos de terminación y razones, deben cumplir estas reglas. Además, cualquier valor de razón especial definido por las salidas de usuario debe ser menor que cero, para asegurarse de que no entra en conflicto con los valores definidos por el gestor de colas. Las salidas pueden establecer razones ya definidas por el gestor de colas, cuando proceda.

También aparecen códigos de razón en:

- El campo *Reason* de la estructura MQDLH

- El campo *Feedback* de la estructura MQMD

Para obtener descripciones completas de los códigos de razón, consulte Códigos de razón .

Reglas para validar las opciones de MQI

Esta sección lista las situaciones que producen un código de razón MQRC_OPTIONS_ERROR a partir de una llamada MQOPEN, MQPUT, MQPUT1, MQGET, MQCLOSE o MQSUB.

llamada MQOPEN

Para las opciones de la llamada MQOPEN:

- Se debe especificar al menos *uno* de los siguientes:
  - MQOO_BROWSE
  - MQOO_INPUT_EXCLUSIVE[1]
  - MQOO_INPUT_SHARED[1]
  - MQOO_INPUT_AS_Q_DEF[1]
  - MQOO_INQUIRE
  - MQOO_OUTPUT
  - MQOO_SET
  - MQOO_BIND_ON_OPEN[2]
  - MQOO_BIND_NOT_FIXED[2]
  - MQOO_BIND_ON_GROUP[2]
  - MQOO_BIND_AS_Q_DEF[2]
- Solo se permite *uno* de los siguientes:
  - MQOO_READ_AHEAD
  - MQOO_NO_READ_AHEAD
  - MQOO_READ_AHEAD_AS_Q_DEF

1. Solo se permite *uno* de los siguientes:
   - MQOO_INPUT_EXCLUSIVE
   - MQOO_INPUT_SHARED
   - MQOO_INPUT_AS_Q_DEF
2. Solo se permite *uno* de los siguientes:
   - MQOO_BIND_ON_OPEN
   - MQOO_BIND_NOT_FIXED
   - MQOO_BIND_ON_GROUP
   - MQOO_BIND_AS_Q_DEF

   Nota: Las opciones listadas anteriormente son mutuamente excluyentes. Sin embargo, como el valor de MQOO_BIND_AS_Q_DEF es cero, si se especifica con cualquiera de las otras dos opciones de vinculación no se genera el código de razón MQRC_OPTIONS_ERROR. MQOO_BIND_AS_Q_DEF se proporciona para ayudar a la documentación del programa.

- Si se especifica MQOO_SAVE_ALL_CONTEXT, también debe especificarse una de las opciones MQOO_INPUT_ *.
- Si se especifica una de las opciones MQOO_SET_ * _CONTEXT o MQOO_PASS_ * _CONTEXT, también debe especificarse MQOO_OUTPUT.
- Si se especifica MQOO_CO_OP, también se debe especificar MQOO_BROWSE
- Si se especifica MQOO_NO_MULTICAST, también debe especificarse MQOO_OUTPUT.

llamada MQPUT

Para las opciones de colocación de mensajes:

- La combinación de MQPMO_SYNCPOINT y MQPMO_NO_SYNCPOINT no está permitida.
- Solo se permite *uno* de los siguientes:
  - MQPMO_DEFAULT_CONTEXT
  - MQPMO_NO_CONTEXT
  - MQPMO_PASS_ALL_CONTEXT
  - MQPMO_PASS_IDENTITY_CONTEXT
  - MQPMO_SET_ALL_CONTEXT
  - MQPMO_SET_IDENTITY_CONTEXT
- Solo se permite *uno* de los siguientes:
  - MQPMO_ASYNC_RESPONSE
  - MQPMO_SYNC_RESPONSE
  - MQPMO_RESPONSE_AS_TOPIC_DEF
  - MQPMO_RESPONSE_AS_Q_DEF
- MQPMO_ALTERNATE_USER_AUTHORITY no está permitido (sólo es válido en la llamada MQPUT1).

llamada MQPUT1

Para las opciones de transferencia de mensajes, las reglas son las mismas que para la llamada MQPUT, excepto para lo siguiente:

- Se permite MQPMO_ALTERNATE_USER_AUTHORITY.
- MQPMO_LOGICAL_ORDER *no* está permitido.

llamada MQGET

Para las opciones get-message:

- Solo se permite *uno* de los siguientes:
  - MQGMO_NO_SYNCPOINT
  - MQGMO_SYNCPOINT
  - MQGMO_SYNCPOINT_IF_PERSISTENT
- Solo se permite *uno* de los siguientes:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
  - MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT no está permitido con ninguno de los siguientes elementos:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
  - MQGMO_LOCK
  - MQGMO_UNLOCK
- MQGMO_SYNCPOINT_IF_PERSISTENT no está permitido con ninguno de los siguientes:
  - MQGMO_BROWSE_FIRST

  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
  - MQGMO_COMPLETE_MSG
  - MQGMO_UNLOCK
- MQGMO_MARK_SKIP_BACKOUT requiere que se especifique MQGMO_SYNCPOINT.
- La combinación de MQGMO_WAIT y MQGMO_SET_SIGNAL no está permitida.
- Si se especifica MQGMO_LOCK, también se debe especificar uno de los siguientes:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
- Si se especifica MQGMO_UNLOCK, sólo se permite lo siguiente:
  - MQGMO_NO_SYNCPOINT
  - MQGMO_NO_WAIT

llamada MQCLOSE

Para las opciones de la llamada MQCLOSE:

- La combinación de MQCO_DELETE y MQCO_DELETE_PURGE no está permitida.
- Sólo se permite uno de los siguientes:
  - MQCO_KEEP_SUB
  - MQCO_REMOVE_SUB

Llamada MQSUB

Para las opciones de la llamada MQSUB:

- Se debe especificar al menos uno de los siguientes:
  - MQSO_ALTER
  - MQSO_RESUME
  - MQSO_CREATE
- Sólo se permite uno de los siguientes:
  - MQSO_DURABLE
  - MQSO_NON_DURABLE

  Nota: Las opciones listadas anteriormente son mutuamente excluyentes. Sin embargo, como el valor de MQSO_NON_DURABLE es cero, si se especifica con MQSO_DURABLE no se genera el código de razón MQRC_OPTIONS_ERROR. Se proporciona MQSO_NON_DURABLE para ayudar a la documentación del programa.
- La combinación de MQSO_GROUP_SUB y MQSO_MANAGED no está permitida.
- MQSO_GROUP_SUB requiere que se especifique MQSO_SET_CORREL_ID.
- Sólo se permite uno de los siguientes:
  - MQSO_ANY_USERID
  - MQSO_FIXED_USERID
- MQSO_NEW_PUBLICATIONS_ONLY sólo se permite en combinación con MQSO_CREATE.
- La combinación de MQSO_PUBLICATIONS_ON_REQUEST y SubLevel mayor que 1 no está permitida.
- Sólo se permite uno de los siguientes:
  - MQSO_WILDCARD_CHAR

  - MQSO_WILDCARD_TOPIC
- MQSO_NO_MULTICAST requiere que se especifique MQSO_MANAGED.

Mensajes del mandato de publicación/suscripción en cola

Una aplicación puede utilizar mensajes de mandato MQRFH2 para controlar una aplicación de publicación/suscripción en cola.

Una aplicación que utiliza MQRFH2 para la publicación/suscripción puede enviar los siguientes mensajes de mandato al SYSTEM.BROKER.CONTROL.QUEUE:

- "Mensaje de supresión de publicación" en la página 859
- "Mensaje de anulación de registro de suscriptor" en la página 860
- "Publicar mensaje" en la página 864
- "Mensaje Registrar suscriptor" en la página 867
- "Mensaje de solicitud de actualización" en la página 872

Si está escribiendo aplicaciones de publicación/suscripción en cola, debe comprender estos mensajes, el mensaje de respuesta del gestor de colas y el descriptor de mensaje (MQMD); consulte la información siguiente:

- "Mensaje de respuesta del gestor de colas" en la página 874
- "Valores de MQMD para publicaciones reenviadas por un gestor de colas" en la página 879
- "Valores de MQMD en mensajes de respuesta del gestor de colas" en la página 880
- "Códigos de razón de publicación/suscripción" en la página 875

Los mandatos están contenidos en una carpeta <psc> en el campo **NameValueData** de la cabecera MQRFH2 . El mensaje que puede enviar un intermediario en respuesta a un mensaje de mandato está contenido en una carpeta <pscr> .

Las descripciones de cada mandato listan las propiedades que puede contener una carpeta. A menos que se especifique lo contrario, las propiedades son opcionales y sólo pueden aparecer una vez.

Los nombres de las propiedades se muestran como <Command>.
Los valores deben estar en formato de serie, por ejemplo: Publish.
Una constante de tipo serie que representa el valor de una propiedad se muestra entre paréntesis, por ejemplo: (MQPSC_PUBLISH).
Las constantes de tipo serie se definen en el archivo de cabecera cmqpsc.h que se proporciona con el gestor de colas.

Mensaje de supresión de publicación

El mensaje del mandato **Delete Publication** se envía a un gestor de colas desde un publicador, o desde otro gestor de colas, para indicar al gestor de colas que suprima las publicaciones retenidas para los temas especificados.

Este mensaje se envía a una cola supervisada por la interfaz de publicación/suscripción en cola del gestor de colas.

La cola de entrada ha de ser la cola a la que se envió la publicación original.

Si tiene autorización para algunos, pero no para todos, los temas especificados en el mensaje de mandato **Delete Publication** , sólo se suprimirán esos temas. Un mensaje **Broker Response** indica qué temas no se suprimen.

De forma similar, si un mandato **Publish** contiene más de un tema, un mandato **Delete Publication** que coincide con algunos de estos temas, pero no todos, suprime sólo las publicaciones para los temas que se especifican en el mandato **Delete Publication** .

Consulte “Valores de MQMD para publicaciones reenviadas por un gestor de colas” en la página 879 para obtener detalles de los parámetros del descriptor de mensaje (MQMD) que son necesarios al enviar un mensaje de mandato al gestor de colas.

Propiedades

<Command> (*MQPSC_COMMAND*)
El valor es DeletePub(*MQPSC_DELETE_PUBLICATION*).

Esta propiedad ha de especificarse.

<Topic> (*MQPSC_TOPIC*)
El valor es una serie de caracteres que contiene un tema para el cual han de suprimirse las publicaciones retenidas. Se pueden incluir caracteres comodín en la serie de caracteres para suprimir publicaciones sobre más de un tema.

Esta propiedad ha de especificarse; puede repetirse para tantos temas como se desee.

<DelOpt> (*MQPSC_DELETE_OPTION*)
La propiedad de opciones de supresión puede tener uno de los siguientes valores:

Local (*MQPSC_LOCAL*)

Todas las publicaciones retenidas para los temas especificados se suprimen en el gestor de colas local (es decir, el gestor de colas al que se envía este mensaje), tanto si se han publicado con la opción Local como si no.

Las publicaciones de otros gestores de colas no se ven afectadas.

None (*MQPSC_NONE*)

Todas las opciones toman sus valores predeterminados. Esto tiene el mismo efecto que especificar la propiedad DelOpt. Si se especifican otras opciones simultáneamente, se ignora None.

El valor predeterminado si se omite esta propiedad es que todas las publicaciones retenidas para los temas especificados se suprimen en todos los gestores de colas de la red, independientemente de si se han publicado con la opción Local .

Ejemplo

A continuación se muestra un ejemplo de NameValueData para un mensaje de mandato **Delete Publication** . Lo utiliza la aplicación de ejemplo para suprimir, en el gestor de colas local, la publicación retenida que contiene la última puntuación en la coincidencia entre Team1 y Team2.

```
<psc>
 <Command>DeletePub</Command>
 <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
 <DelOpt>Local</DelOpt>
</psc>
```

Mensaje de anulación de registro de suscriptor

El mensaje del mandato **Deregister Subscriber** lo envía a un gestor de colas un suscriptor, u otra aplicación en nombre de un suscriptor, para indicar que ya no desea recibir mensajes que coincidan con los parámetros especificados.

Este mensaje se envía a SYSTEM.BROKER.CONTROL.QUEUE, la cola de control del gestor de colas. El usuario debe tener la autorización necesaria para colocar un mensaje en esta cola.

Consulte Valores de MQMD para publicaciones reenviadas por un gestor de colas para obtener detalles de los parámetros del descriptor de mensaje (MQMD) que son necesarios al enviar un mensaje de mandato al gestor de colas.

Se puede anular el registro de una suscripción individual especificando el tema correspondiente, el punto de suscripción y los valores de filtro de la suscripción original. Si alguno de los valores no se ha especificado (es decir, han tomado los valores predeterminados) en la suscripción original, se deben omitir cuando se anule el registro de la suscripción.

Todas las suscripciones para un suscriptor, o un grupo de suscriptores, se pueden anular del registro utilizando la opción `DeregAll` . Por ejemplo, si se especifica `DeregAll` , junto con un punto de suscripción (pero sin tema ni filtro), se anularán el registro de todas las suscripciones para el suscriptor en el punto de suscripción especificado, independientemente del tema y del filtro. Se permite cualquier combinación de tema, filtro y punto de suscripción; si se especifican los tres, sólo puede coincidir una suscripción y se ignora la opción `DeregAll` .

El mensaje debe enviarlo el suscriptor que ha registrado la suscripción; esto se confirma comprobando el ID de usuario del suscriptor.

Un administrador del sistema también puede anular el registro de las suscripciones utilizando los mandatos MQSC o PCF. Sin embargo, las suscripciones registradas con una cola dinámica temporal están asociadas con la cola, no sólo con el nombre de cola. Si la cola se suprime, ya sea de forma explícita, o mediante la desconexión de la aplicación del gestor de colas, ya no es posible utilizar el mandato **`Deregister Subscriber`** para anular el registro de las suscripciones para dicha cola. Las suscripciones se pueden anular del registro utilizando el entorno de trabajo del desarrollador y el gestor de colas las elimina automáticamente la próxima vez que coincida con una publicación de la suscripción o la próxima vez que se reinicie el gestor de colas. En circunstancias normales, las aplicaciones deben anular el registro de sus suscripciones antes de suprimir la cola o de desconectarse del gestor de colas.

Si un suscriptor envía un mensaje para anular el registro de una suscripción y recibe un mensaje de respuesta para decir que se ha procesado correctamente, es posible que algunas publicaciones sigan llegando a la cola de suscriptores si el gestor de colas las estaba procesando al mismo tiempo que se estaba anulando el registro de la suscripción. Si los mensajes no se eliminan de la cola, es posible que haya una acumulación de mensajes no procesados en la cola del suscriptor. Si la aplicación ejecuta un bucle que incluye una llamada MQGET con el `CorrelId` adecuado después de permanecer inactivo durante un tiempo, estos mensajes se borran de la cola.

De forma similar, si el suscriptor utiliza una cola dinámica permanente y anula el registro y cierra la cola con la opción *MQCO_DELETE_PURGE* en una llamada MQCLOSE, es posible que la cola no esté vacía. Si alguna publicación del gestor de colas todavía no se ha confirmado cuando se suprime la cola, la llamada MQCLOSE emite un código de retorno MQRC_Q_NOT_EMPTY. La aplicación puede evitar este problema durmiendo y volviendo a emitir la llamada MQCLOSE de vez en cuando.

Propiedades

<Command> (*MQPSC_COMMAND*)
El valor es `DeregSub` (*MQPSC_DEREGISTER_SUBSCRIBER*).

Esta propiedad ha de especificarse.

<Topic> (*MQPSC_TOPIC*)
El valor es una serie que contiene el tema que se va a anular el registro.

Esta propiedad, opcionalmente, se puede repetir si se van a anular el registro de varios temas. Se puede omitir si se especifica `DeregAll` en <RegOpt>.

Los temas especificados pueden ser un subconjunto de los que están registrados si el suscriptor desea retener suscripciones para otros temas. Se permiten caracteres comodín, pero una serie de tema que contiene caracteres comodín debe coincidir exactamente con la serie correspondiente que se ha especificado en el mensaje de mandato **`Deregister Subscriber`** .

<SubPoint> (*MQPSC_SUBSCRIPTION_POINT*)
El valor es una serie que especifica el punto de suscripción del que se va a desconectar la suscripción.

Esta propiedad no debe repetirse. Se puede omitir si se especifica un < Topic> , o si se especifica DeregAll en <RegOpt>. Si omite esta propiedad, se produce lo siguiente:

- Si **no** especifica DeregAll, las suscripciones que coincidan con la propiedad < Topic> (y la propiedad < Filter > , si está presente) se anularán el registro del punto de suscripción predeterminado.
- Si especifica DeregAll, todas las suscripciones (que coincidan con las propiedades < Topic> y < Filter > si están presentes) se anularán del registro de todos los puntos de suscripción.

Tenga en cuenta que no puede especificar el punto de suscripción predeterminado de forma explícita. Por lo tanto, no hay forma de anular el registro de todas las suscripciones de este punto de suscripción solamente; debe especificar los temas.

<SubIdentity> (*MQPSC_SUBSCRIPTION_IDENTITY*)
Se trata de una serie de longitud variable con una longitud máxima de 64 caracteres. Se utiliza para representar una aplicación con un interés en una suscripción. El gestor de colas mantiene un conjunto de identidades de suscriptor para cada suscripción. Cada suscripción puede permitir que su conjunto de identidades contenga una sola identidad o un número ilimitado de identidades.

Si SubIdentity está en el conjunto de identidades para la suscripción, se elimina del conjunto. Si el conjunto de identidades queda vacío como resultado de esto, la suscripción se elimina del gestor de colas, a menos que se especifique LeaveOnly como valor de la propiedad RegOpt . Si el conjunto de identidades sigue conteniendo otras identidades, la suscripción no se elimina del gestor de colas y el flujo de publicación no se interrumpe.

Si se especifica SubIdentity , pero la SubIdentity no está en el conjunto de identidades para la suscripción, el mandato **Deregister Subscriber** falla con el código de retorno *MQRCCF_SUB_IDENTITY_ERROR*.

< Filtro > (*MQPSC_FILTER*)
El valor es una serie que especifica el filtro que se va a anular el registro. Debe coincidir exactamente, incluyendo mayúsculas y minúsculas y cualquier espacio, con un filtro de suscripción que se haya registrado previamente.

Esta propiedad puede, opcionalmente, repetirse si se va a anular el registro de más de un filtro. Se puede omitir si se especifica un < Topic> , o si se especifica DeregAll en <RegOpt>.

Los filtros especificados pueden ser un subconjunto de los registrados si el suscriptor desea retener suscripciones para otros filtros.

<RegOpt> (*MQPSC_REGISTRATION_OPTION*)
La propiedad de opciones de registro puede tomar los valores siguientes:

DeregAll
(*MQPSC_DEREGISTER_ALL*)

Se anulará el registro de todas las suscripciones coincidentes registradas para este suscriptor.

Si especifica DeregAll:

- < Topic>, <SubPoint>y < Filter > se pueden omitir.
- < Topic> y < Filtro > se pueden repetir, si es necesario.
- <SubPoint> no debe repetirse.

Si **no** especifica DeregAll:

- Se debe especificar < Topic> y se puede repetir si es necesario.
- <SubPoint> y < Filter > se pueden omitir.
- <SubPoint> no debe repetirse.
- < Filtro > se puede repetir, si es necesario.

Si los temas y los filtros se repiten, se eliminarán todas las suscripciones que coincidan con todas las combinaciones de los dos. Por ejemplo, un mandato **Deregister Subscriber** que especifique tres temas y tres filtros intentará eliminar nueve suscripciones.

CorrelAsId
(*MQPSC_CORREL_ID_AS_IDENTITY*)

El CorrelId del descriptor de mensaje (MQMD), que no debe ser cero, se utiliza para identificar al suscriptor. Debe coincidir con el CorrelId utilizado en la suscripción original.

FullResp
(*MQPSC_FULL_RESPONSE*)

Cuando se especificaFullResp , todos los atributos de la suscripción se devuelven en el mensaje de respuesta, si el mandato no falla.

Cuando se especifica FullResp , no se permite DeregAll en el mandato **Deregister Subscriber** . Tampoco es posible especificar varios temas. El mandato falla con el código de retorno *MQRCCF_REG_OPTIONS_ERROR*, en ambos casos.

LeaveOnly
(*MQPSC_LEAVE_ONLY*)

Cuando se especifica con una SubIdentity que está en el conjunto de identidades para la suscripción, la SubIdentity se elimina del conjunto de identidades para la suscripción. La suscripción no se elimina del gestor de colas, aunque el conjunto de identidades resultante esté vacío. Si el valor SubIdentity no está en el conjunto de identidades, el mandato falla con el código de retorno *MQRCCF_SUB_IDENTITY_ERROR*.

Si se especifica LeaveOnly sin SubIdentity, el mandato falla con el código de retorno *MQRCCF_REG_OPTIONS_ERROR*.

Si no se especifica LeaveOnly ni una SubIdentity , la suscripción se elimina independientemente del contenido del conjunto de identidades para la suscripción.

NO
(MQPSC_NONE)

Todas las opciones toman sus valores predeterminados. Esto tiene el mismo efecto que omitir la propiedad de opciones de registro. Si se especifican otras opciones simultáneamente, se ignora None.

VariableUserId
(*MQPSC_VARIABLE_USER_ID*)

Cuando se especifica, la identidad del suscriptor (cola, gestor de colas y correlid) no está restringida a un único ID de usuario. Esto difiere del comportamiento existente del gestor de colas que asocia el ID de usuario del mensaje de registro original con la identidad del suscriptor y, a partir de entonces, impide que cualquier otro usuario utilice dicha identidad. Si un nuevo suscriptor intenta utilizar la misma identidad, se devuelve el código de retorno *MQRCCF_DUPLICATE_SUBSCRIPTION* .

Cualquier usuario puede modificar o anular el registro de la suscripción cuando tenga la autorización adecuada, evitando la comprobación existente de que el ID de usuario debe coincidir con el del suscriptor original.

Para añadir esta opción a una suscripción existente, el mandato debe proceder del mismo ID de usuario que la propia suscripción original.

Si la suscripción que se va a anular el registro tiene establecido VariableUserId , se debe establecer al anular el registro para indicar qué suscripción se va a anular el registro. De lo contrario, el ID de usuario del mandato **Deregister Subscriber** se utiliza para identificar la suscripción. Esto se altera temporalmente, junto con los otros identificadores de suscriptor, si se proporciona un nombre de suscripción.

El valor predeterminado, si se omite esta propiedad, es que no se establecen opciones de registro.

<QMgrName> (*MQPSC_Q_MGR_NAME*)
El valor es el nombre del gestor de colas para la cola de suscriptor. Debe coincidir con el QMgrName utilizado en la suscripción original.

Si se omite esta propiedad, el valor predeterminado es el nombre del gestor de la cola de respuestas (ReplyToQMgr) que hay en el descriptor del mensaje (MQMD). Si el nombre resultante está en blanco, el valor predeterminado es el nombre del gestor de colas.

< QName> (*MQPSC_Q_NAME*)
El valor es el nombre de la cola de suscriptores. Debe coincidir con el QName utilizado en la suscripción original.

Si se omite esta propiedad, el valor predeterminado es el nombre ReplyToQ en el descriptor de mensaje (MQMD), que no debe estar en blanco.

<SubName> (*MQPSC_SUBSCRIPTION_NAME*)
Si especifica SubName en un mandato **Deregister Subscriber** , el valor SubName tiene prioridad sobre todos los demás campos de identificador excepto el ID de usuario, a menos que VariableUserId esté establecido en la propia suscripción. Si no se ha establecido VariableUserId , el mandato **Deregister Subscriber** sólo se ejecuta correctamente si el ID de usuario del mensaje de mandato coincide con el de la suscripción, si no es así, el mandato falla con el código de retorno *MQRCCF_DUPLICATE_IDENTITY*.

Si existe una suscripción que coincide con la identidad tradicional de este mandato pero no tiene ningún SubName , el mandato **Deregister Subscriber** falla con el código de retorno *MQRCCF_SUB_NAME_ERROR*. Si se intenta anular el registro de una suscripción que tiene un SubName utilizando un mensaje de mandato que coincide con la identidad tradicional pero sin ningún SubName especificado, el mandato se ejecuta correctamente.

< Datos deSubUser> (*MQPSC_SUBSCRIPTION_USER_DATA*)
Es una serie de texto de longitud variable. El valor lo almacena el gestor de colas con la suscripción, pero no influye en la entrega de la publicación al suscriptor. El valor se puede modificar volviendo a registrar en la misma suscripción con un nuevo valor. Este atributo es para el uso de la aplicación.

SubUserLos datos se devuelven en la información metatópica (MQCACF_REG_SUB_USER_DATA) para una suscripción, si SubUserData está presente.

Ejemplo

A continuación se muestra un ejemplo de NameValueData para un mensaje de mandato **Deregister Subscriber** . En este ejemplo, la aplicación de ejemplo anula el registro de su suscripción a los temas que contienen la última puntuación para todas las coincidencias. La identidad del suscriptor, incluido el CorrelId, se toma de los valores predeterminados en MQMD.

```
<psc>
 <Command>DeregSub</Command>
 <RegOpt>CorrelAsId</RegOpt>
 <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Publicar mensaje

El mensaje del mandato **Publish** se coloca en una cola, o desde un gestor de colas a un suscriptor, para publicar información sobre un tema o temas especificados.

Es necesaria la autorización para colocar un mensaje en una cola y la autorización para publicar información sobre un tema o temas especificados.

Si el usuario tiene autorización para publicar información sobre algunos temas, pero no todos, sólo se utilizan estos temas para publicar; una respuesta de aviso indica qué temas no se utilizan para publicar.

Si un suscriptor tiene suscripciones coincidentes, el gestor de colas reenvía el mensaje **Publish** a las colas de suscriptor definidas en los mensajes de mandato **Register Subscriber** correspondientes.

Consulte Mensaje de respuesta del gestor de colas para obtener detalles de los parámetros del descriptor de mensaje (MQMD) necesarios al enviar un mensaje de mandato al gestor de colas y utilizados cuando un gestor de colas reenvía una publicación a un suscriptor.

El gestor de colas reenvía el mensaje **Publish** a otros gestores de colas de la red que tienen suscripciones coincidentes, a menos que sea una publicación local.

Los datos de la publicación, si los hay, se incluyen en el cuerpo del mensaje. Los datos se pueden describir en una carpeta <mcd>, en el campo NameValueData de la cabecera MQRFH2.

Propiedades

< Command> (*MQPSC_COMMAND*)
El valor es Publish(*MQPSC_PUBLISH*).

Esta propiedad ha de especificarse.

< Topic> (*MQPSC_TOPIC*)
El valor es una serie de caracteres que contiene un tema que clasifica esta publicación. No se permiten caracteres comodín.

Debe añadir el tema a la lista de nombres SYSTEM.QPUBSUB.QUEUE.NAMELIST, consulte Adición de una corriente para obtener instrucciones sobre cómo completar esta tarea.

Esta propiedad ha de especificarse y puede repetirse, opcionalmente, para tantos temas como se desee.

<SubPoint> (*MQPSC_SUBSCRIPTION_POINT*)
Es el punto de suscripción en el que se publica la publicación.

En WebSphere Event Broker V6, el valor de la propiedad <SubPoint> es el valor del atributo Punto de suscripción del nodo Publication que maneja la publicación.

En WebSphere MQ V7.0.1, el valor de la propiedad <SubPoint> debe coincidir con el nombre de un punto de suscripción. Consulte Adición de un punto de suscripción .

<PubOpt> (*MQPSC_PUBCIONTION_OPTION*)
La propiedad de opciones de publicación puede tener uno de los siguientes valores:

RetainPub
(*MQPSC_RETAIN_PUB*)

El gestor de colas debe retener una copia de la publicación. Si esta opción no está establecida, la publicación se suprime tan pronto como el gestor de colas ha enviado la publicación a todos sus suscriptores actuales.

IsRetainedPub
(*MQPSC_IS_RETAINED_PUB*)

(Sólo puede ser establecido por un gestor de colas.) Esta publicación ha sido retenida por el gestor de colas. El gestor de colas establece esta opción para notificar a un suscriptor que esta publicación se ha publicado anteriormente y se ha retenido, siempre que la suscripción se haya registrado con la opción InformIfRetenido . Sólo se establece en respuesta a un mensaje de mandato Register Subscriber o Request Update . Las publicaciones retenidas que se envían directamente a suscriptores no tiene establecida esta opción.

Local
(*MQPSC_LOCAL*)

Esta opción indica al gestor de colas que esta publicación no debe enviarse a otros gestores de colas. Todos los suscriptores registrados en este gestor de colas reciben esta publicación si tienen suscripciones coincidentes.

OtherSubsOnly
(*MQPSC_OTHER_SUBS_ONLY*)

Esta opción permite el proceso más sencillo de aplicaciones de tipo conferencia, en las que un publicador es también suscriptor del mismo tema. Indica al gestor de colas que no envíe la publicación a la cola de suscriptores del publicador aunque tenga una suscripción coincidente. La

cola de suscriptores del publicador consta de su QMgrName, QNamey CorrelIdopcional, tal como se describe en la lista siguiente.

CorrelAsId
(*MQPSC_CORREL_ID_AS_IDENTITY*)

El CorrelId del MQMD (que no ha de ser cero) forma parte de la cola de suscriptores del publicador en aplicaciones donde el publicador es también un suscriptor.

NO
(*MQPSC_NONE*)

Todas las opciones toman sus valores predeterminados. Esto surte el mismo efecto que omitir la propiedad de opciones de publicación. Si se especifican otras opciones simultáneamente, se ignora None.

Para tener varias opciones de publicación, solo es necesario introducir elementos <PubOpt> adicionales.

El valor predeterminado, si se omite esta propiedad, es que no se establece ninguna opción de publicación.

<PubTime> (*MQPSC_PUBLISH_TIMESTAMP*)
El valor es una indicación de la hora de publicación, opcional, establecida por el publicador. Tiene 16 caracteres de largo con el formato:

```
YYYYMMDDHHMMSSTH
```

y utiliza la Hora Universal. El gestor de colas no comprueba esta información antes de enviarla a los suscriptores.

<SeqNum> (*MQPSC_SEQUENCE_NUMBER*)
El valor es un número de secuencia opcional establecido por el publicador.

Debe incrementarse en 1 con cada publicación. Sin embargo, esto no lo comprueba el gestor de colas, que simplemente transmite esta información a los suscriptores.

Si las publicaciones sobre el mismo tema se publican en distintos gestores de colas interconectados, es responsabilidad de los publicadores asegurarse de que los números de secuencia, si se utilizan, sean significativos.

<QMgrName> (*MQPSC_Q_MGR_NAME*)
El valor es una serie de caracteres que contiene el nombre del gestor de colas correspondiente a la cola de suscriptores del publicador, en aplicaciones donde el publicador es también un suscriptor (vea OtherSubsOnly).

Si se omite esta propiedad, el valor predeterminado es el nombre del gestor de la cola de respuestas (ReplyToQMgr) que hay en el descriptor del mensaje (MQMD). Si el nombre resultante está en blanco, el valor predeterminado es el nombre del gestor de colas.

< QName> (*MQPSC_Q_NAME*)
El valor es una serie de caracteres que contiene el nombre de la cola de suscriptores del publicador, en aplicaciones en las que el publicador es también un suscriptor (vea OtherSubsOnly).

Si se omite esta propiedad, el valor predeterminado es el nombre de la cola de respuestas (ReplyToQ) en el descriptor del mensaje (MQMD), que no ha de estar en blanco si se ha establecido OtherSubsOnly.

Ejemplo

A continuación se muestran algunos ejemplos de *NameValueData* para un mensaje de mandato **Publish** .

El primer ejemplo es para una publicación enviada por el simulador de coincidencias en la aplicación de ejemplo para indicar que se ha iniciado una coincidencia.

```
<psc>
 <Command>Publish</Command>
 <Topic>Sport/Soccer/Event/MatchStarted</Topic>
</psc>
```

El segundo ejemplo es para una publicación retenida. Se publica la última puntuación entre Team1 y Team2.

```
<psc>
 <Command>Publish</Command>
 <PubOpt>RetainPub</PubOpt>
 <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
</psc>
```

Mensaje Registrar suscriptor

El mensaje del mandato **Register Subscriber** lo envía a un gestor de colas un suscriptor, o otra aplicación en nombre de un suscriptor, para indicar que desea suscribirse a uno o más temas en un punto de suscripción. También se puede especificar un filtro de contenido de mensaje.

En las expresiones de filtro de publicación/suscripción, el anidamiento de paréntesis hace que el rendimiento disminuya exponencialmente. Evite anidar paréntesis a una profundidad mayor que aproximadamente 6.

El mensaje se envía a SYSTEM.BROKER.CONTROL.QUEUE, que es la cola de control del gestor de colas. Se necesita autorización para transferir un mensaje a esta cola, además de la autorización de acceso (establecida por el administrador del sistema del gestor de colas) para el tema o temas de la suscripción.

Si el usuario tiene autorización sobre algunos temas, pero no todos, sólo se registran los que tienen autorización; una respuesta de aviso indica los que no están registrados.

Consulte "Valores de MQMD en los mensajes de mandato para el gestor de colas" en la página 878 para obtener detalles de los parámetros del descriptor de mensaje (MQMD) que son necesarios al enviar un mensaje de mandato al gestor de colas.

Si la respuesta a la cola es una cola dinámica temporal, el gestor de colas anula automáticamente el registro de la suscripción cuando se cierra la cola.

Propiedades

<Command> (*MQPSC_COMMAND*)
El valor es RegSub (*MQPSC_REGISTER_SUBSCRIBER*). Esta propiedad ha de especificarse.

<Topic> (*MQPSC_TOPIC*)
El tema para el que el suscriptor desea recibir publicaciones. Los caracteres comodín se pueden especificar como parte del tema.

Si utiliza el mandato MQSC **display sub** para examinar la suscripción creada de esta forma, el valor de la etiqueta < Topic> se muestra como la propiedad TOPICSTR de la suscripción.

Esta propiedad es necesaria y, opcionalmente, se puede repetir para tantos temas como sea necesario.

<SubPoint> (*MQPSC_SUBSCRIPTION_POINT*)
El valor es el punto de suscripción al que se adjunta la suscripción.

Si se omite esta propiedad, se utiliza el punto de suscripción predeterminado.

En WebSphere Event Broker V6, el valor de la propiedad <SubPoint> debe coincidir con el valor del atributo Punto de suscripción de los nodos de publicación a los que están suscritos.

En WebSphere MQ V7.0.1, el valor de la propiedad <SubPoint> debe coincidir con el nombre de un punto de suscripción. Consulte Adición de un punto de suscripción .

< Filtro > (*MQPSC_FILTER*)
El valor es una expresión SQL que se utiliza como filtro en el contenido de los mensajes de publicación. Si una publicación en el tema especificado coincide con el filtro, se envía al suscriptor. Esta propiedad corresponde a la serie de selección que se utiliza en las llamadas MQSUB y MQOPEN. Para obtener más información, consulte Selección del contenido de un mensaje

Si se omite esta propiedad, no se realiza ningún filtrado de contenido.

<RegOpt> (*MQPSC_REGISTRATION_OPTION*)
Esta propiedad Opciones de registro puede tomar los valores siguientes:

AddName
(*MQPSC_ADD_NAME*)

Cuando se especifica para una suscripción existente que coincide con la identidad tradicional de este mandato Registrar suscripción, pero sin ningún valor SubName actual, el SubName especificado en este mandato se añade a la suscripción.

Si se especifica AddName , el campo SubName es obligatorio; de lo contrario, se devuelve MQRCCF_REG_OPTIONS_ERROR.

CorrelAsId
(*MQPSC_CORREL_ID_AS_IDENTITY*)

El CorrelId del descriptor de mensaje (MQMD) se utiliza al enviar publicaciones coincidentes a la cola de suscriptores. El CorrelId no debe ser cero,

FullResp
(*MQPSC_FULL_RESPONSE*)

Cuando se especifica, todos los atributos de la suscripción se devuelven en el mensaje de respuesta, si el mandato no falla.

FullResp sólo es válido cuando el mensaje de mandato hace referencia a una única suscripción. Por lo tanto, sólo se permite un tema en el mandato; de lo contrario, el mandato falla con el código de retorno *MQRCCF_REG_OPTIONS_ERROR*.

InformIfRet
(*MQPSC_INFORM_IF_RETAINED*)

El gestor de colas informa al suscriptor si una publicación se retiene cuando envía un mensaje de publicación en respuesta a un mensaje de mandato **Register Subscriber** o **Request Update** . El gestor de colas lo hace incluyendo la opción de publicación IsRetainedPub en el mensaje.

JoinExcl
(*MQPSC_JOIN_EXCLUSIVE*)

Esta opción indica que la SubIdentity especificada se debe añadir como miembro exclusivo del conjunto de identidades para la suscripción y que no se pueden añadir otras identidades al conjunto.

Si la identidad ya se ha unido a 'shared' y es la única entrada del conjunto, el conjunto se cambia a un bloqueo exclusivo mantenido por esta identidad. De lo contrario, si la suscripción tiene actualmente otras identidades en el conjunto de identidades (con acceso compartido), el mandato falla con el código de retorno *MQRCCF_SUBSCRIPTION_IN_USE*.

JoinShared
(*MQPSC_JOIN_SHARED*)

Esta opción indica que la SubIdentity especificada debe añadirse al conjunto de identidades para la suscripción.

Si la suscripción está bloqueada actualmente de forma exclusiva (utilizando la opción JoinExcl), el mandato falla con el código de retorno *MQRCCF_SUBSCRIPTION_LOCKED*, a menos que la identidad que tiene la suscripción bloqueada sea la misma que la de este mensaje de mandato. En este caso, el bloqueo se modifica automáticamente a un bloqueo compartido.

Local
(*MQPSC_LOCAL*)

La suscripción es local y no se distribuye a otros gestores de colas de la red. Las publicaciones realizadas en otros gestores de colas no se entregan a este suscriptor, a menos que también tenga una suscripción global correspondiente.

SóloNewPubs
(*MQPSC_NEW_PUBS_ONLY*)

Las publicaciones retenidas que existen en el momento en que se registra la suscripción no se envían al suscriptor; sólo se envían nuevas publicaciones.

Si un suscriptor se vuelve a registrar y cambia esta opción para que ya no se establezca, es posible que se vuelva a enviar una publicación que ya se le ha enviado.

NoAlter
(*MQPSC_NO_ALTER*)

Los atributos de una suscripción coincidente existente no se cambian.

Cuando se está creando una suscripción, esta opción se ignora. Todas las demás opciones especificadas se aplican a la nueva suscripción.

Si una `SubIdentity` también tiene especificada una de las opciones de unión (`JoinExcl` o `JoinShared`), la identidad se añade al conjunto de identidades independientemente de si se ha especificado `NoAlter` .

NO
(*MQPSC_NONE*)

Todas las opciones de registro toman sus valores predeterminados.

Si el suscriptor ya está registrado, sus opciones se restablecen a sus valores predeterminados (tenga en cuenta que esto *no* tiene el mismo efecto que omitir la propiedad de opciones de registro) y la caducidad de la suscripción se actualiza desde el MQMD del mensaje **Register Subscriber** .

Si se especifican otras opciones de registro al mismo tiempo, se ignora `Ninguna` .

NonPers
(*MQPSC_NON_PERSISTENT*)

Las publicaciones que coinciden con esta suscripción se entregan al suscriptor como mensajes no persistentes.

Pers
(*MQPSC_PERSISTENT*)

Las publicaciones que coinciden con esta suscripción se entregan al suscriptor como mensajes persistentes.

PersAsPub
(*MQPSC_PERSISTENT_AS_PUBLISH*)

Las publicaciones que coinciden con esta suscripción se entregan al suscriptor con la persistencia especificada por el publicador. Éste es el comportamiento predeterminado.

Cola dePersAs
(*MQPSC_PERSISTENT_AS_Q*)

Las publicaciones que coinciden con esta suscripción se entregan al suscriptor con la persistencia especificada en la cola de suscriptor.

PubOnReqOnly
(*MQPSC_PUB_ON_REQUEST_ONLY*)

El gestor de colas no envía publicaciones al suscriptor, excepto en respuesta a un mensaje de mandato **Request Update** .

VariableUserId
(*MQPSC_VARIABLE_USER_ID*)

Cuando se especifica, la identidad del suscriptor (cola, gestor de colas y correlid) no está restringida a un único ID de usuario. Esto difiere del comportamiento existente del gestor de colas que asocia el ID de usuario del mensaje de registro original con la identidad del suscriptor y, a partir de entonces, impide que cualquier otro usuario utilice dicha identidad. Si un nuevo suscriptor intenta utilizar la misma identidad, se devuelve *MQRCCF_DUPLICATE_SUBSCRIPTION* .

Esto permite a cualquier usuario modificar o anular el registro de la suscripción si el usuario tiene la autorización adecuada. Por lo tanto, no es necesario comprobar que el ID de usuario coincide con el del suscriptor original.

Para añadir esta opción a una suscripción existente, el mandato debe proceder del mismo ID de usuario que la propia suscripción original.

Si la suscripción del mandato **Request Update** tiene establecido `VariableUserId` , se debe establecer en el momento de la solicitud de actualización para indicar a qué suscripción se hace referencia. De lo contrario, el ID de usuario del mandato **Request Update** se utiliza para identificar la suscripción. Esto se altera temporalmente, junto con los otros identificadores de suscriptor, si se proporciona un nombre de suscripción.

Si un mensaje de mandato **Register Subscriber** sin este conjunto de opciones hace referencia a una suscripción existente que tiene este conjunto de opciones, la opción se elimina de esta suscripción y el ID de usuario de la suscripción se arregla ahora. Si ya existe un suscriptor que tiene la misma identidad (cola, gestor de colas e identificador de correlación) pero con un ID de usuario diferente asociado, el mandato falla con el código de retorno *MQRCCF_DUPLICATE_IDENTITY* porque sólo puede haber un ID de usuario asociado a una identidad de suscriptor.

Si se omite la propiedad de opciones de registro y el suscriptor ya está registrado, sus opciones de registro no se modifican y la caducidad de la suscripción se actualiza desde el MQMD del mensaje **Register Subscriber** .

Si el suscriptor todavía no está registrado, se crea una nueva suscripción con todas las opciones de registro que toman sus valores predeterminados.

Los valores predeterminados son `PersAsPub` y no se ha establecido ninguna otra opción.

<QMgrName> (*MQPSC_Q_MGR_NAME*)
El valor es el nombre del gestor de colas para la cola de suscriptor, a la que el gestor de colas envía las publicaciones coincidentes.

Si se omite esta propiedad, el valor predeterminado es el nombre del gestor de la cola de respuestas (`ReplyToQMgr`) que hay en el descriptor del mensaje (MQMD). Si el nombre resultante está en blanco, el valor predeterminado es `QMgrName`del gestor de colas.

< QName> (*MQPSC_Q_NAME*)
El valor es el nombre de la cola de suscriptores, a la que el gestor de colas envía las publicaciones coincidentes.

Si se omite esta propiedad, el valor predeterminado es el nombre `ReplyToQ` en el descriptor de mensaje (MQMD), que no debe estar en blanco en este caso.

Si la cola es una cola dinámica temporal, la entrega no persistente de publicaciones (`NonPers`) debe especificarse en la propiedad `<RegOpt>` .

Si la cola es una cola dinámica temporal, el gestor de colas anula automáticamente el registro de la suscripción cuando se cierra la cola.

<SubName> (*MQPSC_SUBSCRIPTION_NAME*)
Se trata de un nombre asignado a una suscripción determinada. Puede utilizarlo en lugar del gestor de colas, cola y correlId opcional para hacer referencia a una suscripción.

Si ya existe una suscripción con este **SubName**, cualquier otro atributo de la suscripción (Topic, QMgrName, QName, CorrelId, UserId, RegOpts, UserSubData y Caducidad) se altera temporalmente

con los atributos, si se especifican, que se pasan en el nuevo mensaje de mandato `Register Subscriber` . Sin embargo, si se utiliza **SubName** sin ningún campo QName especificado y se especifica una cola ReplyToen la cabecera MQMD, la cola de suscriptor cambia a ReplyToQ.

Si ya existe una suscripción que coincide con la identidad tradicional de este mandato, pero no tiene ningún **SubName**, el mandato Registration falla con el código de retorno *MQRCCF_DUPLICATE_SUBSCRIPTION*, a menos que se especifique la opción **AddName** .

Si intenta modificar una suscripción con nombre existente utilizando otro mandato `Register Subscriber` que especifique el mismo **SubName**, y los valores de Topic, QMgrName, QName y CorrelId en el nuevo mandato coinciden con una suscripción existente diferente, con o sin un SubName definido, el mandato falla con el código de retorno *MQRCCF_DUPLICATE_SUBSCRIPTION*. Esto impide que dos nombres de suscripción hagan referencia a la misma suscripción.

<SubIdentity> (*MQPSC_SUBSCRIPTION_IDENTITY*)
Esta serie se utiliza para representar una aplicación con un interés en una suscripción. Es una serie de caracteres de longitud variable con una longitud máxima de 64 caracteres y es opcional. El gestor de colas mantiene un conjunto de identidades de suscriptor para cada suscripción. Cada suscripción puede permitir que su conjunto de identidades contenga sólo una identidad o un número ilimitado de identidades (consulte las opciones **JoinShared** y **JoinExcl**).

Un mandato subscribe que especifica la opción **JoinShared** o **JoinExcl** añade la **SubIdentity** al conjunto de identidades de la suscripción, si todavía no está presente y si el conjunto de identidades existente permite dicha acción; es decir, ningún otro suscriptor se ha unido de forma exclusiva o el conjunto de identidades está vacío.

Cualquier modificación de los atributos de la suscripción como resultado de un mandato `Register Subscriber` en el que se especifica una **SubIdentity** , sólo se realiza correctamente si sería el único miembro del conjunto de identidades para esta suscripción. De lo contrario, el mandato falla con el código de retorno *MQRCCF_SUBSCRIPTION_IN_USE*. Esto impide que los atributos de una suscripción cambien sin que otros suscriptores interesados sean conscientes.

Si especifica una serie de caracteres de más de 64 caracteres, el mandato falla con el código de retorno *MQRCCF_SUB_IDENTITY_ERROR*.

< Datos deSubUser> (*MQPSC_SUBSCRIPTION_USER_DATA*)
Es una serie de texto de longitud variable. El valor lo almacena el gestor de colas con la suscripción, pero no influye en la entrega de publicaciones al suscriptor. El valor se puede modificar volviendo a registrar en la misma suscripción con un nuevo valor. Este atributo está ahí para el uso de la aplicación.

Los datos de **SubUser** se devuelven en la información metatópica (*MQCACF_REG_SUB_USER_DATA*) para una suscripción si está presente.

Si especifica más de uno de los valores de opción de registro `NonPers, PersAsPub, PersAsQueue, and Pers`, sólo se utiliza el último. No puede combinar estas opciones en una suscripción individual.

Ejemplo

A continuación se muestra un ejemplo de `NameValueData` para un mensaje de mandato **Register Subscriber** . En la aplicación de ejemplo, el servicio de resultados utiliza este mensaje para registrar una suscripción a los temas que contienen las últimas puntuaciones en todas las coincidencias, con la opción 'Persistente como publicación' establecida. La identidad del suscriptor, incluido el `CorrelId`, se toma de los valores predeterminados en MQMD.

```
<psc>
 <Command>RegSub</Command>
 <RegOpt>PersAsPub</RegOpt>
 <RegOpt>CorrelAsId</RegOpt>
 <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Mensaje de solicitud de actualización

El mensaje del mandato **Request Update** se envía desde un suscriptor a un gestor de colas, para solicitar las publicaciones retenidas actuales para el tema especificado y el punto de suscripción que coinciden con el filtro especificado (opcional).

Este mensaje se envía a *SYSTEM.BROKER.CONTROL.QUEUE*, la cola de control del gestor de colas. Se necesita autorización para colocar un mensaje en esta cola, además de autorización de acceso para el tema en la solicitud de actualización; esto lo establece el administrador del sistema del gestor de colas.

Este mandato se utiliza normalmente si el suscriptor ha especificado la opción PubOnReqOnly cuando se ha registrado. Si el gestor de colas tiene publicaciones retenidas coincidentes, se envían al suscriptor. Si el gestor de colas no tiene publicaciones retenidas coincidentes, la solicitud falla con el código de retorno *MQRCCF_NO_RETAINED_MSG*. El solicitante debe haber registrado previamente una suscripción con los mismos valores de Tema, SubPointy Filtro.

Propiedades

<Command> (*MQPSC_COMMAND*)
El valor es ReqUpdate (*MQPSC_REQUEST_UPDATE*). Esta propiedad ha de especificarse.

<Topic> (*MQPSC_TOPIC*)
El valor es el tema que solicita el suscriptor; se permiten caracteres comodín.

Esta propiedad debe especificarse, pero sólo se permite una aparición en este mensaje.

<SubPoint> (*MQPSC_SUBSCRIPTION_POINT*)
El valor es el punto de suscripción al que se adjunta la suscripción.

Si se omite esta propiedad, se utiliza el punto de suscripción predeterminado.

< Filtro > (*MQPSC_FILTER*)
El valor es una expresión ESQL que se utiliza como filtro en el contenido de los mensajes de publicación. Si una publicación en el tema especificado coincide con el filtro, se envía al suscriptor.

La propiedad < Filtro > debe tener el mismo valor que el especificado en la suscripción original para la que está solicitando una actualización.

Si se omite esta propiedad, no se realiza ningún filtrado de contenido.

<RegOpt> (*MQPSC_REGISTRATION_OPTION*)
La propiedad de opciones de registro puede tomar el valor siguiente:

CorrelAsId
(*MQPSC_CORREL_ID_AS_IDENTITY*)

El CorrelId del descriptor de mensaje (MQMD), que no debe ser cero, se utiliza al enviar publicaciones coincidentes a la cola de suscriptores.

NO
(*MQPSC_NONE*)

Todas las opciones toman sus valores predeterminados. Esto tiene el mismo efecto que omitir la propiedad <RegOpt> . Si se especifican otras opciones simultáneamente, se ignora None.

VariableUserId
(*MQPSC_VARIABLE_USER_ID*)

Cuando se especifica, la identidad del suscriptor (cola, gestor de colas y correlid) no está restringida a un único ID de usuario. Esto difiere del comportamiento existente del gestor de colas que asocia el ID de usuario del mensaje de registro original con la identidad del suscriptor y, a partir de entonces, impide que cualquier otro usuario utilice dicha identidad. Si un nuevo suscriptor intenta utilizar la misma identidad, el mandato falla con el código de retorno *MQRCCF_DUPLICATE_SUBSCRIPTION*.

Esto permite a cualquier usuario modificar o anular el registro de la suscripción cuando tenga la autorización adecuada. Por lo tanto, no es necesario comprobar que el ID de usuario coincide con el del suscriptor original.

Para añadir esta opción a una suscripción existente, el mandato debe proceder del mismo ID de usuario que la suscripción original.

Si la suscripción del mandato **Request Update** tiene establecido `VariableUserId` , se debe establecer en el momento de la solicitud de actualización para indicar a qué suscripción se hace referencia. De lo contrario, el ID de usuario del mandato **Request Update** se utiliza para identificar la suscripción. Esto se altera temporalmente, junto con los otros identificadores de suscriptor, si se proporciona un nombre de suscripción.

El valor predeterminado, si se omite esta propiedad, es que no se establecen opciones de registro.

<QMgrName> (*MQPSC_Q_MGR_NAME*)
El valor es el nombre del gestor de colas para la cola de suscriptor, a la que el gestor de colas envía la publicación retenida coincidente.

Si se omite esta propiedad, el valor predeterminado es el nombre del gestor de la cola de respuestas (`ReplyToQMgr`) que hay en el descriptor del mensaje (MQMD). Si el nombre resultante está en blanco, el valor predeterminado es `QMgrName`del gestor de colas.

< QName> (*MQPSC_Q_NAME*)
El valor es el nombre de la cola de suscriptores, a la que el gestor de colas envía la publicación retenida coincidente.

Si se omite esta propiedad, el valor predeterminado es el nombre `ReplyToQ` en el descriptor de mensaje (MQMD), que no debe estar en blanco en este caso.

<SubName> (*MQPSC_SUBSCRIPTION_NAME*)
Se trata de un nombre asignado a una suscripción determinada. Si se especifica en un mandato **Request Update** , el valor `SubName` tiene prioridad sobre todos los demás campos de identificador excepto el ID de usuario, a menos que `VariableUserId` esté establecido en la propia suscripción. Si no se ha establecido `VariableUserId` , el mandato *Request Update* sólo se ejecuta correctamente si el ID de usuario del mensaje de mandato coincide con el de la suscripción. Si el ID de usuario del mensaje de mandato no coincide con el de la suscripción, el mandato falla con el código de retorno *MQRCCF_DUPLICATE_IDENTITY*.

Si se ha establecido `VariableUserId` , y el ID de usuario difiere del de la suscripción, el mandato se ejecuta correctamente si el ID de usuario del nuevo mensaje de mandato tiene autorización para examinar la cola de secuencia y colocarla en la cola de suscriptor de la suscripción. De lo contrario, el mandato falla con el código de retorno *MQRCCF_NOT_AUTHORIZED*.

Si existe una suscripción que coincide con la identidad tradicional de este mandato, pero no tiene ningún `SubName`, el mandato**Request Update** falla con el código de retorno *MQRCCF_SUB_NAME_ERROR*.

Si se intenta solicitar una actualización para una suscripción que tiene un `SubName` utilizando un mensaje de mandato que coincide con la identidad tradicional, pero sin ningún `SubName` especificado, el mandato se ejecuta correctamente.

Ejemplo

A continuación se muestra un ejemplo de `NameValueData` para un mensaje de mandato **Request Update** . En la aplicación de ejemplo, el servicio de resultados utiliza este mensaje para solicitar publicaciones retenidas que contienen las puntuaciones más recientes para todos los equipos. La identidad del suscriptor, incluido el `CorrelId`, se toma de los valores predeterminados en MQMD.

```
<psc>
 <Command>ReqUpdate</Command>
 <RegOpt>CorrelAsId</RegOpt>
 <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Mensaje de respuesta del gestor de colas

Un mensaje **Queue Manager Response** se envía desde un gestor de colas a la cola ReplyTo de un publicador o un suscriptor, para indicar el éxito o el error de un mensaje de mandato recibido por el gestor de colas si el descriptor de mensaje de mandato especifica que se necesita una respuesta.

El mensaje de respuesta está contenido en el campo NameValueData de la cabecera MQRFH2, en una carpeta <pscr>.

En el caso de un aviso o error, el mensaje de respuesta contiene la carpeta <psc> del mensaje de mandato, así como la carpeta <pscr>. Los datos del mensaje, si los hay, no están contenidos en el mensaje de respuesta del gestor de colas. Si se trata de un error, ninguno de los mensajes que haya causado el error se procesará; si se trata de un aviso, es posible que alguno de los mensajes se haya procesado correctamente.

Si se produce una anomalía al enviar una respuesta:

- Para los mensajes de publicación, el gestor de colas intenta enviar la respuesta a la cola de mensajes no entregados de WebSphere MQ si MQPUT falla. Esto permite enviar la publicación a suscriptores incluso si la respuesta no puede devolver al publicador.
- Para otros mensajes o si la respuesta a la publicación no puede enviarse a la cola de mensajes no entregados, se anota un error y, normalmente, el mensaje se sustituye. El comportamiento depende de cómo se haya configurado el nodo MQInput.

Propiedades

< Final> (*MQPSCR_COMPLETE*)
Es el código de terminación, que puede tener uno de estos tres valores.

ok
El mandato ha terminado correctamente

aviso
El mandato ha terminado pero con un error

error
El comando ha fallado

< Responso> (*MQPSCR_RESPONSE*)
La respuesta a un mensaje de mandato, si dicho mandato ha devuelto un código de terminación aviso o error. Contiene una propiedad <Reason> y puede contener otras propiedades que indiquen la causa del aviso o error.

Si hay uno o más errores, habrá sólo una carpeta de respuesta que indicará únicamente la causa del primer error. En el caso de uno o más avisos, habrá una carpeta de respuesta para cada aviso.

< Razón> (*MQPSCR_REASON*)
Es el código de razón que califica al código de terminación, si el código de terminación es un aviso o error. Se establece en uno de los códigos de error listados en el ejemplo siguiente. La propiedad <Reason> está contenida en una carpeta <Response>. El código de razón puede ir seguido de cualquier propiedad válida de la carpeta <psc> (por ejemplo, un nombre de tema), que indique la causa del error o aviso. Si obtiene un código de razón de? ???, comprobar la exactitud de los datos, por ejemplo, corchetes coincidentes (< >).

Ejemplos

A continuación se muestran algunos ejemplos de NameValueData en un mensaje **Queue Manager Response** . Una respuesta satisfactoria podría ser la siguiente:

```
<pscr>
  <Completion>ok</Completion>
</pscr>
```

Este es un ejemplo de respuesta de anomalía; la anomalía es un error de filtro. La primera serie NameValueData contiene la respuesta; la segunda contiene el mandato original.

```
<pscr>
  <Completion>error</Completion>
  <Response>
    <Reason>3150</Reason>
  </Reponse>
</pscr>

<psc>
 ...
 command message (to which
 the queue manager is responding)
 ...
</psc>
```

Este es un ejemplo de respuesta de aviso (debida a temas no autorizados). La primera serie NameValueData contiene la respuesta; la segunda serie NameValueData contiene el mandato original.

```
<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>
```

Códigos de razón de publicación/suscripción

Estos códigos de razón se pueden devolver en el campo Razón de una carpeta <pscr> de respuesta de publicación/suscripción. También se listan las constantes que se pueden utilizar para representar estos códigos en los lenguajes de programación C o C++.

Las constantes MQRC_ requieren el archivo de cabecera WebSphere MQ cmqc.h . Las constantes MQRCCF_ requieren el archivo de cabecera WebSphere MQ cmqcfc.h (aparte de *MQRCCF_FILTER_ERROR* y *MQRCCF_INJU_USER*, que requieren el archivo de cabecera cmqpsc.h).

Código de razón y texto	Explicación	Emitido por
2336 MQRC_RFH_COMMAND_ERROR	Los valores válidos para el campo < Command> de una carpeta <psc> son: RegSub, DeregSub, Publish, DeletePuby ReqUpdate. Cualquier otro valor hace que se emita este código de error.	Cualquier mandato
2337 MQRC_RFH_ PARM_ERROR	Las carpetas <psc> y <mcd> tienen un conjunto de parámetros válidos que se pueden especificar dentro de ellas. Compruebe las descripciones de estas carpetas y asegúrese de que no ha especificado parámetros incorrectos.	Cualquier mandato

Código de razón y texto	Explicación	Emitido por
2338 MQRC_RFH_DUPLICATE_PARM	Algunos parámetros (por ejemplo, Tema) dentro de una carpeta <psc> se pueden repetir, pero otros (por ejemplo, Mandato) no se pueden repetir. Compruebe que no ha duplicado un parámetro no repetible.	Cualquier mandato
2339 MQRC_RFH_PARM_MISSING	Algunos parámetros de las carpetas <psc> o <mcd> son opcionales y se pueden omitir; algunos son obligatorios y no se deben omitir. Compruebe que ha incluido todos los parámetros obligatorios en las carpetas <psc> y <mcd> .	Cualquier mandato
2551 MQRC_SELECTION_NOT_AVAILABLE	No había ningún proveedor de selección de mensajes ampliado disponible para determinar qué suscriptores con un filtro especificado deben recibir la publicación.	Publicar, registrar suscriptor y solicitar actualización
	No había ningún proveedor de selección de mensajes ampliado disponible para manejar el filtro del suscriptor especificado.	Registrar suscriptor y solicitar actualización
2554 ERROR DE MQRC_CONTENT_	Un proveedor de selección de mensajes ampliados ha encontrado un error en la publicación actual o retenida.	Publicar y solicitar actualización
3008 MQRCCF_COMMAND_FAILED	Se ha producido un error interno que ha impedido que el mandato se ejecutara correctamente. El error puede producirse si se vuelve a emitir el mandato. El registro de sucesos del sistema para el gestor de colas contiene información que se debe utilizar al notificar el problema a IBM.	Cualquier mandato
3072 MQRCCF_TOPIC_ERROR	Uno o varios de los valores que ha proporcionado para el parámetro Topic son incorrectos. Compruebe que los valores de Topic se ajustan a las restricciones especificadas.	Cualquier mandato
3073 MQRCCF_NO_REGISTRADO	La combinación de SubPoint, Topic y Filter que ha especificado en el mandato DeregSub o ReqUpdate no era una combinación con la que se había registrado anteriormente o, para el mandato DeregSub si se ha especificado la opción DeregAll , no se ha utilizado una de las propiedades SubPoint, Topic o Filter para anular el registro de ninguna suscripción.	Mandatos de anulación de registro de suscriptor y solicitud de actualización

Código de razón y texto	Explicación	Emitido por
3074 MQRCCF_Q_MGR_NAME_ERROR	El gestor de colas especificado no era válido, o el gestor de colas no estaba disponible o no existía.	Mandatos Anular registro de suscriptor, Publicar, Registrar suscriptor y Solicitar actualización
3076 MQRCCF_Q_NAME_ERROR	El nombre de cola especificado no era válido o la cola no existía en el gestor de colas especificado.	Mandatos Anular registro de suscriptor, Publicar, Registrar suscriptor y Solicitar actualización
3077 MQRCCF_NO_RETAINED_MSG	No había mensajes retenidos para el tema que ha especificado. Esto puede ser o no un error, en función del diseño del programa de aplicación.	Mandato de solicitud de actualización
3079 MQRCCF_INCORRECT_Q	Los mandatos RegSub, DeregSuby ReqUpdate siempre se envían al SYSTEM.BROKER.CONTROL.QUEUE del gestor de colas para el que están destinados. Los mandatos de publicación y supresión se envían a la cola de entrada para el flujo de mensajes de publicación/suscripción concreto para el que están previstos; esto se determina cuando se diseña el flujo de mensajes. Este código de error se devuelve si se envía un mandato a la cola incorrecta.	Cualquier mandato
3080 MQRCCF_ID_CORREL_ERROR	Ha especificado CorrelAsId como uno de los parámetros RegOpt . Sin embargo, el campo CorrelId del MQMD no contiene un identificador de correlación válido (es decir, se establece en MQCI_NONE).	Mandatos de anulación de registro de suscriptor y registro de suscriptor
3081 MQRCCF_NO_AUTORIZADO	No tiene autorización para realizar la acción solicitada. Los valores de autorización para el gestor de colas los maneja el administrador del sistema utilizando el editor Jerarquía de temas.	Publicar y registrar mandatos de suscriptor
3083 MQRCCF_REG_OPTIONS_ERROR	Ha especificado un parámetro RegOpt no reconocido en la carpeta <psc> que contiene el mandato RegSub o DeregSub .	Mandatos de anulación de registro de suscriptor y registro de suscriptor
3084 MQRCCF_PUB_OPTIONS_ERROR	Ha especificado un parámetro PubOpt no reconocido en la carpeta <psc> que contiene el mandato Publish.	Mandato de publicación
3087 MQRCCF_DEL_OPTIONS_ERROR	Ha especificado un parámetro DelOpt no reconocido en la carpeta <psc> que contiene el mandato DeletePub .	Mandato Suprimir publicación

Código de razón y texto	Explicación	Emitido por
3150 MQRCCF_FILTER_ERROR	El valor especificado para el parámetro Filtro no es válido. Compruebe la sección que describe la sintaxis válida para las expresiones de filtro y asegúrese de que la expresión se ajusta.	Mandatos Anular registro de suscriptor, Registrar suscriptor y Solicitar actualización
3151 MQRCCF_USUARIO_INCORRECTO	Ya existe una suscripción que coincide con la especificada; sin embargo, la ha registrado un usuario diferente. Una suscripción sólo puede ser cambiada o anulada del registro por el usuario que la registró originalmente.	Mandatos Anular registro de suscriptor, Registrar suscriptor y Solicitar actualización
3152 MQRCCF_DUPLICATE_SUBSCRIPTION	Ya existe una suscripción coincidente con un nombre de suscripción diferente.	
3153 MQRCCF_SUB_NAME_ERROR	El formato del nombre de suscripción no es válido o ya existe una suscripción coincidente sin nombre de suscripción.	
3154 MQRCCF_SUB_IDENTITY_ERROR	El parámetro de identidad de suscripción contiene un error. El valor proporcionado supera la longitud máxima permitida, o la identidad de suscripción no es actualmente miembro del conjunto de identidades de la suscripción y no se ha especificado una opción de registro de unión.	
3155 MQRCCF_SUBSCRIPTION_IN_USE	Un miembro del conjunto de identidades ha intentado modificar o anular el registro de una suscripción cuando no era el único miembro de este conjunto.	
3156 MQRCCF_SUBSCRIPTION_LOCKED	La suscripción está actualmente bloqueada de modo exclusivo por otra identidad.	
3157 MQRCCF_ALREADY_JOINED	Se ha especificado una opción de registro de unión, pero la identidad del suscriptor ya era miembro del conjunto de identidades de la suscripción.	

Valores de MQMD en los mensajes de mandato para el gestor de colas

Las aplicaciones que envían mensajes de mandatos al gestor de colas utilizan los siguientes valores de campos en el descriptor de mensaje (MQMD). Los campos que se dejan como valor predeterminado, o que se pueden establecer en cualquier valor válido de la forma habitual, no se listan aquí.

Report
Consulte `MsgType` y `CorrelId`.

MsgType
MsgType debe establecerse en *MQMT_REQUEST* o *MQMT_DATAGRAM*. Se devolverá *MQRC_MSG_TYPE_ERROR* si MsgType no está establecido en uno de estos valores.

MsgType debe establecerse en *MQMT_REQUEST* para un mensaje de mandato si siempre se necesita una respuesta. Los distintivos MQRO_PAN y MQRO_NAN del campo Informe no son significativos en este caso.

Si MsgType se establece en *MQMT_DATAGRAM*, las respuestas dependen del valor de los distintivos MQRO_PAN y MQRO_NAN en el campo Informe :

- Sólo MQRO_PAN significa que el gestor de colas envía una respuesta sólo si el mandato se ejecuta correctamente.
- MQRO_NAN solo significa que el gestor de colas envía una respuesta sólo si el mandato falla.
- Si un mandato se completa con un aviso, se envía una respuesta si se establece MQRO_PAN o MQRO_NAN.
- MQRO_PAN + MQRO_NAN significa que el gestor de colas envía una respuesta tanto si el mandato se ejecuta correctamente como si falla. Esto tiene el mismo efecto desde la perspectiva del gestor de colas que establecer MsgType en MQMT_REQUEST.
- Si no se establece MQRO_PAN ni MQRO_NAN, nunca se envía ninguna respuesta.

Format
Establézcalo en MQFMT_RF_HEADER_2

MsgId
Este campo se establece normalmente en MQMI_NONE, de modo que el gestor de colas genera un valor exclusivo.

CorrelId
Este campo se puede establecer en cualquier valor. Si la identidad del remitente incluye un CorrelId, especifique este valor, junto con MQRO_PASS_CORREL_ID en el campo Informe , para asegurarse de que se establece en todos los mensajes de respuesta enviados por el gestor de colas al remitente.

ReplyToQ
Este campo define la cola a la que se enviarán las respuestas, si las hay. Puede ser la cola del remitente; esto tiene la ventaja de que el parámetro QName se puede omitir del mensaje. Sin embargo, si las respuestas se van a enviar a una cola diferente, se necesita el parámetro QName .

GestorColasRespuesta
Este campo define el gestor de colas para las respuestas. Si deja este campo en blanco (el valor predeterminado), el gestor de colas local pone su propio nombre en este campo.

Valores de MQMD para publicaciones reenviadas por un gestor de colas

Un gestor de colas utiliza estos valores de campos en el descriptor de mensaje (MQMD) cuando envía una publicación a un suscriptor. Todos los demás campos del MQMD se establecen en sus valores predeterminados.

Report
Informe se establece en MQRO_NONE.

MsgType
MsgType se establece en MQMT_DATAGRAM.

Caducidad
Caducidad se establece en el valor del mensaje Publish recibido del publicador. En el caso de un mensaje retenido, el tiempo pendiente se reduce en el tiempo aproximado que el mensaje ha estado en el gestor de colas.

Format
Formato se establece en MQFMT_RF_HEADER_2

MsgId
MsgId se establece en un valor exclusivo.

CorrelId
Si CorrelId forma parte de la identidad del suscriptor, este es el valor especificado por el suscriptor al registrarse. De lo contrario, es un valor distinto de cero elegido por el gestor de colas.

Priority
Prioridad toma el valor establecido por el publicador, o como resuelto si el publicador ha especificado MQPRI_PRIORITY_AS_Q_DEF.

Persistencia
Persistencia toma el valor establecido por el publicador, o como resuelto si el publicador ha especificado MQPER_PERSISTENCE_AS_Q_DEF, a menos que se especifique lo contrario en el mensaje Register Subscriber para el suscriptor al que se envía esta publicación.

ReplyToQ
ReplyToQ se establece en blancos.

GestorColasRespuesta
ReplyToQMgr se establece en el nombre del gestor de colas.

UserIdentifier
UserIdentifier es el identificador de usuario del suscriptor, tal como se establece cuando se registra el suscriptor.

AccountingToken
AccountingToken es la señal de contabilidad del suscriptor, tal como se ha establecido cuando el suscriptor se registró por primera vez.

ApplIdentityData
Datos deApplIdentity son los datos de identidad de aplicación del suscriptor, tal como se ha establecido cuando el suscriptor se registró por primera vez.

PutApplType
PutApplTipo se establece en MQAT_BROKER.

PutApplName
PutApplNombre se establece en los primeros 28 caracteres del nombre del gestor de colas.

PutDate
PutDate es la fecha en la que se ha colocado el mensaje.

PutTime
PutTime es la hora en que se ha colocado el mensaje.

ApplOriginData
ApplOriginDatos se establece en blancos.

Valores de MQMD en mensajes de respuesta del gestor de colas

Un gestor de colas utiliza estos valores de campos en el descriptor de mensaje (MQMD) al enviar una respuesta a un mensaje de publicación. Todos los demás campos del MQMD se establecen en sus valores predeterminados.

Report
Informe se establece en todos los ceros.

MsgType
MsgType se establece en MQMT_REPLY.

Format
Formato se establece en MQFMT_RF_HEADER_2

MsgId
El valor de MsgId depende de las opciones de Informe del mensaje de mandato original. De forma predeterminada, se establece en MQMI_NONE, para que el gestor de colas genere un valor exclusivo.

CorrelId
El valor de CorrelId depende de las opciones de Informe del mensaje de mandato original. De forma predeterminada, esto significa que el CorrelId se establece en el mismo valor que el MsgId del mensaje de mandato. Esto se puede utilizar para correlacionar mandatos con sus respuestas.

Priority
Prioridadse establece en el mismo valor que en el mensaje de mandato original.

Persistencia
Persistence se establece en el valor establecido en el mensaje de mandato original.

Caducidad
Caducidad se establece en el mismo valor que en el mensaje de mandato original recibido por el gestor de colas.

PutApplType
PutApplTipo se establece en MQAT_BROKER.

PutApplName
PutApplNombre se establece en los primeros 28 caracteres del nombre del gestor de colas.

Otros campos de contexto se establecen como si se hubieran generado con MQPMO_PASS_IDENTITY_CONTEXT.

Codificaciones de máquina

Esta sección describe la estructura del campo *Encoding* en el descriptor de mensaje.

Consulte “MQMD - Descriptor de mensaje” en la página 392 para obtener un resumen de los campos de la estructura.

El campo *Encoding* es un entero de 32 bits que se divide en cuatro subcampos separados; estos subcampos identifican:

- La codificación utilizada para enteros binarios
- La codificación utilizada para enteros de decimal empaquetado
- La codificación utilizada para números de coma flotante
- Bits reservados

Cada subcampo se identifica mediante una máscara de bits que tiene 1-bits en las posiciones correspondientes al subcampo y 0-bits en otro lugar. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Se definen las máscaras siguientes:

MÁSCARA_ENTERO_MQENC_MÁSCARA
Máscara para codificación de enteros binarios.

Este subcampo ocupa las posiciones de bits 28 a 31 dentro del campo *Encoding* .

MÁSCARA_DECIMAL_MQENC_MÁSCARA
Máscara para codificación de enteros decimales empaquetados.

Este subcampo ocupa las posiciones de bits 24 a 27 dentro del campo *Encoding* .

Máscara MQENC_FLOAT_MASK
Máscara para codificación de coma flotante.

Este subcampo ocupa las posiciones de bits 20 a 23 dentro del campo *Encoding* .

MÁSCARA_RESERVA_MQENC_RESERVADO
Máscara para bits reservados.

Este subcampo ocupa las posiciones de bits 0 a 19 dentro del campo *Encoding* .

Codificación de enteros binarios

Los valores siguientes son válidos para la codificación de entero binario:

MQENC_INTEGER_UNDEFINED
Los enteros binarios se representan utilizando una codificación que no está definida.

MQENC_INTEGER_NORMAL
Los enteros binarios se representan de la forma convencional:

- El byte menos significativo del número tiene la dirección más alta de cualquiera de los bytes del número; el byte más significativo tiene la dirección más baja
- El bit menos significativo de cada byte es adyacente al byte con la siguiente dirección superior; el bit más significativo de cada byte es adyacente al byte con la siguiente dirección inferior

MQENC_INTEGER_REVERSED
Los enteros binarios se representan de la misma forma que MQENC_INTEGER_NORMAL, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se organizan de la misma forma que MQENC_INTEGER_NORMAL.

Codificación de enteros decimales empaquetados

Los valores siguientes son válidos para la codificación de enteros decimales empaquetados:

MQENC_DECIMAL_UNDEFINED
Los enteros de decimal empaquetado se representan utilizando una codificación que no está definida.

MQENC_DECIMAL_NORMAL
Los enteros decimales empaquetados se representan de la forma convencional:

- Cada dígito decimal en la forma imprimible del número se representa en decimal empaquetado por un dígito hexadecimal único en el rango de X' 0 'a X' 9'. Cada dígito hexadecimal ocupa cuatro bits, por lo que cada byte del número decimal empaquetado representa dos dígitos decimales en la forma imprimible del número.
- El byte menos significativo del número decimal empaquetado es el byte que contiene el dígito decimal menos significativo. Dentro de ese byte, los cuatro bits más significativos contienen el dígito decimal menos significativo, y los cuatro bits menos significativos contienen el signo. El signo es X'C '(positivo), X 'D' (negativo) o X'F ' (sin signo).
- El byte menos significativo del número tiene la dirección más alta de cualquiera de los bytes del número; el byte más significativo tiene la dirección más baja.
- El bit menos significativo de cada byte es adyacente al byte con la siguiente dirección superior; el bit más significativo de cada byte es adyacente al byte con la siguiente dirección inferior.

MQENC_DECIMAL_REVERSED
Los enteros decimales empaquetados se representan de la misma forma que MQENC_DECIMAL_NORMAL, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se organizan de la misma forma que MQENC_DECIMAL_NORMAL.

Codificación de coma flotante

Los valores siguientes son válidos para la codificación de coma flotante:

MQENC_FLOAT_UNDEFINED
Los números de coma flotante se representan utilizando una codificación que no está definida.

MQENC_FLOAT_IEEE_NORMAL
Los números de coma flotante se representan utilizando el IEEE estándar[3]formato de coma flotante, con los bytes ordenados de la siguiente manera:

- El byte menos significativo de la mantisa tiene la dirección más alta de cualquiera de los bytes del número; el byte que contiene el exponente tiene la dirección más baja
- El bit menos significativo de cada byte es adyacente al byte con la siguiente dirección superior; el bit más significativo de cada byte es adyacente al byte con la siguiente dirección inferior

[3] El Instituto de Ingenieros Eléctricos y Electrónicos

Los detalles de la codificación flotante IEEE se pueden encontrar en el estándar IEEE 754.

MQENC_FLOAT_IEEE_REVERSED
Los números de coma flotante se representan de la misma forma que MQENC_FLOAT_IEEE_NORMAL, pero con los bytes ordenados en orden inverso. Los bits dentro de cada byte se organizan de la misma forma que MQENC_FLOAT_IEEE_NORMAL.

MQENC_FLOAT_S390
Los números de coma flotante se representan utilizando el formato de coma flotante estándar System/390 ; esto también lo utiliza System/370.

Construcción de codificaciones

Para construir un valor para el campo *Encoding* en MQMD, las constantes relevantes que describen las codificaciones necesarias pueden ser:

- Sumado, o
- Se combina utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit)

Sea cual sea el método utilizado, combine sólo una de las codificaciones MQENC_INTEGER_ * con una de las codificaciones MQENC_DECIMAL_ * y una de las codificaciones MQENC_FLOAT_ *.

Análisis de codificaciones

El campo *Encoding* contiene subcampos; debido a esto, las aplicaciones que necesitan examinar la codificación entera, decimal empaquetado o flotante deben utilizar una de las técnicas descritas.

Utilización de operaciones de bits

Si el lenguaje de programación da soporte a operaciones de bits, realice los pasos siguientes:

1. Seleccione uno de los valores siguientes, según el tipo de codificación necesario:
   - MQENC_INTEGER_MASK para la codificación de enteros binarios
   - MQENC_DECIMAL_MASK para la codificación de enteros empaquetados
   - MQENC_FLOAT_MASK para la codificación de coma flotante

   Llame al valor A.
2. Combine el campo *Encoding* con A utilizando la operación AND a nivel de bit; llame al resultado B.
3. B es la codificación necesaria y se puede probar la igualdad con cada uno de los valores válidos para ese tipo de codificación.

Utilización aritmética

Si el lenguaje de programación *no* da soporte a operaciones de bits, realice los pasos siguientes utilizando aritmética de enteros:

1. Seleccione uno de los valores siguientes, según el tipo de codificación necesario:
   - 1 para la codificación de enteros binarios
   - 16 para la codificación de enteros decimales empaquetados
   - 256 para la codificación de coma flotante

   Llame al valor A.
2. Divida el valor del campo *Encoding* por A; llame al resultado B.
3. Divida B por 16; llame al resultado C.
4. Multiplique C por 16 y resta de B; llame al resultado D.
5. Multiplique D por A; llame al resultado E.

6. E es la codificación necesaria y se puede probar la igualdad con cada uno de los valores válidos para ese tipo de codificación.

Resumen de codificaciones de arquitectura de máquina

Las codificaciones para arquitecturas de máquina se muestran en Tabla 578 en la página 884.

Tabla 578. Resumen de codificaciones para arquitecturas de máquina

Arquitectura de máquina	Codificación de enteros binarios	Codificación de enteros de decimal empaquetado	Codificación de coma flotante
IBM i	normal	normal	IEEE normal
Intel x86	reversed	reversed	IEEE invertido
PowerPC	normal	normal	IEEE normal
System/390	normal	normal	System/390

Opciones de informe y distintivos de mensaje

En esta sección se describen los campos *Report* y *MsgFlags* que forman parte del MQMD del descriptor de mensaje especificado en las llamadas MQGET, MQPUT y MQPUT1 .

Los temas de esta sección describen:

- La estructura del campo de informe y cómo lo procesa el gestor de colas
- Cómo analiza una aplicación el campo de informe
- La estructura del campo de distintivos de mensaje

Para obtener más información sobre el descriptor de mensaje MQMD, consulte “MQMD - Descriptor de mensaje” en la página 392.

Estructura del campo de informe

Esta información describe la estructura del campo de informe.

El campo *Report* es un entero de 32 bits que se divide en tres subcampos separados. Estos subcampos identifican:

- Opciones de informe que se rechazan si el gestor de colas local no las reconoce
- Opciones de informe que siempre se aceptan, incluso si el gestor de colas local no las reconoce
- Opciones de informe que sólo se aceptan si se cumplen otras condiciones

Cada subcampo se identifica mediante una máscara de bits que tiene 1-bits en las posiciones correspondientes al subcampo y 0-bits en otro lugar. Los bits de un subcampo no son necesariamente adyacentes. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Las máscaras siguientes están definidas para identificar los subcampos:

MQRO_REJECT_UNSUP_MASK
Esta máscara identifica las posiciones de bits dentro del campo *Report* donde las opciones de informe no soportadas por el gestor de colas local hacen que la llamada MQPUT o MQPUT1 falle con el código de terminación MQCC_FAILED y el código de razón MQRC_REPORT_OPTIONS_ERROR.

Este subcampo ocupa las posiciones de bits 3 y 11 a 13.

MQRO_ACCEPT_UNSUP_MASK
Esta máscara identifica las posiciones de bits dentro del campo *Report* donde las opciones de informe que no están soportadas por el gestor de colas local se aceptan en las llamadas MQPUT o MQPUT1 . En este caso, se devuelve el código de terminación MQCC_WARNING con el código de razón MQRC_UNKNOWN_REPORT_OPTION.

Este subcampo ocupa las posiciones de bits 0 a 2, 4 a 10 y 24 a 31.

En este subcampo se incluyen las siguientes opciones de informe:

- MQRO_ACTIVITY
- MQRO_COPY_MSG_ID_TO_CORREL_ID
- MQRO_DEAD_LETTER_Q
- MQRO_DISCARD_MSG
- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA
- MQRO_EXPIRATION
- MQRO_EXPIRATION_WITH_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA
- MQRO_NAN
- MQRO_NEW_MSG_ID
- MQRO_NONE
- MQRO_PAN
- MQRO_PASS_CORREL_ID
- MQRO_PASS_MSG_ID

MQRO_ACCEPT_UNSUP_IF_XMIT_MASK
Esta máscara identifica las posiciones de bits dentro del campo *Report* donde las opciones de informe que no están soportadas por el gestor de colas local se aceptan en las llamadas MQPUT o MQPUT1 *siempre que* se cumplan las dos condiciones siguientes:

- El mensaje está destinado a un gestor de colas remoto.
- La aplicación no coloca el mensaje directamente en una cola de transmisión local (es decir, la cola identificada por los campos *ObjectQMgrName* y *ObjectName* en el descriptor de objeto especificado en la llamada MQOPEN o MQPUT1 no es una cola de transmisión local).

Se devuelve el código de terminación MQCC_WARNING con el código de razón MQRC_UNKNOWN_REPORT_OPTION si se cumplen estas condiciones y MQCC_FAILED con el código de razón MQRC_REPORT_OPTIONS_ERROR si no es así.

Este subcampo ocupa las posiciones de bits 14 a 23.

En este subcampo se incluyen las siguientes opciones de informe:

- MQRO_COA
- MQRO_COA_WITH_DATA
- MQRO_COA_WITH_FULL_DATA
- MQRO_COD
- MQRO_COD_WITH_DATA
- MQRO_COD_WITH_FULL_DATA

Si se especifica alguna opción en el campo *Report* que el gestor de colas no reconoce, el gestor de colas comprueba cada subcampo a su vez utilizando la operación AND a nivel de bit para combinar el campo *Report* con la máscara para ese subcampo. Si el resultado de esa operación no es cero, se devuelven el código de terminación y los códigos de razón descritos anteriormente.

Si se devuelve MQCC_WARNING, no se define qué código de razón se devuelve si existen otras condiciones de aviso.

La posibilidad de especificar y tener opciones de informe aceptadas que no son reconocidas por el gestor de colas local es útil cuando se envía un mensaje con una opción de informe que es reconocida y procesada por un gestor de colas *remoto* .

Análisis del campo de informe

El campo *Report* contiene subcampos; debido a esto, las aplicaciones que necesitan comprobar si el remitente del mensaje ha solicitado un informe determinado deben utilizar una de las técnicas descritas.

Utilización de operaciones de bits

Si el lenguaje de programación da soporte a operaciones de bits, realice los pasos siguientes:

1. Seleccione uno de los valores siguientes, de acuerdo con el tipo de informe que se va a comprobar:
   - Informe MQRO_COA_WITH_FULL_DATA para COA
   - Informe MQRO_COD_WITH_FULL_DATA para COD
   - MQRO_EXCEPTION_WITH_FULL_DATA para informe de excepciones
   - MQRO_EXPIRATION_WITH_FULL_DATA para informe de caducidad

   Llame al valor A.

   En z/OS, utilice los valores MQRO_ * _WITH_DATA en lugar de los valores MQRO_ * _WITH_FULL_DATA.
2. Combine el campo *Report* con A utilizando la operación AND a nivel de bit; llame al resultado B.
3. Pruebe B para la igualdad con cada valor que sea posible para ese tipo de informe.

   Por ejemplo, si A es MQRO_EXCEPTION_WITH_FULL_DATA, pruebe la igualdad de B con cada uno de los siguientes para determinar qué ha especificado el remitente del mensaje:
   - MQRO_NONE
   - MQRO_EXCEPTION
   - MQRO_EXCEPTION_WITH_DATA
   - MQRO_EXCEPTION_WITH_FULL_DATA

   Las pruebas se pueden realizar en el orden que sea más conveniente para la lógica de la aplicación.

Utilice un método similar para probar las opciones MQRO_PASS_MSG_ID o MQRO_PASS_CORREL_ID; seleccione como valor A cualquiera de estas dos constantes y, a continuación, continúe como se ha descrito anteriormente.

Utilización aritmética

Si el lenguaje de programación *no* da soporte a operaciones de bits, realice los pasos siguientes utilizando aritmética de enteros:

1. Seleccione uno de los valores siguientes, de acuerdo con el tipo de informe que se va a comprobar:
   - Informe MQRO_COA para COA
   - Informe MQRO_COD para COD
   - MQRO_EXCEPTION para el informe de excepciones
   - MQRO_EXPIRATION para informe de caducidad

   Llame al valor A.
2. Divida el campo *Report* por A; llame al resultado B.
3. Divida B por 8; llame al resultado C.
4. Multiplique C por 8 y resta de B; llame al resultado D.
5. Multiplique D por A; llame al resultado E.

6. Pruebe E para la igualdad con cada valor que sea posible para ese tipo de informe.

   Por ejemplo, si A es MQRO_EXCEPTION, pruebe la igualdad de E con cada uno de los siguientes para determinar qué ha especificado el remitente del mensaje:

   - MQRO_NONE
   - MQRO_EXCEPTION
   - MQRO_EXCEPTION_WITH_DATA
   - MQRO_EXCEPTION_WITH_FULL_DATA

   Las pruebas se pueden realizar en el orden que sea más conveniente para la lógica de la aplicación.

El pseudocódigo siguiente ilustra esta técnica para los mensajes de informe de excepción:

```
A = MQRO_EXCEPTION
B = Report/A
C = B/8
D = B - C*8
E = D*A
```

Utilice un método similar para probar las opciones MQRO_PASS_MSG_ID o MQRO_PASS_CORREL_ID; seleccione como valor A cualquiera de estas dos constantes y, a continuación, continúe como se ha descrito anteriormente, pero sustituyendo el valor 8 en los pasos anteriores por el valor 2.

Estructura del campo de distintivos de mensaje

Esta información describe la estructura del campo de distintivos de mensaje.

El campo *MsgFlags* es un entero de 32 bits que se divide en tres subcampos separados. Estos subcampos identifican:

- Distintivos de mensaje que se rechazan si el gestor de colas local no los reconoce
- Distintivos de mensajes que siempre se aceptan, incluso si el gestor de colas local no los reconoce
- Distintivos de mensaje que se aceptan sólo si se cumplen otras condiciones

Nota: Todos los subcampos de *MsgFlags* están reservados para que los utilice el gestor de colas.

Cada subcampo se identifica mediante una máscara de bits que tiene 1-bits en las posiciones correspondientes al subcampo y 0-bits en otro lugar. Los bits están numerados de tal manera que el bit 0 es el bit más significativo, y el bit 31 el bit menos significativo. Las máscaras siguientes están definidas para identificar los subcampos:

MQMF_REJECT_UNSUP_MASK
Esta máscara identifica las posiciones de bits dentro del campo *MsgFlags* donde los distintivos de mensaje que no están soportados por el gestor de colas local hacen que la llamada MQPUT o MQPUT1 falle con el código de terminación MQCC_FAILED y el código de razón MQRC_MSG_FLAGS_ERROR.

Este subcampo ocupa las posiciones de bits 20 a 31.

En este subcampo se incluyen los distintivos de mensaje siguientes:

- MQMF_LAST_MSG_IN_GROUP
- MQMF_LAST_SEGMENT
- MQMF_MSG_IN_GROUP
- SEGMENTO MQMF_SEGMENT
- MQMF_SEGMENTATION_ALLOWED
- MQMF_SEGMENTATION_INHIBIDO

MQMF_ACCEPT_UNSUP_MASK
Esta máscara identifica las posiciones de bits dentro del campo *MsgFlags* donde los distintivos de mensaje que no están soportados por el gestor de colas local se aceptan en las llamadas MQPUT o MQPUT1 . El código de terminación es MQCC_OK.

Este subcampo ocupa las posiciones de bits 0 a 11.

MQMF_ACCEPT_UNSUP_IF_XMIT_MASK
Esta máscara identifica las posiciones de bits dentro del campo *MsgFlags* donde los distintivos de mensaje que no están soportados por el gestor de colas local se aceptan en las llamadas MQPUT o MQPUT1 *siempre que* se cumplan las dos condiciones siguientes:

- El mensaje está destinado a un gestor de colas remoto.
- La aplicación no coloca el mensaje directamente en una cola de transmisión local (es decir, la cola identificada por los campos *ObjectQMgrName* y *ObjectName* en el descriptor de objeto especificado en la llamada MQOPEN o MQPUT1 no es una cola de transmisión local).

Se devuelve el código de terminación MQCC_OK si se cumplen estas condiciones y MQCC_FAILED con el código de razón MQRC_MSG_FLAGS_ERROR si no es así.

Este subcampo ocupa las posiciones de bits 12 a 19.

Si hay distintivos especificados en el campo *MsgFlags* que el gestor de colas no reconoce, el gestor de colas comprueba cada subcampo a su vez utilizando la operación AND a nivel de bit para combinar el campo *MsgFlags* con la máscara para ese subcampo. Si el resultado de esa operación no es cero, se devuelven el código de terminación y los códigos de razón descritos anteriormente.

salida de conversión de datos

Esta colección de temas describe la interfaz para la salida de conversión de datos y el proceso realizado por el gestor de colas cuando es necesaria la conversión de datos.

Para obtener más información sobre la conversión de datos, consulte *Conversión de datos en WebSphere MQ* en https://www.ibm.com/support/docview.wss?uid=swg27005729.

La salida de conversión de datos se invoca como parte del proceso de la llamada MQGET para convertir los datos del mensaje de aplicación a la representación que necesita la aplicación receptora. La conversión de los datos del mensaje de aplicación es opcional; requiere que se especifique la opción MQGMO_CONVERT en la llamada MQGET.

Se describen los siguientes temas:

- El proceso realizado por el gestor de colas en respuesta a la opción MQGMO_CONVERT; consulte “Proceso de conversión” en la página 888.
- Convenios de proceso utilizados por el gestor de colas al procesar un formato incorporado; estos convenios también se recomiendan para las salidas escritas por el usuario. Consulte “Convenios de proceso” en la página 890.
- Consideraciones especiales para convertir mensajes de informe; consulte “Conversión de mensajes de informe” en la página 894.
- Los parámetros pasados a la salida de conversión de datos; consulte “MQ_DATA_CONV_EXIT-Salida de conversión de datos” en la página 907.
- Una llamada que se puede utilizar desde la salida para convertir datos de caracteres entre distintas representaciones; consulte “MQXCNVC-Convertir caracteres” en la página 901.
- El parámetro de estructura de datos que es específico de la salida; consulte “MQDXP-Parámetro de salida de conversión de datos” en la página 895.

Proceso de conversión

Esta información describe el proceso realizado por el gestor de colas en respuesta a la opción MQGMO_CONVERT.

El gestor de colas realiza las acciones siguientes si se especifica la opción MQGMO_CONVERT en la llamada MQGET y hay un mensaje que se debe devolver a la aplicación:

1. Si se cumple una o más de las condiciones siguientes, no es necesaria ninguna conversión:

- Los datos del mensaje ya están en el juego de caracteres y la codificación que necesita la aplicación que emite la llamada MQGET. La aplicación debe establecer los campos *CodedCharSetId* y *Encoding* en el parámetro *MsgDesc* de la llamada MQGET en los valores necesarios, antes de emitir la llamada.
- La longitud de los datos del mensaje es cero.
- La longitud del parámetro *Buffer* de la llamada MQGET es cero.

En estos casos, el mensaje se devuelve sin conversión a la aplicación que emite la llamada MQGET; los valores *CodedCharSetId* y *Encoding* del parámetro *MsgDesc* se establecen en los valores de la información de control del mensaje y la llamada se completa con una de las siguientes combinaciones de código de terminación y código de razón:

Código de terminación	**Código de razón**
MQCC_OK	MQRC_NONE
MQCC_WARNING	MQRC_TRUNCATED_MSG_ACCEPTED
MQCC_WARNING	MQRC_TRUNCATED_MSG_FAILED

Los pasos siguientes sólo se realizan si el juego de caracteres o la codificación de los datos del mensaje difiere del valor correspondiente en el parámetro *MsgDesc* y hay datos que se deben convertir:

2. Si el campo *Format* de la información de control del mensaje tiene el valor MQFMT_NONE, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_FORMAT_ERROR.

   En todos los demás casos, el proceso de conversión continúa.
3. El mensaje se elimina de la cola y se coloca en un almacenamiento intermedio temporal que tiene el mismo tamaño que el parámetro *Buffer* . Para las operaciones de examen, el mensaje se copia en el almacenamiento intermedio temporal, en lugar de eliminarse de la cola.
4. Si el mensaje debe truncarse para que quepa en el almacenamiento intermedio, se realiza lo siguiente:
   - Si la opción MQGMO_ACCEPT_TRUNCATED_MSG *no* se ha especificado, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_FAILED.
   - Si se ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG , el código de terminación se establece en MQCC_WARNING, el código de razón se establece en MQRC_TRUNCATED_MSG_ACCEPTED y el proceso de conversión continúa.
5. Si el mensaje se puede acomodar en el almacenamiento intermedio sin truncamiento, o se ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG, se hace lo siguiente:
   - Si el formato es un formato incorporado, el almacenamiento intermedio se pasa al servicio de conversión de datos del gestor de colas.
   - Si el formato no es un formato incorporado, el almacenamiento intermedio se pasa a una salida escrita por el usuario con el mismo nombre que el formato. Si no se puede encontrar la salida, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_FORMAT_ERROR.

   Si no se produce ningún error, la salida del servicio de conversión de datos o de la salida escrita por el usuario es el mensaje convertido, más el código de terminación y el código de razón que se devolverá a la aplicación que emite la llamada MQGET.
6. Si la conversión es satisfactoria, el gestor de colas devuelve el mensaje convertido a la aplicación. En este caso, el código de terminación y el código de razón devueltos por la llamada MQGET son una de las combinaciones siguientes:

Código de terminación	**Código de razón**
MQCC_OK	MQRC_NONE

Código de terminación	Código de razón
MQCC_WARNING	MQRC_TRUNCATED_MSG_ACCEPTED

Sin embargo, si la conversión la realiza una salida escrita por el usuario, se pueden devolver otros códigos de razón, incluso cuando la conversión es satisfactoria.

Si la conversión falla, el gestor de colas devuelve el mensaje sin convertir a la aplicación, con los campos *CodedCharSetId* y *Encoding* del parámetro *MsgDesc* establecidos en los valores de la información de control del mensaje y con el código de terminación MQCC_WARNING.

Convenios de proceso

Al convertir un formato incorporado, el gestor de colas sigue los convenios de proceso descritos.

Las salidas escritas por el usuario también deben seguir estas convenciones, aunque el gestor de colas no las aplica. Los formatos incorporados convertidos por el gestor de colas son:

- MQFMT_ADMIN
- MQFMT_CICS (solo z/OS)
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_DIST_HEADER
- MQFMT_EVENT versión 1
- MQFMT_EVENT versión 2
- MQFMT_IMS
- MQFMT_IMS_VAR_STRING
- MQFMT_MD_EXTENSION
- 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. Si el mensaje se expande durante la conversión y supera el tamaño del parámetro *Buffer* , se realiza lo siguiente:
   - Si la opción MQGMO_ACCEPT_TRUNCATED_MSG *no* se ha especificado, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_CONVERTED_MSG_TOO_BIG.
   - Si se especifica la opción MQGMO_ACCEPT_TRUNCATED_MSG *era* , el mensaje se trunca, el código de terminación se establece en MQCC_WARNING, el código de razón se establece en MQRC_TRUNCATED_MSG_ACCEPTED y el proceso de conversión continúa.
2. Si se produce un truncamiento (antes o durante la conversión), el número de bytes válidos devueltos en el parámetro *Buffer* puede ser *menor que* la longitud del almacenamiento intermedio.

   Esto puede ocurrir, por ejemplo, si un entero de 4 bytes o un carácter DBCS se encuentra entre los extremos del almacenamiento intermedio. El elemento de información incompleto no se convierte y los bytes del mensaje devuelto no contienen información válida. Esto también puede ocurrir si un mensaje que se ha truncado antes de la conversión se reduce durante la conversión.

Si el número de bytes válidos devueltos es menor que la longitud del almacenamiento intermedio, los bytes no utilizados al final del almacenamiento intermedio se establecen en nulos.

3. Si una matriz o serie se encuentra entre el final del almacenamiento intermedio, se convierten tantos datos como sea posible; sólo no se convierte el elemento de matriz concreto o el carácter DBCS que está incompleto; se convierten los elementos o caracteres de matriz anteriores.
4. Si se produce un truncamiento (antes o durante la conversión), la longitud devuelta para el parámetro *DataLength* es la longitud del mensaje *unconvert* antes del truncamiento.
5. Cuando las series se convierten entre juegos de caracteres de un solo byte (SBCS), juegos de caracteres de doble byte (DBCS) o juegos de caracteres de varios bytes (MBCS), las series pueden expandirse o contraerse.
   - En los formatos PCF MQFMT_ADMIN, MQFMT_EVENT y MQFMT_PCF, las series de las estructuras MQCFST y MQCFSL se expanden o contraen según sea necesario para acomodar la serie después de la conversión.

     Para la estructura de lista de series MQCFSL, las series de la lista pueden expandirse o contraerse en cantidades diferentes. Si esto sucede, el gestor de colas rellena las series más cortas con espacios en blanco para que tengan la misma longitud que la serie más larga después de la conversión.
   - En el formato MQFMT_REF_MSG_HEADER, las series a las que se dirigen los campos *SrcEnvOffset*, *SrcNameOffset*, *DestEnvOffset*y *DestNameOffset* se expanden o contraen según sea necesario para acomodar las series después de la conversión.
   - En el formato MQFMT_RF_HEADER, el campo *NameValueString* se expande o contrae según sea necesario para acomodar los pares de nombre/valor después de la conversión.
   - En estructuras con tamaños de campo fijos, el gestor de colas permite que las series se expandan o se contraigan dentro de sus campos fijos, siempre que no se pierda información significativa. En este sentido, los espacios en blanco finales y los caracteres que siguen al primer carácter nulo del campo se tratan como insignificantes.
     - Si la serie se expande, pero sólo es necesario descartar caracteres insignificantes para acomodar la serie convertida en el campo, la conversión se realiza correctamente y la llamada se completa con MQCC_OK y el código de razón MQRC_NONE (suponiendo que no haya otros errores).
     - Si la serie se expande, pero la serie convertida requiere que se descarten caracteres significativos para que quepan en el campo, el mensaje se devuelve sin convertir y la llamada se completa con MQCC_WARNING y el código de razón MQRC_CONVERTED_STRING_TOO_BIG.

       Nota: El código de razón MQRC_CONVERTED_STRING_TOO_BIG da como resultado en este caso si se ha especificado o no la opción MQGMO_ACCEPT_TRUNCATED_MSG.
     - Si la serie se contrae, el gestor de colas rellena la serie con espacios en blanco hasta la longitud del campo.
6. Para los mensajes que constan de una o más estructuras de cabecera MQ seguidas de datos de usuario, es posible que se conviertan una o más de las estructuras de cabecera, mientras que el resto del mensaje no lo es. Sin embargo, (con dos excepciones) los campos *CodedCharSetId* y *Encoding* de cada estructura de cabecera siempre indican correctamente el juego de caracteres y la codificación de los datos que siguen a la estructura de cabecera.

   Las dos excepciones son las estructuras MQCIH y MQIIH, donde los valores de los campos *CodedCharSetId* y *Encoding* de esas estructuras no son significativos. Para esas estructuras, los datos que siguen a la estructura están en el mismo juego de caracteres y codificación que la propia estructura MQCIH o MQIIH.
7. Si los campos *CodedCharSetId* o *Encoding* de la información de control del mensaje que se está recuperando, o en el parámetro *MsgDesc* , especifican valores que no están definidos o no están soportados, el gestor de colas puede ignorar el error si no es necesario utilizar el valor no definido o no soportado al convertir el mensaje.

   Por ejemplo, si el campo *Encoding* del mensaje especifica una codificación flotante no soportada, pero el mensaje contiene sólo datos enteros, o contiene datos de coma flotante que no requieren

conversión (porque las codificaciones flotante de origen y destino son idénticas), es posible que el error no se diagnostique.

Si se diagnostica el error, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y uno de los códigos de razón MQRC_SOURCE_ * _ERROR o MQRC_TARGET_ * _ERROR (según corresponda); los campos *CodedCharSetId* y *Encoding* del parámetro *MsgDesc* se establecen en los valores de la información de control del mensaje.

Si el error no se diagnostica y la conversión se completa correctamente, los valores devueltos en los campos *CodedCharSetId* y *Encoding* en el parámetro *MsgDesc* son los especificados por la aplicación que emite la llamada MQGET.

8. En todos los casos, si el mensaje se devuelve a la aplicación sin convertir, el código de finalización se establece en MQCC_WARNING, y los campos *CodedCharSetId* y *Encoding* del parámetro *MsgDesc* se establecen en los valores adecuados para los datos sin convertir. Esto también se realiza para MQFMT_NONE.

   El parámetro *Reason* se establece en un código que indica por qué no se ha podido llevar a cabo la conversión, a menos que el mensaje también se haya tenido que truncar; los códigos de razón relacionados con el truncamiento tienen prioridad sobre los códigos de razón relacionados con la conversión. (Para determinar si se ha convertido un mensaje truncado, compruebe los valores devueltos en los campos *CodedCharSetId* y *Encoding* en el parámetro *MsgDesc* .)

   Cuando se diagnostica un error, se devuelve un código de razón específico o el código de razón general MQRC_NOT_CONVERT. El código de razón devuelto depende de las prestaciones de diagnóstico del servicio de conversión de datos subyacente.

9. Si se devuelve el código de terminación MQCC_WARNING y hay más de un código de razón relevante, el orden de prioridad es el siguiente:

   a. Las razones siguientes tienen prioridad sobre todas las demás; sólo puede surgir una de las razones de este grupo:

      - MQRC_SIGNAL_REQUEST_ACCEPTED
      - MQRC_TRUNCATED_MSG_ACCEPTED

   b. El orden de prioridad dentro de los códigos de razón restantes no está definido.

10. Al finalizar la llamada MQGET:

    - El siguiente código de razón indica que el mensaje se ha convertido correctamente:
      - MQRC_NONE
    - Los siguientes códigos de razón indican que el mensaje *podría* haberse convertido correctamente (compruebe los campos *CodedCharSetId* y *Encoding* en el parámetro *MsgDesc* para averiguarlo):
      - MQRC_MSG_MARKED_BROWSE_CO_OP
      - MQRC_TRUNCATED_MSG_ACCEPTED
    - Los demás códigos de razón indican que el mensaje no se ha convertido.

    El siguiente proceso es específico de los formatos incorporados; no se aplica a los formatos definidos por el usuario:

11. Con la excepción de los formatos siguientes:

    - MQFMT_ADMIN
    - MQFMT_COMMAND_1
    - MQFMT_COMMAND_2
    - MQFMT_EVENT
    - MQFMT_IMS_VAR_STRING
    - MQFMT_PCF
    - MQFMT_STRING

ninguno de los formatos incorporados se puede convertir de o a conjuntos de caracteres que no tengan caracteres SBCS para los caracteres que son válidos en los nombres de cola. Si se intenta realizar una conversión de este tipo, el mensaje se devuelve sin convertir, con el código de terminación MQCC_WARNING y el código de razón MQRC_SOURCE_CCSID_ERROR o MQRC_TARGET_CCSID_ERROR, según corresponda.

El juego de caracteres Unicode UCS-2 es un ejemplo de un juego de caracteres que no tiene caracteres SBCS para los caracteres que son válidos en los nombres de cola.

12. Si los datos del mensaje para un formato incorporado se truncan, los campos del mensaje que contienen longitudes de series, o recuentos de elementos o estructuras, *no* se ajustan para reflejar la longitud de los datos realmente devueltos a la aplicación; los valores devueltos para dichos campos dentro de los datos del mensaje son los valores aplicables al mensaje *antes del truncamiento*.

    Al procesar mensajes como, por ejemplo, un mensaje MQFMT_ADMIN truncado, asegúrese de que la aplicación no intenta acceder a los datos más allá del final de los datos devueltos.

13. Si el nombre de formato es MQFMT_DEAD_LETTER_HEADER, los datos de mensaje empiezan con una estructura MQDLH, posiblemente seguida de cero o más bytes de datos de mensaje de aplicación. El formato, el juego de caracteres y la codificación de los datos del mensaje de aplicación se definen mediante los campos *Format*, *CodedCharSetId*y *Encoding* en la estructura MQDLH al principio del mensaje. Debido a que la estructura MQDLH y los datos de mensajes de aplicación pueden tener diferentes conjuntos de caracteres y codificaciones, uno, otro o ambos de la estructura MQDLH y los datos de mensajes de aplicación pueden requerir conversión.

    El gestor de colas convierte primero la estructura MQDLH, según sea necesario. Si la conversión es satisfactoria, o la estructura MQDLH no requiere conversión, el gestor de colas comprueba los campos *CodedCharSetId* y *Encoding* de la estructura MQDLH para ver si es necesaria la conversión de los datos del mensaje de aplicación. Si la conversión *es* necesaria, el gestor de colas invoca la salida escrita por el usuario con el nombre proporcionado por el campo *Format* en la estructura MQDLH, o realiza la propia conversión (si *Format* es el nombre de un formato incorporado).

    Si la llamada MQGET devuelve un código de terminación de MQCC_WARNING, y el código de razón es uno de los que indica que la conversión no ha sido satisfactoria, se aplica uno de los siguientes:

    - No se ha podido convertir la estructura MQDLH. En este caso, los datos del mensaje de aplicación tampoco se habrán convertido.
    - La estructura MQDLH se ha convertido, pero los datos del mensaje de aplicación no.

    La aplicación puede examinar los valores devueltos en los campos *CodedCharSetId* y *Encoding* en el parámetro *MsgDesc* , y los de la estructura MQDLH, para determinar cuál de los anteriores se aplica.

14. Si el nombre de formato es MQFMT_XMIT_Q_HEADER, los datos del mensaje empiezan con una estructura MQXQH, posiblemente seguida de cero o más bytes de datos adicionales. Estos datos adicionales suelen ser los datos de mensaje de aplicación (que pueden ser de longitud cero), pero también puede haber una o más estructuras de cabecera de MQ adicionales presentes, al principio de los datos adicionales.

    La estructura MQXQH debe estar en el juego de caracteres y la codificación del gestor de colas. El formato, el juego de caracteres y la codificación de los datos que siguen a la estructura MQXQH se proporcionan mediante los campos *Format*, *CodedCharSetId*y *Encoding* en la estructura MQMD contenida *en* MQXQH. Para cada estructura de cabecera MQ posterior presente, los campos *Format*, *CodedCharSetId*y *Encoding* de la estructura describen los datos que siguen a esa estructura; esos datos son otra estructura de cabecera MQ o los datos de mensaje de aplicación.

    Si se especifica la opción MQGMO_CONVERT para un mensaje MQFMT_XMIT_Q_HEADER, los datos del mensaje de aplicación y algunas de las estructuras de cabecera de MQ se convierten, *pero los datos de la estructura MQXQH no son*. Al volver de la llamada MQGET, por lo tanto:

    - Los valores de los campos *Format*, *CodedCharSetId*y *Encoding* del parámetro *MsgDesc* describen los datos de la estructura MQXQH y *no* los datos del mensaje de aplicación; por lo tanto,

los valores *no* son los mismos que los especificados por la aplicación que ha emitido la llamada MQGET.

El efecto de esto es que una aplicación que obtiene repetidamente mensajes de una cola de transmisión con la opción MQGMO_CONVERT especificada debe restablecer los campos *CodedCharSetId* y *Encoding* en el parámetro *MsgDesc* a los valores necesarios para los datos de mensaje de aplicación, antes de cada llamada MQGET.

- Los valores de los campos *Format*, *CodedCharSetId*y *Encoding* de la última estructura de cabecera de MQ presente describen los datos del mensaje de aplicación. Si no hay otras estructuras de cabecera MQ presentes, los datos del mensaje de aplicación se describen mediante estos campos en la estructura MQMD dentro de la estructura MQXQH. Si la conversión es satisfactoria, los valores serán los mismos que los especificados en el parámetro *MsgDesc* por la aplicación que ha emitido la llamada MQGET.

Si el mensaje es un mensaje de lista de distribución, la estructura MQXQH va seguida de una estructura MQDH (más sus matrices de registros MQOR y MQPMR), que a su vez puede ir seguida de cero o más estructuras de cabecera MQ adicionales y de cero o más bytes de datos de mensaje de aplicación. Al igual que la estructura MQXQH, la estructura MQDH debe estar en el juego de caracteres y la codificación del gestor de colas, y no se convierte en la llamada MQGET, incluso si se especifica la opción MQGMO_CONVERT.

El proceso de las estructuras MQXQH y MQDH descritas anteriormente está pensado principalmente para que las utilicen los agentes de canal de mensajes cuando obtienen mensajes de las colas de transmisión.

Conversión de mensajes de informe

En general, un mensaje de informe puede contener cantidades variables de datos de mensaje de aplicación, según las opciones de informe especificadas por el remitente del mensaje original. Sin embargo, un informe de actividad puede contener datos pero sin que la opción de informe mencione * _WITH_DATA en la constante.

En concreto, un mensaje de informe puede contener:

1. No hay datos de mensaje de aplicación
2. Algunos de los datos de mensaje de aplicación del mensaje original

   Esto ocurre cuando el remitente del mensaje original especifica MQRO_ * _WITH_DATA y el mensaje tiene más de 100 bytes.
3. Todos los datos de mensaje de aplicación del mensaje original

   Esto ocurre cuando el remitente del mensaje original especifica MQRO_ * _WITH_FULL_DATA, o especifica MQRO_ * _WITH_DATA y el mensaje tiene 100 bytes o menos.

Cuando el gestor de colas o el agente de canal de mensajes genera un mensaje de informe, copia el nombre de formato del mensaje original en el campo *Format* de la información de control del mensaje de informe. Por lo tanto, el nombre de formato del mensaje de informe puede implicar una longitud de datos diferente de la longitud realmente presente en el mensaje de informe (casos 1 y 2 anteriores).

Si se especifica la opción MQGMO_CONVERT cuando se recupera el mensaje de informe:

- Para el caso 1 anterior, no se invoca la salida de conversión de datos (porque el mensaje de informe no tiene datos).
- Para el caso 3 anterior, el nombre de formato implica correctamente la longitud de los datos del mensaje.
- Pero para el caso 2 anterior, se invoca la salida de conversión de datos para convertir un mensaje que es *más corto* que la longitud implícita por el nombre de formato.

  Además, el código de razón pasado a la salida suele ser MQRC_NONE (es decir, el código de razón no indica que el mensaje se haya truncado). Esto sucede porque los datos del mensaje han sido truncados por el *emisor* del mensaje de informe y no por el gestor de colas del receptor en respuesta a la llamada MQGET.

Debido a estas posibilidades, la salida de conversión de datos *no* debe utilizar el nombre de formato para deducir la longitud de los datos que se le pasan; en su lugar, la salida debe comprobar la longitud de los datos proporcionados y estar preparada para convertir *menos* datos que la longitud implícita en el nombre de formato. Si los datos se pueden convertir correctamente, la salida debe devolver el código de terminación MQCC_OK y el código de razón MQRC_NONE. La longitud de los datos de mensaje que se van a convertir se pasa a la salida como parámetro *InBufferLength* .

Interfaz de programación sensible al producto

MQDXP-Parámetro de salida de conversión de datos

La estructura MQDXP es un parámetro que el gestor de colas pasa a la salida de conversión de datos cuando se invoca la salida para convertir los datos del mensaje como parte del proceso de la llamada MQGET. Consulte la descripción de la llamada MQ_DATA_CONV_EXIT para obtener detalles de la salida de conversión de datos.

Los datos de caracteres en MQDXP están en el juego de caracteres del gestor de colas local; esto lo proporciona el atributo de gestor de colas *CodedCharSetId* . Los datos numéricos en MQDXP están en la codificación de máquina nativa; esto lo proporciona MQENC_NATIVE.

La salida sólo puede cambiar los campos *DataLength*, *CompCode*, *Reason*y *ExitResponse* en MQDXP; los cambios en otros campos se ignoran. Sin embargo, el campo *DataLength no se puede* cambiar si el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico.

Cuando el control vuelve al gestor de colas desde la salida, el gestor de colas comprueba los valores devueltos en MQDXP. Si los valores devueltos no son válidos, el gestor de colas continúa el proceso como si la salida hubiera devuelto MQXDR_CONVERSION_FAILED en *ExitResponse*; sin embargo, el gestor de colas ignora los valores de los campos *CompCode* y *Reason* devueltos por la salida en este caso, y utiliza en su lugar los valores que esos campos tenían en *entrada* a la salida. Los siguientes valores en MQDXP hacen que se produzca este proceso:

- El campo *ExitResponse* no es MQXDR_OK y no es MQXDR_CONVERSION_FAILED
- El campo *CompCode* no es MQCC_OK y no es MQCC_WARNING
- *DataLength* campo menor que cero, o *DataLength* campo cambiado cuando el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico.

La tabla siguiente resume los campos de la estructura.

Tabla 579. Campos en MQDXP

Campo	Descripción	Tema
StrucId	Identificador de la estructura	StrucId
Version	Número de versión de la estructura	Versión
AppOptions	Opciones de aplicación	AppOptions
Encoding	Codificación numérica necesaria para la aplicación	Codificación
CodedCharSetId	Juego de caracteres necesario para la aplicación	CodedCharSetId
DataLength	Longitud en bytes de datos de mensaje	DataLength
CompCode	Código de terminación	CompCode
Reason	Código de razón que califica *CompCode*	Razón

Tabla 579. Campos en MQDXP (continuación)		
Campo	**Descripción**	**Tema**
ExitResponse	Respuesta de la salida	ExitResponse
Hconn	Descriptor de contexto de conexión	Hconn
pEntryPoints	Dirección de la estructura MQIEP	PuntospEntry

Campos

La estructura MQDXP contiene los campos siguientes; los campos se describen en orden alfabético.

AppOptions
Tipo: MQLONG

Se trata de una copia del campo *Options* de la estructura MQGMO especificada por la aplicación que emite la llamada MQGET. Es posible que la salida tenga que examinarlos para determinar si se ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG.

Es un campo de entrada para la salida.

CodedCharSetId
Tipo: MQLONG

Este es el identificador de juego de caracteres codificado del juego de caracteres que necesita la aplicación que emite la llamada MQGET; consulte el campo *CodedCharSetId* en la estructura MQMD para obtener más detalles. Si la aplicación especifica el valor especial MQCCSI_Q_MGR en la llamada MQGET, el gestor de colas lo cambia por el identificador de juego de caracteres real del juego de caracteres utilizado por el gestor de colas, antes de invocar la salida.

Si la conversión es satisfactoria, la salida debe copiarla en el campo *CodedCharSetId* del descriptor de mensaje.

Es un campo de entrada para la salida.

CompCode
Tipo: MQLONG

Cuando se invoca la salida, contiene el código de terminación que se devuelve a la aplicación que ha emitido la llamada MQGET, si la salida no hace nada. Siempre es MQCC_WARNING, porque el mensaje se ha truncado o el mensaje requiere conversión y esto todavía no se ha realizado.

En la salida de la salida, este campo contiene el código de terminación que se devolverá a la aplicación en el parámetro *CompCode* de la llamada MQGET; sólo son válidos MQCC_OK y MQCC_WARNING. Consulte la descripción del campo *Reason* para obtener sugerencias sobre cómo la salida puede establecer este campo en la salida.

Es un campo de entrada/salida para la salida.

DataLength
Tipo: MQLONG

Cuando se invoca la salida, este campo contiene la longitud original de los datos del mensaje de aplicación. Si el mensaje se ha truncado para que quepa en el almacenamiento intermedio proporcionado por la aplicación, el tamaño del mensaje proporcionado a la salida es *menor* que el valor de *DataLength*. El tamaño del mensaje proporcionado a la salida siempre lo proporciona el parámetro *InBufferLength* de la salida, independientemente de cualquier truncamiento que se haya producido.

El truncamiento se indica mediante el campo *Reason* que tiene el valor MQRC_TRUNCATED_MSG_ACCEPTED en la entrada de la salida.

La mayoría de las conversiones no necesitan cambiar esta longitud, pero una salida puede hacerlo si es necesario; el valor establecido por la salida se devuelve a la aplicación en el parámetro *DataLength* de la llamada MQGET. Sin embargo, esta longitud *no se puede* cambiar si el mensaje que se está convirtiendo es un segmento que sólo contiene parte de un mensaje lógico. Esto se debe a que cambiar la longitud haría que los desplazamientos de segmentos posteriores en el mensaje lógico fueran incorrectos.

Tenga en cuenta que, si la salida desea cambiar la longitud de los datos, tenga en cuenta que el gestor de colas ya ha decidido si los datos del mensaje se ajustan al almacenamiento intermedio de la aplicación, basándose en la longitud de los datos *sin convertir* . Esta decisión determina si el mensaje se elimina de la cola (o se mueve el cursor para examinar, para una solicitud de examinar) y no se ve afectado por ningún cambio en la longitud de los datos causado por la conversión. Por este motivo, se recomienda que las salidas de conversión no provoquen un cambio en la longitud de los datos del mensaje de aplicación.

Si la conversión de caracteres implica un cambio de longitud, una serie se puede convertir en otra serie con la misma longitud en bytes, truncando los espacios en blanco finales o rellenando con espacios en blanco según sea necesario.

La salida no se invoca si el mensaje no contiene datos de mensaje de aplicación; por lo tanto, *DataLength* siempre es mayor que cero.

Es un campo de entrada/salida para la salida.

Encoding
Tipo: MQLONG

Codificación numérica necesaria para la aplicación.

Esta es la codificación numérica que necesita la aplicación que emite la llamada MQGET; consulte el campo *Encoding* en la estructura MQMD para obtener más detalles.

Si la conversión es satisfactoria, la salida la copia en el campo *Encoding* del descriptor de mensaje.

Es un campo de entrada para la salida.

ExitOptions
Tipo: MQLONG

Se trata de un campo reservado; su valor es `0`.

ExitResponse
Tipo: MQLONG

Respuesta de la salida.Esto lo establece la salida para indicar el éxito o no de la conversión. Debe ser uno de los siguientes:

MQXDR_Correcto
La conversión ha sido satisfactoria.

Si la salida especifica este valor, el gestor de colas devuelve lo siguiente a la aplicación que ha emitido la llamada MQGET:

- El valor del campo *CompCode* en la salida de la salida
- El valor del campo *Reason* en la salida de la salida
- El valor del campo *DataLength* en la salida de la salida
- El contenido del almacenamiento intermedio de salida de la salida *OutBuffer*. El número de bytes devueltos es el menor del parámetro *OutBufferLength* de la salida y el valor del campo *DataLength* en la salida de la salida.

  Si los campos *Encoding* y *CodedCharSetId* del parámetro de descriptor de mensaje de la salida *ambos* no se modifican, el gestor de colas devuelve:
- El valor de los campos *Encoding* y *CodedCharSetId* en la estructura MQDXP en *entrada* a la salida.

Si se ha cambiado uno o ambos campos *Encoding* y *CodedCharSetId* en el parámetro de descriptor de mensaje de la salida, el gestor de colas devuelve:

- El valor de los campos *Encoding* y *CodedCharSetId* en el parámetro de descriptor de mensaje de la salida en la salida de la salida

MQXDR_CONVERSION_FAILED
La conversión no ha sido satisfactoria.

Si la salida especifica este valor, el gestor de colas devuelve lo siguiente a la aplicación que ha emitido la llamada MQGET:

- El valor del campo *CompCode* en la salida de la salida
- El valor del campo *Reason* en la salida de la salida
- El valor del campo *DataLength* en *entrada* a la salida
- El contenido del almacenamiento intermedio de entrada de la salida *InBuffer*. El número de bytes devueltos lo proporciona el parámetro *InBufferLength*

Si la salida ha modificado *InBuffer*, los resultados no están definidos.

ExitResponse es un campo de salida de la salida.

Hconn
Tipo: MQHCONN

Se trata de un descriptor de conexión que se puede utilizar en la llamada MQXCNVC. Este descriptor de contexto no es necesariamente el mismo que el descriptor de contexto especificado por la aplicación que ha emitido la llamada MQGET.

pEntryPoints
Tipo: PMQIEP

La dirección de una estructura MQIEP a través de la cual se pueden realizar llamadas MQI y DCI.

Reason
Tipo: MQLONG

Código de razón que califica *CompCode*.

Cuando se invoca la salida, contiene el código de razón que se devuelve a la aplicación que ha emitido la llamada MQGET, si la salida elige no hacer nada. Entre los valores posibles se encuentran MQRC_TRUNCATED_MSG_ACCEPTED, que indica que el mensaje se ha truncado en orden de ajuste en el almacenamiento intermedio proporcionado por la aplicación, y MQRC_NOT_CONVERT, que indica que el mensaje requiere conversión pero que todavía no se ha realizado.

En la salida de la salida, este campo contiene la razón que debe devolverse a la aplicación en el parámetro *Reason* de la llamada MQGET; se recomienda lo siguiente:

- Si *Reason* tenía el valor MQRC_TRUNCATED_MSG_ACCEPTED en la entrada a la salida, los campos *Reason* y *CompCode* no deben alterarse, independientemente de si la conversión es satisfactoria o falla.

  (Si el campo *CompCode* no es MQCC_OK, la aplicación que recupera el mensaje puede identificar un error de conversión comparando los valores *Encoding* y *CodedCharSetId* devueltos en el descriptor de mensaje con los valores solicitados; por el contrario, la aplicación no puede distinguir un mensaje truncado de un mensaje que encajaba en el almacenamiento intermedio. Por este motivo, MQRC_TRUNCATED_MSG_ACCEPTED debe devolverse con preferencia a cualquiera de las razones que indican un error de conversión.)
- Si *Reason* tenía cualquier otro valor en la entrada para la salida:
  - Si la conversión se realiza correctamente, *CompCode* debe establecerse en MQCC_OK y *Reason* debe establecerse en MQRC_NONE.
  - Si la conversión falla, o el mensaje se expande y se tiene que truncar para que quepa en el almacenamiento intermedio, *CompCode* debe establecerse en MQCC_WARNING (o dejarse sin

modificar), y *Reason* establecerse en uno de los valores listados, para indicar la naturaleza de la anomalía.

Tenga en cuenta que si el mensaje después de la conversión es demasiado grande para el almacenamiento intermedio, sólo se debe truncar si la aplicación que ha emitido la llamada MQGET ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG:

- Si ha especificado esta opción, se devuelve la razón MQRC_TRUNCATED_MSG_ACCEPTED.
- Si no ha especificado esa opción, el mensaje se devuelve sin convertir, con el código de razón MQRC_CONVERTED_MSG_TOO_BIG.

Los códigos de razón listados se recomiendan para que los utilice la salida para indicar la razón por la que ha fallado la conversión, pero la salida puede devolver otros valores del conjunto de códigos MQRC_ * si se considera apropiado. Además, el rango de valores de MQRC_APPL_FIRST a MQRC_APPL_LAST se asignan para que los utilice la salida para indicar las condiciones en las que la salida desea comunicarse con la aplicación que emite la llamada MQGET.

Nota: Si el mensaje no se puede convertir correctamente, la salida *debe* devolver MQXDR_CONVERSION_FAILED en el campo *ExitResponse* , para que el gestor de colas devuelva el mensaje no convertido. Esto es cierto independientemente del código de razón devuelto en el campo *Reason* .

MQRC_APPL_PRIMERO
(900, X'384 ') Valor más bajo para el código de razón definido por la aplicación.

MQRC_APPL_LAST
(999, X'3E7') Valor más alto para el código de razón definido por la aplicación.

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848') Los datos convertidos son demasiado grandes para el almacenamiento intermedio.

MQRC_NOT_CONVERTED
(2119, X'847') Los datos del mensaje no se han convertido.

MQRC_SOURCE_CCSID_ERROR
(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

MQRC_SOURCE_DECIMAL_ENC_ERROR
(2113, X'841') Codificación de decimal empaquetado en el mensaje no reconocida.

MQRC_SOURCE_FLOAT_ENC_ERROR
(2114, X'842') Codificación de coma flotante en el mensaje no reconocida.

MQRC_SOURCE_INTEGER_ENC_ERROR
(2112, X'840') Codificación de entero de origen no reconocida.

MQRC_TARGET_CCSID_ERROR
(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

MQRC_TARGET_DECIMAL_ENC_ERROR
(2117, X'845') Codificación de decimal empaquetado especificada por receptor no reconocida.

MQRC_TARGET_FLOAT_ENC_ERROR
(2118, X'846') Codificación de coma flotante especificada por receptor no reconocida.

MQRC_TARGET_INTEGER_ENC_ERROR
(2116, X'844') Codificación de entero de destino no reconocida.

MQRC_TRUNCATED_MSG_ACCEPTED
(2079, X'81F') Se ha devuelto un mensaje truncado (el proceso se ha completado).

Es un campo de entrada/salida para la salida.

StrucId
Tipo: MQCHAR4

Identificador de estructura.El valor debe ser:

MQDXP_STRUC_ID
Identificador de la estructura de parámetros de salida de conversión de datos.

Para el lenguaje de programación C, también se define la constante MQDXP_STRUC_ID_ARRAY; tiene el mismo valor que MQDXP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para la salida.

Version
Tipo: MQLONG

Número de versión de la estructura.El valor debe ser:

MQDXP_VERSION_1
Número de versión para la estructura de parámetros de salida de conversión de datos.

La constante siguiente especifica el número de versión de la versión actual:

MQDXP_CURRENT_VERSION
Versión actual de la estructura de parámetros de salida de conversión de datos.

Nota: Cuando se introduce una nueva versión de esta estructura, el diseño de la parte existente no cambia. Por lo tanto, la salida debe comprobar que el campo *Version* es igual o mayor que la versión más baja que contiene los campos que la salida necesita utilizar.

Es un campo de entrada para la salida.

Declaración 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 */
};
```

Declaración COBOL (soloIBM 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.
```

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-Convertir caracteres

La llamada MQXCNVC convierte los caracteres de un juego de caracteres a otro utilizando el lenguaje de programación C.

Esta llamada forma parte de la interfaz de conversión de datos (DCI) de WebSphere MQ , que es una de las interfaces de infraestructura de WebSphere MQ .

Nota: La llamada se puede utilizar desde los entornos de salida de aplicación y de conversión de datos.

Sintaxis

MQXCNVC (*Hconn*, *Options*, *SourceCCSID*, *SourceLength*, *SourceBuffer*, *TargetCCSID*, *TargetLength*, *TargetBuffer*, *DataLength*, *CompCode*, *Reason*)

Parámetros

Hconn
Tipo: MQHCONN - entrada

Este manejador representa la conexión con el gestor de colas.

En una salida de conversión de datos, *Hconn* es normalmente el descriptor de contexto que se pasa a la salida de conversión de datos en el campo *Hconn* de la estructura MQDXP; este descriptor de contexto no es necesariamente el mismo que el descriptor de contexto especificado por la aplicación que ha emitido la llamada MQGET.

En IBM i, se puede especificar el siguiente valor especial para *Hconn*:

MQHC_DEF_HCONN
Manejador de conexión predeterminado.

Si ejecuta una aplicación CICS TS 3.2 o superior, asegúrese de que el programa de salida de conversión de caracteres, que invoca la llamada MQXCNVC, esté definido como OPENAPI. Esta definición evita el error 2018 MQRC_HCONN_ERROR provocado por una conexión incorrecta y permite que se complete la MQGET.

opciones
Tipo: MQLONG - entrada

Opciones que controlan la acción de MQXCNVC.

Se pueden especificar cero o más de las opciones descritas. Si se necesita más de uno, los valores pueden ser:

- Añadido (no añadir la misma constante más de una vez), o

- Se combina utilizando la operación OR a nivel de bit (si el lenguaje de programación da soporte a operaciones de bit)

Opción de conversión predeterminada: la siguiente opción controla el uso de la conversión de caracteres predeterminada:

MQDCC_DEFAULT_CONVERSION
Conversión predeterminada.

Esta opción especifica que se puede utilizar la conversión de caracteres por omisión si uno o ambos de los juegos de caracteres especificados en la llamada no están soportados. Esto permite al gestor de colas utilizar un juego de caracteres predeterminado especificado por la instalación que se aproxima al juego de caracteres especificado, al convertir la serie.

Nota: El resultado de utilizar un juego de caracteres aproximado para convertir la serie es que algunos caracteres se pueden convertir incorrectamente. Esto se puede evitar utilizando en la serie sólo caracteres que son comunes tanto al juego de caracteres especificado como al juego de caracteres predeterminado.

Los juegos de caracteres predeterminados se definen mediante una opción de configuración cuando se instala o se reinicia el gestor de colas.

Si no se especifica MQDCC_DEFAULT_CONVERSION, el gestor de colas sólo utiliza los juegos de caracteres especificados para convertir la serie, y la llamada falla si uno o ambos juegos de caracteres no están soportados.

Esta opción está soportada en los entornos siguientes: AIX, HP-UX, IBM i, Solaris, Linux, Windows.

Opción de relleno: la opción siguiente permite al gestor de colas rellenar la serie convertida con espacios en blanco o descartar caracteres finales insignificantes, para que la serie convertida se ajuste al almacenamiento intermedio de destino:

MQDCC_FILL_TARGET_BUFFER
Rellene el almacenamiento intermedio de destino.

Esta opción solicita que la conversión tenga lugar de forma que el almacenamiento intermedio de destino se llene completamente:

- Si la serie se contrae cuando se convierte, se añaden espacios en blanco finales para rellenar el almacenamiento intermedio de destino.
- Si la serie se expande cuando se convierte, los caracteres finales que no son significativos se descartan para que la serie convertida se ajuste al almacenamiento intermedio de destino. Si esto se puede realizar correctamente, la llamada se completa con MQCC_OK y el código de razón MQRC_NONE.

  Si hay muy pocos caracteres finales insignificantes, la mayor parte de la serie que cabe se coloca en el almacenamiento intermedio de destino y la llamada se completa con MQCC_WARNING y el código de razón MQRC_CONVERTED_MSG_TOO_BIG.

  Los caracteres insignificantes son:

  – Blancos de cola
  – Caracteres que siguen al primer carácter nulo de la serie (pero excluyendo el propio primer carácter nulo)
- Si la serie *TargetCCSID*y *TargetLength* son tales que el almacenamiento intermedio de destino no se puede establecer completamente con caracteres válidos, la llamada falla con MQCC_FAILED y el código de razón MQRC_TARGET_LENGTH_ERROR. Esto puede ocurrir cuando *TargetCCSID* es un juego de caracteres DBCS puro (como UCS-2), pero *TargetLength* especifica una longitud que es un número impar de bytes.
- *TargetLength* puede ser menor o mayor que *SourceLength*. Al volver de MQXCNVC, *DataLength* tiene el mismo valor que *TargetLength*.

Si no se especifica esta opción:

- La serie puede contraerse o expandirse dentro del almacenamiento intermedio de destino según sea necesario. Los caracteres finales insignificantes no se añaden ni descartan.

  Si la serie convertida se ajusta al almacenamiento intermedio de destino, la llamada se completa con MQCC_OK y el código de razón MQRC_NONE.

  Si la serie convertida es demasiado grande para el almacenamiento intermedio de destino, la mayor parte de la serie que cabe se coloca en el almacenamiento intermedio de destino y la llamada se completa con MQCC_WARNING y el código de razón MQRC_CONVERTED_MSG_TOO_BIG. Tenga en cuenta que se pueden devolver menos de *TargetLength* bytes en este caso.
- *TargetLength* puede ser menor o mayor que *SourceLength*. Al volver de MQXCNVC, *DataLength* es menor o igual que *TargetLength*.

Esta opción está soportada en los entornos siguientes: AIX, HP-UX, IBM i, Solaris, Linux, Windows.

Opciones de codificación: Las opciones descritas se pueden utilizar para especificar las codificaciones de enteros de las series de origen y de destino. La codificación relevante se utiliza *sólo* cuando el identificador de juego de caracteres correspondiente indica que la representación del juego de caracteres en el almacenamiento principal depende de la codificación utilizada para los enteros binarios. Esto sólo afecta a determinados juegos de caracteres de varios bytes (por ejemplo, juegos de caracteres UCS-2).

La codificación se ignora si el juego de caracteres es un juego de caracteres de un solo byte (SBCS), o un juego de caracteres de varios bytes con representación en el almacenamiento principal que no depende de la codificación de enteros.

Sólo se debe especificar uno de los valores MQDCC_SOURCE_ *, combinado con uno de los valores MQDCC_TARGET_ *:

MQDCC_SOURCE_ENC_NATIVE
La codificación de origen es el valor predeterminado para el entorno y el lenguaje de programación.

MQDCC_SOURCE_ENC_NORMAL
La codificación de origen es normal.

MQDCC_SOURCE_ENC_REVERSE
La codificación de origen se invierte.

MQDCC_SOURCE_ENC_UNDEFINED
La codificación de origen no está definida.

MQDCC_TARGET_ENC_NATIVE
La codificación de destino es el valor predeterminado para el entorno y el lenguaje de programación.

MQDCC_TARGET_ENC_NORMAL
La codificación de destino es normal.

MQDCC_TARGET_ENC_REVERSE
La codificación de destino se invierte.

MQDCC_TARGET_ENC_UNDEFINED
La codificación de destino no está definida.

Los valores de codificación definidos anteriormente se pueden añadir directamente al campo *Options* . Sin embargo, si la codificación de origen o destino se obtiene del campo *Encoding* en el MQMD u otra estructura, se debe realizar el proceso siguiente:

1. La codificación de enteros debe extraerse del campo *Encoding* eliminando las codificaciones decimal flotante y empaquetado; consulte “Análisis de codificaciones” en la página 883 para obtener detalles sobre cómo hacerlo.
2. La codificación de enteros resultante del paso 1 debe multiplicarse por el factor adecuado antes de añadirse al campo *Options* . Estos factores son:

- MQDCC_SOURCE_ENC_FACTOR para la codificación de origen
- MQDCC_TARGET_ENC_FACTOR para la codificación de destino

El código de ejemplo siguiente ilustra cómo se puede codificar en el lenguaje de programación C:

```
Options = (MsgDesc.Encoding & MQENC_INTEGER_MASK)
           * MQDCC_SOURCE_ENC_FACTOR
        + (DataConvExitParms.Encoding & MQENC_INTEGER_MASK)
           * MQDCC_TARGET_ENC_FACTOR;
```

Si no se especifica, las opciones de codificación toman como valor predeterminado sin definir (MQDCC_ * _ENC_UNDEFINED). En la mayoría de los casos, esto no afecta a la finalización satisfactoria de la llamada MQXCNVC. Sin embargo, si el juego de caracteres correspondiente es un juego de caracteres multibyte con representación que depende de la codificación (por ejemplo, un juego de caracteres UCS-2), la llamada falla con el código de razón MQRC_SOURCE_INTEGER_ENC_ERROR o MQRC_TARGET_INTEGER_ENC_ERROR según corresponda.

Las opciones de codificación están soportadas en los entornos siguientes: AIX, HP-UX, z/OS, IBM i, Solaris, Linux, Windows.

Opción predeterminada: si no se especifica ninguna de las opciones descritas anteriormente, se puede utilizar la opción siguiente:

MQDCC_NONE
No se ha especificado ninguna opción.

MQDCC_NONE está definido para ayudar a la documentación del programa. No se pretende que esta opción se utilice con ninguna otra, pero como su valor es cero, no se puede detectar dicho uso.

SourceCCSID
Tipo: MQLONG - entrada

Es el identificador de juego de caracteres codificado de la serie de entrada en *SourceBuffer*.

SourceLength
Tipo: MQLONG - entrada

Es la longitud en bytes de la serie de entrada en *SourceBuffer*; debe ser cero o mayor.

SourceBuffer
Tipo: MQCHARxSourceLongitud-entrada

Es el almacenamiento intermedio que contiene la serie que se va a convertir de un juego de caracteres a otro.

TargetCCSID
Tipo: MQLONG - entrada

Es el identificador de juego de caracteres codificado del juego de caracteres al que se va a convertir *SourceBuffer* .

TargetLength
Tipo: MQLONG - entrada

Es la longitud en bytes del almacenamiento intermedio de salida *TargetBuffer*; debe ser cero o mayor. Puede ser menor o mayor que *SourceLength*.

TargetBuffer
Tipo: MQCHARxTargetLongitud-salida

Esta es la serie después de que se haya convertido al juego de caracteres definido por *TargetCCSID*. La serie convertida puede ser más corta o más larga que la serie no convertida. El parámetro *DataLength* indica el número de bytes válidos devueltos.

DataLength
Tipo: MQLONG - salida

Es la longitud de la serie devuelta en el almacenamiento intermedio de salida *TargetBuffer*. La serie convertida puede ser más corta o más larga que la serie no convertida.

CompCode
Tipo: MQLONG - salida

Es uno de los siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el valor de *CompCode* es MQCC_WARNING:

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848') Los datos convertidos son demasiado grandes para el almacenamiento intermedio.

Si el valor de *CompCode* es MQCC_FAILED:

MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') El parámetro longitud de datos no es válido.

MQRC_DBCS_ERROR
(2150, X'866') La serie DBCS no es válida.

MQRC_HCONN_ERROR
(2018, X'7E2') El manejador de conexión no es válido.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_RESOURCE_PROBLEM
(2102, X'836') No hay disponibles suficientes recursos del sistema.

MQRC_SOURCE_BUFFER_ERROR
(2145, X'861') El parámetro de almacenamiento intermedio de origen no es válido.

MQRC_SOURCE_CCSID_ERROR
(2111, X'83F') El identificador de juego de caracteres codificado origen no es válido.

MQRC_SOURCE_INTEGER_ENC_ERROR
(2112, X'840') Codificación de entero de origen no reconocida.

MQRC_SOURCE_LENGTH_ERROR
(2143, X'85F') Parámetro de longitud de origen no válido.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') No hay suficiente almacenamiento disponible.

MQRC_TARGET_BUFFER_ERROR
(2146, X'862') Parámetro de almacenamiento intermedio de destino no válido.

MQRC_TARGET_CCSID_ERROR
(2115, X'843') El identificador de juego de caracteres codificado de destino no es válido.

MQRC_TARGET_INTEGER_ENC_ERROR
(2116, X'844') Codificación de entero de destino no reconocida.

MQRC_TARGET_LENGTH_ERROR
(2144, X'860 ') El parámetro de longitud de destino no es válido.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Se ha producido un error inesperado.

Para obtener información detallada sobre estos códigos, consulte Códigos de razón.

Invocación en C

```
MQXCNVC (Hconn, Options, SourceCCSID, SourceLength, SourceBuffer,
        TargetCCSID, TargetLength, TargetBuffer, &DataLength,
        &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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 */
```

Declaración COBOL (soloIBM i)

```
     CALL 'MQXCNVC' USING HCONN, OPTIONS, SOURCECCSID, SOURCELENGTH,
                          SOURCEBUFFER, TARGETCCSID, TARGETLENGTH,
                          TARGETBUFFER, DATALENGTH, COMPCODE, REASON.
```

Declare los parámetros como se indica a continuación:

```
**   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.
```

S/390 declaración de ensamblador

```
        CALL MQXCNVC,(HCONN,OPTIONS,SOURCECCSID,SOURCELENGTH,         X
              SOURCEBUFFER,TARGETCCSID,TARGETLENGTH,TARGETBUFFER,     X
              DATALENGTH,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
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-Salida de conversión de datos

La llamada MQ_DATA_CONV_EXIT describe los parámetros que se pasan a la salida de conversión de datos.

El gestor de colas no proporciona ningún punto de entrada denominado MQ_DATA_CONV_EXIT (consulte la nota de uso 11).

Esta definición forma parte de la interfaz de conversión de datos (DCI) de WebSphere MQ , que es una de las interfaces de infraestructura de WebSphere MQ .

Sintaxis

MQ_DATA_CONV_EXIT (*DataConvExitParms*, *MsgDesc*, *InBufferLength*, *InBuffer*, *OutBufferLength*, *OutBuffer*)

Parámetros

DataConvExitParms
Tipo: MQDXP-entrada/salida

Esta estructura contiene información relacionada con la invocación de la salida. La salida establece información en esta estructura para indicar el resultado de la conversión. Consulte "MQDXP-Parámetro de salida de conversión de datos" en la página 895 para obtener detalles de los campos de esta estructura.

MsgDesc
Tipo: MQMD - entrada/salida

En la entrada a la salida, este es el descriptor de mensaje asociado con los datos de mensaje pasados a la salida en el parámetro *InBuffer* .

Nota: El parámetro *MsgDesc* pasado a la salida es siempre la versión más reciente de MQMD soportada por el gestor de colas que invoca la salida. Si la salida está pensada para ser portable entre distintos entornos, la salida comprobará el campo *Version* en *MsgDesc* para verificar que los campos a los que la salida necesita acceder están presentes en la estructura.

En los entornos siguientes, a la salida se le pasa un MQMD version-2 : AIX, HP-UX, IBM i, Solaris, Linux, Windows. En todos los demás entornos que dan soporte a la salida de conversión de datos, a la salida se le pasa un MQMD de version-1 .

En la salida, la salida cambiará los campos *Encoding* y *CodedCharSetId* por los valores solicitados por la aplicación, si la conversión ha sido satisfactoria; estos cambios se reflejan de nuevo en la aplicación. Cualquier otro cambio que realice la salida en la estructura se ignorará; no se reflejarán de nuevo en la aplicación.

Si la salida devuelve MQXDR_OK en el campo *ExitResponse* de la estructura MQDXP, pero no cambia los campos *Encoding* o *CodedCharSetId* del descriptor de mensaje, el gestor de colas devuelve para esos campos los valores que los campos correspondientes de la estructura MQDXP tenían en la entrada de la salida.

InBufferLongitud
Tipo: MQLONG - entrada

Longitud en bytes de *InBuffer*.

Es la longitud del almacenamiento intermedio de entrada *InBuffer*y especifica el número de bytes que debe procesar la salida. *InBufferLength* es la longitud menor de los datos del mensaje antes de la conversión y la longitud del almacenamiento intermedio proporcionado por la aplicación en la llamada MQGET.

El valor es siempre mayor que cero.

InBuffer
Tipo: MQBYTExInBufferLength -entrada

Almacenamiento intermedio que contiene el mensaje no convertido.

Contiene los datos del mensaje antes de la conversión. Si la salida no puede convertir los datos, el gestor de colas devuelve el contenido de este almacenamiento intermedio a la aplicación una vez completada la salida.

Nota: La salida no debe modificar *InBuffer*; si se modifica este parámetro, los resultados no están definidos.

En el lenguaje de programación C, este parámetro se define como un puntero a vacío.

OutBufferLongitud
Tipo: MQLONG - entrada

Longitud en bytes de *OutBuffer*.

Es la longitud del almacenamiento intermedio de salida *OutBuffer*y es la misma que la longitud del almacenamiento intermedio proporcionado por la aplicación en la llamada MQGET.

El valor es siempre mayor que cero.

OutBuffer
Tipo: MQBYTExOutBufferLength -salida

Almacenamiento intermedio que contiene el mensaje convertido.

En la salida de la salida, si la conversión ha sido satisfactoria (como indica el valor MQXDR_OK en el campo *ExitResponse* del parámetro *DataConvExitParms*), *OutBuffer* contiene los datos de mensaje que se van a entregar a la aplicación, en la representación solicitada. Si la conversión no ha sido satisfactoria, se ignoran los cambios que la salida ha realizado en este almacenamiento intermedio.

En el lenguaje de programación C, este parámetro se define como un puntero a vacío.

Notas de uso

1. Una salida de conversión de datos es una salida escrita por el usuario que recibe el control durante el proceso de una llamada MQGET. La función realizada por la salida de conversión de datos la define el proveedor de la salida; sin embargo, la salida debe ajustarse a las reglas descritas aquí y en la estructura de parámetros asociada MQDXP.

El entorno determina los lenguajes de programación que se pueden utilizar para una salida de conversión de datos.

2. La salida sólo se invoca si se cumplen *todas* las condiciones siguientes:
   - La opción MQGMO_CONVERT se especifica en la llamada MQGET
   - El campo *Format* del descriptor de mensaje no es MQFMT_NONE
   - El mensaje no está ya en la representación necesaria; es decir, uno o ambos *CodedCharSetId* y *Encoding* del mensaje son diferentes del valor especificado por la aplicación en el descriptor de mensaje proporcionado en la llamada MQGET
   - El gestor de colas todavía no ha realizado la conversión correctamente
   - La longitud del almacenamiento intermedio de la aplicación es mayor que cero
   - La longitud de los datos del mensaje es mayor que cero
   - El código de razón hasta ahora durante la operación MQGET es MQRC_NONE o MQRC_TRUNCATED_MSG_ACCEPTED
3. Cuando se está escribiendo una salida, considere la posibilidad de codificar la salida de una forma que le permita convertir los mensajes que se han truncado. Los mensajes truncados pueden aparecer de las siguientes maneras:
   - La aplicación receptora proporciona un almacenamiento intermedio que es menor que el mensaje, pero especifica la opción MQGMO_ACCEPT_TRUNCATED_MSG en la llamada MQGET.

     En este caso, el campo *Reason* del parámetro *DataConvExitParms* en la entrada de la salida tiene el valor MQRC_TRUNCATED_MSG_ACCEPTED.
   - El remitente del mensaje lo ha truncado antes de enviarlo. Esto puede suceder con los mensajes de informe, por ejemplo (consulte "Conversión de mensajes de informe" en la página 894 para obtener más detalles).

     En este caso, el campo *Reason* del parámetro *DataConvExitParms* en la entrada de la salida tiene el valor MQRC_NONE (si la aplicación receptora ha proporcionado un almacenamiento intermedio lo suficientemente grande para el mensaje).

   Por lo tanto, el valor del campo *Reason* en la entrada a la salida no siempre se puede utilizar para decidir si el mensaje se ha truncado.

   La característica distintiva de un mensaje truncado es que la longitud proporcionada a la salida en el parámetro *InBufferLength* es *menor que* la longitud implícita por el nombre de formato contenido en el campo *Format* del descriptor de mensaje. Por lo tanto, la salida debe comprobar el valor de *InBufferLength* antes de intentar convertir cualquiera de los datos; la salida *no* debe suponer que se ha proporcionado la cantidad completa de datos implícita en el nombre de formato.

   Si la salida *no* se ha grabado para convertir mensajes truncados, y *InBufferLength* es menor que el valor esperado, la salida devolverá MQXDR_CONVERSION_FAILED en el campo *ExitResponse* del parámetro *DataConvExitParms* , con los campos *CompCode* y *Reason* establecidos en MQCC_WARNING y MQRC_FORMAT_ERROR.

   Si la salida *ha* se ha grabado para convertir mensajes truncados, la salida convertirá la mayor cantidad de datos posible (consulte la siguiente nota de uso), teniendo cuidado de no intentar examinar o convertir datos más allá del final de *InBuffer*. Si la conversión se completa correctamente, la salida dejará el campo *Reason* en el parámetro *DataConvExitParms* sin modificar. Esto devuelve MQRC_TRUNCATED_MSG_ACCEPTED si el gestor de colas del receptor ha truncado el mensaje y MQRC_NONE si el emisor del mensaje ha truncado el mensaje.

   También es posible que un mensaje expanda *durante la conversión de* , hasta el punto en el que es mayor que *OutBuffer*. En este caso, la salida debe decidir si trunca el mensaje; el campo *AppOptions* del parámetro *DataConvExitParms* indica si la aplicación receptora ha especificado la opción MQGMO_ACCEPT_TRUNCATED_MSG.
4. Generalmente, todos los datos del mensaje proporcionados a la salida en *InBuffer* se convierten, o que ninguno de ellos lo es. Sin embargo, se produce una excepción a esto si el mensaje se trunca, ya sea antes de la conversión o durante la conversión; en este caso puede haber un elemento

incompleto al final del almacenamiento intermedio (por ejemplo: 1 byte de un carácter de doble byte o 3 bytes de un entero de 4 bytes). En esta situación, considere la posibilidad de omitir el elemento incompleto y establecer los bytes no utilizados en *OutBuffer* en nulos. Sin embargo, los elementos o caracteres completos dentro de una matriz o serie *deben* convertirse.

5. Cuando se necesita una salida por primera vez, el gestor de colas intenta cargar un objeto que tiene el mismo nombre que el formato (aparte de las extensiones). El objeto cargado debe contener la salida que procesa los mensajes con ese nombre de formato. Considere la posibilidad de hacer que el nombre de salida y el nombre del objeto que contiene la salida sean idénticos, aunque no todos los entornos lo requieran.
6. Se carga una nueva copia de la salida cuando una aplicación intenta recuperar el primer mensaje que utiliza ese *Format* desde que la aplicación se conectó al gestor de colas. Para aplicaciones CICS o IMS , esto significa cuando el subsistema CICS o IMS está conectado al gestor de colas. Una nueva copia también se puede cargar en otros momentos, si el gestor de colas ha descartado una copia cargada anteriormente. Por este motivo, una salida no debe intentar utilizar el almacenamiento estático para comunicar información de una invocación de la salida a la siguiente-la salida se puede descargar entre las dos invocaciones.
7. Si hay una salida proporcionada por el usuario con el mismo nombre que uno de los formatos incorporados soportados por el gestor de colas, la salida proporcionada por el usuario no sustituye la rutina de conversión incorporada. Las únicas circunstancias en las que se invoca una salida de este tipo son:
   - Si la rutina de conversión incorporada no puede manejar conversiones a o desde los *CodedCharSetId* o *Encoding* implicados, o
   - Si la rutina de conversión incorporada no ha podido convertir los datos (por ejemplo, porque hay un campo o carácter que no se puede convertir).
8. El ámbito de la salida depende del entorno. Los nombres de *Format* deben elegirse para minimizar el riesgo de conflictos con otros formatos. Considere la posibilidad de empezar con caracteres que identifiquen la aplicación que define el nombre de formato.
9. La salida de conversión de datos se ejecuta en un entorno como el del programa que ha emitido la llamada MQGET; el entorno incluye el espacio de direcciones y el perfil de usuario (si procede). El programa podría ser un agente de canal de mensajes que envía mensajes a un gestor de colas de destino que no soporta la conversión de mensajes. La salida no puede comprometer la integridad del gestor de colas, ya que no se ejecuta en el entorno del gestor de colas.
10. La única llamada MQI que puede utilizar la salida es MQXCNVC; el intento de utilizar otras llamadas MQI falla con el código de razón MQRC_CALL_IN_PROGRESS, u otros errores imprevisibles.
11. El gestor de colas no proporciona ningún punto de entrada denominado MQ_DATA_CONV_EXIT. Sin embargo, se proporciona un `typedef` para el nombre MQ_DATA_CONV_EXIT en el lenguaje de programación C, y esto se puede utilizar para declarar la salida escrita por el usuario, para asegurarse de que los parámetros son correctos. El nombre de la salida debe ser el mismo que el nombre de formato (el nombre contenido en el campo *Format* en MQMD), aunque esto no es necesario en todos los entornos.

    El ejemplo siguiente ilustra cómo se puede declarar la salida que procesa el formato MYFORMAT en el lenguaje de programación 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. En z/OS, si también está en vigor una salida cruzada de API, se llama después de la salida de conversión de datos.

Invocación en C

```
exitname (&DataConvExitParms, &MsgDesc, InBufferLength,
         InBuffer, OutBufferLength, OutBuffer);
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
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 */
```

Declaración COBOL (soloIBM i)

```
     CALL 'exitname' USING DATACONVEXITPARMS, MSGDESC, INBUFFERLENGTH,
                           INBUFFER, OUTBUFFERLENGTH, OUTBUFFER.
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
**   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).
```

System/390

```
         CALL EXITNAME,(DATACONVEXITPARMS,MSGDESC,INBUFFERLENGTH,      X
               INBUFFER,OUTBUFFERLENGTH,OUTBUFFER)
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
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
```

Propiedades especificadas como elementos MQRFH2

Las propiedades de descriptor que no son de mensaje se pueden especificar como elementos en las carpetas de cabecera MQRFH2 . Visión general de los elementos MQRFH2 que se especifican como propiedades.

Esto mantiene la compatibilidad con las versiones anteriores de los clientes WebSphere MQ JMS y XMS . En esta sección se describe cómo especificar propiedades en las cabeceras MQRFH2 .

Para utilizar elementos MQRFH2 como propiedades, especifique los elementos tal como se describe en Utilización de clases de WebSphere MQ para Java. Esta información complementa la información descrita en “MQRFH2 – Reglas y formato de la cabecera 2” en la página 504.

Correlación de tipos de datos de propiedad con tipos de datos MQRFH2

Este tema proporciona información sobre los tipos de propiedad de mensaje correlacionados con sus tipos de datos MQRFH2 correspondientes.

Tabla 580. Tipos de datos MQRFH2 soportados

Tipo de propiedad de mensaje	**Tipo de datos MQRFH2**
MQBYTE []	bin.hex
MQBOOL	boolean
MQINT8	i1
MQINT16	i2
MQINT32	i4
MQINT64	i8
MQFLOAT32	r4
MQFLOAT64	r8
MQCHAR []	serie

Se supone que cualquier elemento sin un tipo de datos es de tipo "serie".

Un tipo de datos MQRFH2 de `int`, que significa un entero de tamaño no especificado, se trata como si fuera un `i8`.

Un valor nulo se indica mediante el atributo de elemento `xsi:nil='true'`. No utilice el atributo `xsi:nil='false'` para valores no nulos.

Por ejemplo, la propiedad siguiente tiene un valor nulo:

```
<NullProperty xsi:nil='true'></NullProperty>
```

Una propiedad de serie de bytes o caracteres puede tener un valor vacío. Esto se representa mediante un elemento MQRFH2 con un valor de elemento de longitud cero.

Por ejemplo, la propiedad siguiente tiene un valor vacío:

```
<EmptyProperty></EmptyProperty>
```

Carpetas MQRFH2 soportadas

Visión general del uso de campos de descriptor de mensaje como propiedades.

Las carpetas <jms>, <mcd>, <mqext>y <usr> se describen en La cabecera MQRFH2 y JMS. La carpeta <usr> se utiliza para transportar las propiedades definidas por la aplicación JMS que están asociadas con un mensaje. Los grupos no están permitidos en la carpeta <usr> .

La cabecera MQRFH2 y JMS dan soporte a las carpetas adicionales siguientes:

- <mq>

  Esta carpeta se utiliza y se reserva para las propiedades definidas por MQque utiliza IBM WebSphere MQ.

- <mq_usr>

  Esta carpeta se puede utilizar para transportar cualquier propiedad definida por la aplicación que no esté expuesta como propiedades definidas por el usuario JMS, ya que es posible que las propiedades no cumplan los requisitos de una propiedad JMS. Esta carpeta puede contener grupos que la carpeta <usr> no puede.

- Cualquier carpeta marcada con el atributo content='properties' .

  Una carpeta de este tipo es equivalente a la carpeta <mq_usr> en el contenido.

- <mqps>

  Esta carpeta se utiliza para las propiedades de publicación/suscripción de IBM WebSphere MQ .

IBM WebSphere MQ también da soporte a las carpetas siguientes que ya están en uso por WAS/SIB:

- <sib>

  Esta carpeta se utiliza y se reserva para las propiedades de mensaje del sistema WAS/SIB que no están expuestas como propiedades JMS, o se correlacionan con las propiedades JMS_IBM_ *, pero están expuestas a las aplicaciones WAS/SIB; estas incluyen las propiedades de vías de acceso de direccionamiento inverso y de reenvío.

  Al menos algunas no se pueden exponer como propiedades JMS, porque son matrices de bytes. Si la aplicación añade propiedades a esta carpeta, el valor se ignora o se elimina.

- <sib_usr>

  Esta carpeta se utiliza y se reserva para las propiedades de mensaje de usuario WAS/SIB que no se pueden exponer como propiedades de usuario JMS porque no son de tipos soportados; se exponen a aplicaciones WAS/SIB.

  Estas son propiedades de usuario, que puede obtener o establecer a través de la interfaz SIMessage, pero el contenido de la matriz de bytes se correlaciona con el valor de propiedad necesario.

  Si la aplicación IBM WebSphere MQ escribe un elemento bin.hex arbitrario en la carpeta, la aplicación probablemente recibirá un IOException, ya que no tiene el formato que se espera restaurar. Si añade algo que no sea un elemento bin.hex , recibirá un ClassCastException.

  No intente que las propiedades estén disponibles para WAS/SIB utilizando esta carpeta; en su lugar, utilice la carpeta <usr> para este fin.

- <sib_context>

  Esta carpeta se utiliza para las propiedades de mensaje del sistema WAS/SIB que no están expuestas a las aplicaciones de usuario WAS/SIB o como propiedades JMS. Estas incluyen propiedades de seguridad y transaccionales que se utilizan para servicios web y similares.

  La aplicación no debe añadir propiedades a esta carpeta.

- <mqema>

  WAS/SIB ha utilizado esta carpeta en lugar de la carpeta <mqext> .

Los nombres de carpeta MQRFH2 distinguen entre mayúsculas y minúsculas.

Las carpetas siguientes están reservadas, en cualquier combinación de caracteres en minúsculas o mayúsculas:

- Cualquier carpeta con el prefijo mq o wmq; reservado para su uso por parte de IBM WebSphere MQ.
- Cualquier carpeta con el prefijo sib; reservado para su uso por WAS/SIB.
- Carpetas <Root> y <Body> ; reservadas pero no utilizadas.

Las carpetas siguientes no se reconocen como que contienen propiedades de mensaje:

- <psc>

  Lo utiliza WebSphere Message Broker para transmitir mensajes de mandatos de publicación/suscripción al intermediario.

- <pscr>

  Utilizado por WebSphere Message Broker para contener información del intermediario, en respuesta a mensajes de mandato de publicación/suscripción.

- Cualquier carpeta no definida por WebSphere Message Broker, que no esté marcada con el atributo content='properties' .

No especifique content='properties' en las carpetas <psc> o <pscr> . Si lo hace, estas carpetas se tratan como propiedades y es probable que WebSphere Message Broker deje de funcionar como se esperaba.

Si la aplicación está creando mensajes con propiedades, en las cabeceras MQRFH2 que se van a reconocer como una cabecera MQRFH2 que contiene propiedades, la cabecera debe estar en la lista de cabeceras que se pueden encadenar en la cabecera del mensaje.

La MQRFH2 puede ir precedida de cualquier número de cabeceras estándar MQH, o una MQCIH, una MQDLH, una MQIIH, una MQTM, una MQTMC2o una MQXQH. Una serie o un MQCFH finaliza el análisis porque no se pueden encadenar.

Es posible que un mensaje contenga varias cabeceras MQRFH2 que transporten todas las propiedades del mensaje. Las carpetas con el mismo nombre pueden coexistir en cabeceras diferentes a menos que se restrinja lo contrario, por ejemplo, WAS/SIB. Las carpetas se tratan como una carpeta lógica, si todas están en cabeceras significativas.

Mientras que las carpetas de las cabeceras significativas no se pueden fusionar con esas carpetas en cabeceras no significativas, las carpetas con el mismo nombre dentro de las cabeceras significativas se pueden fusionar, eliminando las propiedades en conflicto. Las aplicaciones no deben depender del diseño de las propiedades dentro de su mensaje.

Los grupos MQRFH2 se analizan para ver las propiedades de las carpetas definidas por el usuario, es decir, no las carpetas <wmq>, <jms>, <mcd>, <usr>, <mqext>, <sib>, <sib_usr>, <sib_context>y <mqema> .

Los grupos de las carpetas de propiedades definidas por IBM, excepto las carpetas <wmq> y <mq> , se analizan para las propiedades.

Una carpeta MQRFH2 no puede contener contenido mixto; una carpeta o grupo puede contener grupos o propiedades, o un valor, pero no ambos.

Un segmento de un mensaje, ya sea el primero o un segmento posterior, no puede contener propiedades definidas por IBM WebSphere MQque no sean las propiedades del descriptor de mensaje. Por lo tanto, colocar un mensaje que contenga dichas propiedades con el conjunto MQMF_SEGMENT o MQMF_SEGMENTATION_ALLOWED hace que la colocación falle con MQRC_SEGMENTATION_NOT_ALLOWED.

Sin embargo, los grupos de mensajes pueden contener propiedades definidas por IBM WebSphere MQ.

Generación de cabeceras MQRFH2

Si WebSphere MQ convierte las propiedades de mensaje a su representación MQRFH2 , debe añadir el MQRFH2 al mensaje. Añade MQRFH2 como una cabecera separada o lo fusiona con una cabecera existente.

La generación de nuevas cabeceras MQRFH2 mediante WebSphere MQ puede interrumpir las cabeceras existentes en un mensaje. Las aplicaciones que analizan un almacenamiento intermedio de mensajes para cabeceras deben tener en cuenta que el número y la posición de las cabeceras en un almacenamiento intermedio pueden cambiar en algunas circunstancias. WebSphere MQ intenta minimizar el impacto de añadir propiedades a un mensaje fusionando propiedades de mensaje en una

cabecera MQRFH2 existente, donde puede. También intenta minimizar el impacto insertando un MQRFH2 generado en una posición fija relativa a otras cabeceras en el almacenamiento intermedio de mensajes.

Una cabecera MQRFH2 generada se coloca a continuación de la MQMDy de cualquier número de cabeceras MQXQH, MQRFHy MQDLH , independientemente del orden en el que se encuentren. La cabecera MQRFH2 generada se coloca inmediatamente antes de la primera cabecera que no es una cabecera MQMD, MQXQH, MQDLHo MQRFH .

Reglas para fusionar los MQRFH2 generados

Las reglas siguientes se aplican a la fusión de un MQRFH2 generado con un MQRFH2existente. La cabecera MQRFH2 generada se fusiona con una cabecera MQRFH2 existente, si:

1. El MQRFH2 existente está en la misma posición WebSphere MQ colocaría un MQRFH2generado o anterior en la cadena de cabecera.
2. El CCSID de las propiedades generadas es el mismo que el NameValueCCSID del MQRFH2existente.

De lo contrario, la cabecera generada se coloca por separado en el almacenamiento intermedio, en la posición descrita anteriormente.

Reglas para fusionar carpetas en un MQRFH2 existente

Si las propiedades de mensaje se fusionan en un MQRFH2existente, el MQRFH2 existente se explora en busca de carpetas que coincidan con las propiedades de mensaje y las fusiona. Si no existe una carpeta coincidente, se añade una carpeta nueva al final de las carpetas existentes. Si existe una carpeta coincidente, se busca en la carpeta. Las propiedades coincidentes se sobrescriben. Las nuevas se añaden al final de la carpeta.

Restricciones de la carpeta MQRFH2

Visión general de las restricciones de carpeta en cabeceras MQRFH2

Las restricciones MQRFH2 se aplican a las carpetas siguientes:

- Los nombres de elemento de la carpeta <usr> no deben empezar por el prefijo JMS; estos nombres de propiedad están reservados para que los utilice JMS y no son válidos para las propiedades definidas por el usuario.

  Un nombre de elemento de este tipo no hace que falle el análisis de MQRFH2 , pero no es accesible para las API de propiedad de mensajes de WebSphere MQ .
- Los nombres de elemento de la carpeta <usr> no pueden ser, en cualquier combinación de minúsculas o mayúsculas, NULL, TRUE, FALSE, NOT, AND, OR, BETWEEN, LIKE, IN, IS y ESCAPE. Estos nombres coinciden con las palabras clave SQL y dificultan el análisis de los selectores, porque <usr> es la carpeta predeterminada que se utiliza cuando no se especifica ninguna carpeta para una propiedad determinada en un selector.

  Un nombre de elemento de este tipo no hace que falle el análisis de MQRFH2 , pero no es accesible para las API de propiedad de mensajes de WebSphere MQ .
- Los nombres de elemento de cualquier carpeta que se considere que contiene propiedades de mensaje no deben contener un punto (.) (Carácter Unicode U+002E), porque se utiliza en los nombres de propiedad para indicar la jerarquía.

  Un nombre de elemento de este tipo no hace que falle el análisis de MQRFH2 , pero no es accesible para las API de propiedad de mensajes de WebSphere MQ .

En general, las cabeceras MQRFH2 que contienen datos de estilo XML válidos pueden ser analizadas por WebSphere MQ sin errores, aunque determinados elementos de MQRFH2 no son accesibles a través de las API de propiedad de mensaje WebSphere MQ .

Conflictos de nombre de elemento MQRFH2

Visión general de los conflictos dentro de los nombres de elemento MQRFH2 .

Sólo se puede adjuntar un valor a una propiedad de mensaje. Si un intento de acceder a una propiedad da lugar a un conflicto de valores, se elige uno con preferencia sobre otro.

La sintaxis de WebSphere MQ para acceder a elementos MQRFH2 permite la identificación exclusiva de un elemento, si una carpeta no contiene elementos con el mismo nombre. Si una carpeta contiene más de un elemento con el mismo nombre, el valor de la propiedad utilizada es el más cercano a la cabecera del mensaje.

Esto se aplica si dos o más carpetas con el mismo nombre están contenidas en diferentes cabeceras MQRFH2 significativas dentro del mismo mensaje.

Puede producirse un conflicto cuando la llamada MQGET se procesa después de que se haya establecido una propiedad de descriptor que no es de mensaje dos veces: a través de una llamada MQSETMP y directamente en la cabecera MQRFH2 en bruto.

Si esto sucede, la propiedad asociada al mensaje por una llamada de API tiene preferencia sobre una en los datos del mensaje, es decir, la de la cabecera MQRFH2 sin formato. Si se produce un conflicto, se considera que se produce lógicamente antes de los datos del mensaje.

Correlación de nombres de propiedad con nombres de carpeta y elemento MQRFH2

Visión general de las diferencias entre los nombres de propiedad y los nombres de elemento en la cabecera MQRFH2 .

Cuando se utiliza cualquiera de las API definidas que finalmente generan cabeceras MQRFH2 , para especificar propiedades de mensaje (por ejemplo, MQ JMS), el nombre de propiedad no es necesariamente el nombre de elemento de la carpeta MQRFH2 .

Por lo tanto, se produce una correlación entre el nombre de propiedad y el elemento MQRFH2 , y de forma inversa, teniendo en cuenta tanto el nombre de carpeta que contiene el elemento como el nombre de elemento. Algunos ejemplos de IBM WebSphere MQ classes for JMS ya están documentados en Utilización de Java.

Nombre de propiedad	Nombre de carpeta de MQRFH2	Nombre de elemento MQRFH2
JMSDestination	jms	Dst
JMSType	mcd	Type, Set, Fmt
xxx (definido por el usuario, donde xxx no empieza por JMS)	usr	xxx

Por lo tanto, cuando una aplicación JMS accede a la propiedad JMSDestination , se correlaciona con el elemento Dst de la carpeta <jms> .

Al especificar propiedades como elementos MQRFH2 , IBM WebSphere MQ define sus elementos de la forma siguiente:

Nombre de propiedad	Nombre de carpeta de MQRFH2	Nombre de grupo MQRFH2	Nombre de elemento MQRFH2
<Property>	<usr>	n/d	<Property>
<folder>.<Property>	<folder>	n/d	<Property>
<folder>.<group>.<Property>	<folder>	<group>	<Property>

Por ejemplo, cuando una aplicación JMS de IBM WebSphere MQ intenta acceder a la propiedad Property1 , esto se correlaciona con el elemento Property1 en la carpeta <usr> . La propiedad wmq.Property2 se correlaciona con la propiedad Property2 en la carpeta <wmq> .

Si el nombre de propiedad contiene más de uno. , el nombre de elemento MQRFH2 utilizado es el que sigue al final. y los grupos MQRFH2 se utilizan para formar una jerarquía; se permiten grupos MQRFH2 anidados.

A la cabecera JMS y a las propiedades específicas de proveedor contenidas en una MQRFH2 en las carpetas <mcd>, <jms>y <mqext> se accede mediante una aplicación IBM WebSphere MQ utilizando los nombres abreviados definidos en Utilización de clases WebSphere MQ para Java.

Se accede a las propiedades definidas por el usuario JMS desde la carpeta <usr> . Una aplicación IBM WebSphere MQ puede utilizar la carpeta <usr> para sus propiedades de aplicación si es aceptable que la propiedad aparezca en las aplicaciones JMS como una de sus propiedades definidas por el usuario.

Si no es aceptable, elija otra carpeta; la carpeta <wmq_usr> se proporciona como ubicación estándar para dichas propiedades no JMS.

Las aplicaciones pueden especificar y utilizar cualquier carpeta MQRFH2 con un uso bien definido, no documentado en "Propiedades especificadas como elementos MQRFH2" en la página 912 si observa lo siguiente:

1. Es posible que la carpeta ya esté en uso, o que la utilice en el futuro, otra aplicación que proporcione acceso no definido a las propiedades contenidas en ella; consulte Nombres de propiedad para ver el convenio de denominación sugerido para los nombres de propiedad.
2. Las propiedades no son accesibles para las versiones anteriores del cliente IBM WebSphere MQ classes for JMS o XMS que solo puede acceder a la carpeta <usr> para las propiedades definidas por el usuario
3. La carpeta debe estar marcada con el atributo `content` con el valor establecido en `properties`, por ejemplo, `content='properties'`.

   "MQSETMP-Establecer propiedad de mensaje" en la página 762 añade automáticamente este atributo según sea necesario. Este atributo no se debe añadir a ninguna de las carpetas definidas por IBM, por ejemplo, <jms> y <usr>. De este modo, hace que el mensaje sea rechazado por el cliente IBM WebSphere MQ classes for JMS anterior a la versión 7.0. con un `MessageFormatException`.

Puesto que la carpeta <usr> es la ubicación predeterminada para las propiedades de la sintaxis <Property> , una aplicación IBM WebSphere MQ y una aplicación JMS para acceder al mismo valor de propiedad definido por el usuario utilizando el mismo nombre.

Nombres de carpeta reservados

Hay varios nombres de carpeta reservados. No puede utilizar nombres como los prefijos de carpeta; por ejemplo, `Root.Property1` no accede a una propiedad válida porque `Root` está reservado. La lista siguiente contiene nombres de carpeta reservados:

- Raíz
- Body (Cuerpo)
- Propiedades
- Entorno
- LocalEnvironment
- DestinationList
- ExceptionList
- InputBody
- InputRoot
- InputProperties
- InputLocalEnvironment
- InputDestinationList
- InputExceptionList
- OutputRoot

- OutputLocalEnvironment
- OutputDestinationList
- OutputExceptionList

Correlación de campos de descriptor de propiedad en cabeceras MQRFH2

Cuando una propiedad se convierte en un elemento MQRFH2 , se utilizan los siguientes atributos de elemento para especificar los campos significativos del descriptor de propiedad: describe cómo se convierten los campos MQPD en atributos del elemento MQRFH2 .

Soporte

El campo Descriptor de propiedad de soporte se divide en tres atributos de elemento

- El atributo de elemento **sr** especifica valores en la máscara de bits MQPD_REJECT_UNSUP_MASK.
- El atributo de elemento **sa** especifica valores en la máscara de bits MQPD_ACCEPT_UNSUP_MASK.
- El atributo de elemento **sx** especifica valores en la máscara de bits MQPD_ACCEPT_UNSUP_IF_XMIT_MASK.

Estos atributos de elemento sólo son válidos en la carpeta < mq> y se ignoran si se establecen en elementos de las otras carpetas que contienen propiedades.

Valor de soporte	Atributo de elemento MQRFH2	Valor de atributo MQRFH2
MQPD_SUPPORT_OPTIONAL	sa	opcional Este es el valor predeterminado.
MQPD_SUPPORT_REQUIRED	sr	necesario
MQPD_SUPPORT_REQUIRED_IF_LOCAL	sx	locales

Contexto

Utilice el atributo de elemento **context** para indicar el contexto de mensaje al que pertenece una propiedad. Utilice un solo valor. Este atributo de elemento es válido en una propiedad de cualquier carpeta que contenga propiedades.

Valor de contexto	Valor de atributo MQRFH2
MQPD_NO_CONTEXT	ninguno Este es el valor predeterminado.
MQPD_USER_CONTEXT	user

CopyOptions

Utilice el atributo de elemento **copy** para indicar los mensajes en los que se debe copiar una propiedad. Se acepta más de un valor; separe varios valores con una coma. Por ejemplo, **copy='reply'** y **copy='publish,report'** son válidos. Este atributo de elemento es válido en una propiedad de cualquier carpeta que contenga propiedades.

Nota: En la definición de atributo, las comillas simples o las comillas dobles son un uso válido, por ejemplo **copy='reply'** o **copy="report"**

Valor de CopyOption	Valor de atributo MQRFH2
MQPD_COPY_FORWARD	hacia adelante

Valor de CopyOption	Valor de atributo MQRFH2
MQPD_COPY_REPLY	responder
MQPD_COPIA_INFORME	informe
MQPD_COPY_PUBLISH	publicar
MQPD_COPY_ALL	Todo No especifique esto con ningún otro valor. Cuando se utiliza con otro valor, tiene prioridad sobre cualquier valor excepto **none**.
MQPD_COPY_DEFAULT	predeterminado Este es el valor predeterminado. Es equivalente a especificar los tres valores MQCOPY_FORWARD, MQCOPY_REPORT y MQCOPY_PUBLISH. No especifique esto con ningún otro valor.
MQPD_COPY_NONE	ninguno No especifique esto con ningún otro valor. Cuando se utiliza con otro valor, esto tiene prioridad.

Restricciones a la carpeta < mq> MQRFH2

Cuando se coloca un mensaje en una cola, se busca una carpeta < mq> para que el mensaje se pueda procesar de acuerdo con sus propiedades definidas por MQ. Para permitir el análisis eficaz de las propiedades definidas por MQ, se aplican las restricciones siguientes a la carpeta:

- MQsólo actúa sobre las propiedades de la primera carpeta < mq> significativa del mensaje; las propiedades de cualquier otra carpeta < mq> del mensaje se ignoran.
- Si la carpeta está en UTF-8, sólo se permiten caracteres UTF-8 de un solo byte en la carpeta. Un carácter de varios bytes en la carpeta puede hacer que falle el análisis y que se rechace el mensaje.
- No incluya grupos MQRFH2 en la carpeta < mq>. La presencia del carácter Unicode U+003C en un valor de propiedad hará que se rechace el mensaje.
- No utilice series de escape en la carpeta. Una serie de escape se trata como el valor real del elemento.
- Sólo el carácter Unicode U+0020 se trata como un espacio en blanco dentro de la carpeta. Todos los demás caracteres se tratan como significativos y pueden hacer que falle el análisis de la carpeta y que se rechace el mensaje.

Si el análisis de la carpeta < mq> falla, o si la carpeta no observa estas restricciones, el mensaje se rechaza con CompCode **MQCC_FAILED** y Reason **MQRC_RFH_RESTRICTED_FORMAT_ERR**.

Cabeceras MQRFH2 que no son válidas

En el momento en que se procesa una llamada MQPUT, MQPUT1o MQGET, se puede producir un análisis parcial de las cabeceras MQRFH2 del mensaje para comprobar qué carpetas se incluyen y para determinar si las carpetas contienen propiedades. Visión general de cabeceras MQRFH2 que no son válidas.

Si el análisis parcial del mensaje no se puede completar correctamente porque la estructura no es válida, por ejemplo, el campo `StrucLength` es demasiado pequeño, entonces:

- La llamada MQPUT o MQPUT1 falla con el código de razón MQRC_RFH_ERROR, si se puede determinar que la aplicación incluye alguna opción de WebSphere MQ Versión 7, para que las aplicaciones existentes no fallen.

- La llamada MQGET se devuelve correctamente y el MQRFH2 que contiene el error se devuelve en el almacenamiento intermedio que ha proporcionado.

Si el análisis parcial falla porque no se puede detectar si una carpeta determinada contiene propiedades o no, por ejemplo, la carpeta empieza `<<jms`, por lo que el análisis falla antes de que se determine el nombre de la carpeta, entonces:

- La llamada MQPUT o MQPUT1 falla con el código de razón MQRC_RFH_FORMAT_ERROR, si se puede determinar que la aplicación incluye alguna opción de WebSphere MQ Versión 7, para que las aplicaciones existentes no fallen.
- La llamada MQGET se devuelve correctamente y el MQRFH2 que contiene el error se devuelve en el almacenamiento intermedio que ha proporcionado.
- Mientras está internamente dentro del gestor de colas, el mensaje no se rechaza debido a la carpeta con formato incorrecto, pero la carpeta siempre se trata como si no hubiera ninguna propiedad dentro de ella.

Un mensaje puede fluir a través de la red de gestores de colas con una carpeta que contenga un error de sintaxis de este tipo, pero que nunca se analice y se detecte, mientras que una o más carpetas del mensaje son:

- Válido
- Se ha analizado correctamente
- Se utiliza en el proceso del mensaje

Por lo tanto, la detección no está garantizada.

Si una de las aplicaciones utiliza "MQSETMP-Establecer propiedad de mensaje" en la página 762, o MQINQMP para acceder a una propiedad, y al hacerlo hace que una carpeta MQRFH2 se analice completamente, detectando un error de forma que el análisis no se puede completar, esto se indica mediante un código de retorno adecuado a la llamada de API. No se pone a disposición de la aplicación ninguna propiedad de la carpeta.

Si se intenta analizar completamente una carpeta MQRFH2 y el analizador encuentra atributos de elemento no reconocidos, o un tipo de datos no reconocido, el análisis continúa y se completa correctamente sin que se emitan avisos; esto no constituye un error de análisis.

Conversión de páginas de códigos

En esta sección se describen los nombres de conjuntos de códigos y los CCSID, el idioma nacional, la conversión z/OS , la conversión IBM i y el soporte de conversión Unicode.

Cada sección de idioma nacional lista la siguiente información:

- Los CCSID nativos soportados
- Las conversiones de página de códigos que **no** están soportadas

En la información se utilizan los términos siguientes:

-8
Indica para HP-UX que el CCSID es para el conjunto de códigos definido de HP-UX *roman8*

AIX
Indica WebSphere MQ para AIX

HP-UX
Indica WebSphere MQ para HP-UX

Linux
Indica WebSphere MQ para Linux para Intel y WebSphere MQ para Linux para zSeries

HP Integrity NonStop Server
Indica WebSphere MQ para HP Integrity NonStop Server

OS/400
Indica WebSphere MQ para IBM i

Solaris
Indica WebSphere MQ para Solaris

Windows
Indica WebSphere MQ para Windows

z/OS
Indica WebSphere MQ para z/OS

El valor predeterminado para la conversión de datos es que la conversión se realice en el sistema de destino (receptor).

Si el producto de origen da soporte a la conversión, se puede configurar un canal y se pueden intercambiar datos estableciendo el atributo de canal CONVERT en YES en el origen.

Nota:

1. La conversión para la información del cliente MQI de WebSphere MQ tiene lugar en el servidor, por lo que el servidor debe dar soporte a la conversión del CCSID del cliente al CCSID del servidor.
2. La conversión puede incluir soporte añadido por CSD/PTF a la última versión de WebSphere MQ. Compruebe el contenido del nivel de servicio más reciente para ver si necesita instalar un CSD/PTF para habilitar esta conversión.

Consulte Tabla 581 en la página 921 para obtener una referencia cruzada entre algunos de los números CCSID y algunos nombres de conjuntos de códigos de la industria.

Nombres de conjuntos de códigos y CCSID

WebSphere MQ para z/OS proporciona más conversión de la que se lista en las tablas específicas del idioma.

Tabla 581. Nombres de conjuntos de códigos y CCSID

Nombres de conjunto de códigos	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 (euros)	923
big5	950
eucJP	954 5050 33722
eucKR	970
eucTW	964
eucCN	1383
pck	943
GBK	1386

Tabla 581. Nombres de conjuntos de códigos y CCSID (continuación)	
Nombres de conjunto de códigos	**CCSID**
koi8-r	878

Idiomas nacionales

Esta información contiene idiomas soportados por WebSphere MQ.

Los idiomas soportados por WebSphere MQ son:

- Inglés de EE.UU.-consulte el tema “Inglés norteamericano” en la página 922
- Alemán-consulte el tema “Alemán” en la página 923
- Danés y noruego-véase el tema “Danés y noruego” en la página 923
- Finés y sueco-véase el tema “Finés y sueco” en la página 924
- Italiano-consulte el tema “Italiano” en la página 925
- Español-consulte el tema “Español” en la página 926
- Inglés/Gaélico del Reino Unido-consulte el tema “Inglés del Reino Unido /Gaélico” en la página 926
- Francés-consulte el tema “Francés” en la página 927
- Multilingüe-consulte el tema “Multi idioma” en la página 927
- Portugués-consulte el tema “Portugués” en la página 928
- Islandés-consulte el tema “Islandés” en la página 929
- Lenguas de Europa oriental-consulte el tema “Lenguas de Europa oriental” en la página 930
- Cirílico-consulte el tema “Cirílico” en la página 931
- Estonio-consulte el tema “Estonio” en la página 931
- Letón y lituano-consulte el tema “Letón y lituano” en la página 932
- Ucraniano-consulte el tema “Ucraniano” en la página 933
- Griego-consulte el tema “Griego” en la página 934
- Turco-consulte el tema “Turco” en la página 935
- Hebreo-consulte el tema “Hebreo” en la página 935
- Farsi-consulte el tema “Farsi” en la página 937
- Urdu-consulte el tema “Urdú” en la página 937
- Tailandés-consulte el tema “Tailandés” en la página 938
- Lao-consulte el tema “lao” en la página 938
- Vietnamita-consulte el tema “Vietnamita” en la página 938
- Japonés latín SBCS-consulte el tema “Japonés latín SBCS” en la página 938
- Japonés Katakana SBCS-consulte el tema “Japonés Katakana SBCS” en la página 940
- Japonés Kanji/latín mixto-consulte el tema “Japonés Kanji/latín mixto” en la página 941
- Japonés Kanji/Katakana mixto-consulte el tema “Japonés Kanji/Katakana mixto” en la página 943
- Coreano-consulte el tema “Coreano” en la página 944
- Chino simplificado-consulte el tema “Chino simplificado” en la página 944
- Chino tradicional-consulte el tema “Chino tradicional” en la página 945

Inglés norteamericano

Detalles de CCSID y conversión de CCSID para inglés de EE.UU.

La tabla siguiente muestra los CCSID nativos para inglés de EE.UU. en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	37, 924, 1140
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 1252, 5348, 858
HP Integrity NonStop Server, Solaris, Linux	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

37
No se convierte a las páginas de códigos 923, 858

924
No se convierte a las páginas de códigos 437, 858, 1051, 1140, 1252, 1275, 5348

1140
No se convierte a las páginas de códigos 924, 1051, 1275

Alemán

Detalles de CCSID y conversión de CCSID para alemán.

La tabla siguiente muestra los CCSID nativos para alemán en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	273, 924, 1141
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 858, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

273
No se convierte a las páginas de códigos 858, 923, 924, 1275

924
No se convierte a las páginas de códigos 273, 437, 858, 1051, 1141, 1252, 1275, 5348

1141
No se convierte a las páginas de códigos 924, 1051, 1275

Danés y noruego

Detalles de CCSID y conversión de CCSID para danés y noruego.

La tabla siguiente muestra los CCSID nativos para danés y noruego en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	277, 924, 1142
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	850, 858, 865, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

277
No se convierte a las páginas de códigos 858, 923, 924, 1275

924
No convierte en páginas de códigos 277, 858, 865, 1051, 1142, 1252, 1275, 5348

1142
No se convierte a las páginas de códigos 924, 865, 1051, 1275

AIX

Página de códigos:

819
No se convierte a la página de códigos 865

HP-UX

Página de códigos:

1051
No se convierte a la página de códigos 865

Windows

Página de códigos:

865
No se convierte a las páginas de códigos 1051, 1275

Finés y sueco

Detalles de CCSID y conversión de CCSID para finés y sueco.

La tabla siguiente muestra los CCSID nativos para finés y sueco en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	278, 924, 1143
AIX	819, 923, 5348
HP-UX	819, 923, 1051

Plataforma	CCSID nativos
Windows	437, 850, 858, 865, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

278
No se convierte a las páginas de códigos 858, 923, 924, 1275

924
No se convierte a las páginas de códigos 278, 437, 858, 865, 1051, 1143, 1252, 1275, 5348

1143
No se convierte a las páginas de códigos 865, 924, 1051, 1275

AIX

Página de códigos:

819
No se convierte a la página de códigos 865

850
No se convierte a la página de códigos 865

HP-UX

Página de códigos:

1051
No se convierte a la página de códigos 865

Windows

Página de códigos:

865
No se convierte a las páginas de códigos 1051, 1275

Italiano

Detalles de CCSID y conversión de CCSID para italiano.

La tabla siguiente muestra los CCSID nativos para italiano en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	280, 924, 1144
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 858, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

280
No se convierte a las páginas de códigos 858, 923, 924, 1275

924
No se convierte a las páginas de códigos 280, 437, 858, 1051, 1144, 1252, 1275, 5348

1144
No se convierte a las páginas de códigos 924, 1051, 1275

Español

Detalles de CCSID y conversión de CCSID para español.

La tabla siguiente muestra los CCSID nativos para español en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	284, 924, 1145
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 858, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

284
No se convierte a las páginas de códigos 858, 923, 924, 1275

924
No se convierte a las páginas de códigos 284, 437, 858, 1051, 1145, 1252, 1275, 5348

1145
No se convierte a las páginas de códigos 924, 1051, 1275

Inglés del Reino Unido /Gaélico

Detalles de CCSID y conversión de CCSID para inglés/gaélico del Reino Unido.

La tabla siguiente muestra los CCSID nativos para inglés/gaélico del Reino Unido en las plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	285, 924, 1146
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 858, 1252, 5348

Plataforma	CCSID nativos
HP Integrity NonStop Server, Solaris, Linux	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

285
No se convierte a las páginas de códigos 858, 923, 924, 1275

924
No se convierte a las páginas de códigos 285, 437, 858, 1051, 1146, 1252, 1275, 5348

1146
No se convierte a las páginas de códigos 924, 1051, 1275

Francés

Detalles de CCSID y conversión de CCSID para francés.

La tabla siguiente muestra los CCSID nativos para francés en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	297, 924, 1147
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 858, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

297
No se convierte a las páginas de códigos 858, 923, 924, 1275, 5348

924
No se convierte a las páginas de códigos 297, 437, 858, 1051, 1147, 1252, 1275, 5348

1147
No se convierte a las páginas de códigos 924, 1051, 1275

Multi idioma

Detalles de CCSID y conversión de CCSID para multilingüe.

La tabla siguiente muestra los CCSID nativos para la conversión multilingüe en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	500, 924, 1148

Plataforma	CCSID nativos
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	437, 850, 858, 1252, 5348
HP Integrity NonStop Server, SINIX, Solaris, Linux	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

500
No se convierte a las páginas de códigos 858, 923

924
No se convierte a las páginas de códigos 437, 858, 1051, 1148, 1252, 1275, 5348

1148
No se convierte a las páginas de códigos 924, 1051, 1275

Portugués

Detalles de CCSID y conversión de CCSID para portugués.

La tabla siguiente muestra los CCSID nativos para portugués en plataformas soportadas:

Plataforma	CCSID nativos
IBM i	37, 500, 924, 1140
z/OS	500, 924, 1140
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	850, 858, 860, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

37
No se convierte a las páginas de códigos 858, 923, 1275

500
No se convierte a las páginas de códigos 858, 923, 1275

924
No se convierte a las páginas de códigos 858, 860, 1051, 1140, 1252, 1275, 5348

1140
No se convierte a las páginas de códigos 860, 924, 1051, 1275

HP-UX

Página de códigos:

1051
No se convierte a la página de códigos 860

Windows

Página de códigos:

860
No se convierte a las páginas de códigos 1051, 1275

Islandés

Detalles de CCSID y conversión de CCSID para islandés.

La tabla siguiente muestra los CCSID nativos para islandés en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	871, 924, 1149
AIX	819, 923, 5348
HP-UX	819, 923, 1051
Windows	850, 858, 861, 1252, 5348
HP Integrity NonStop Server, Solaris, Linux	819, 923
Cliente Apple	1275

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

871
No se convierte a las páginas de códigos 858, 923, 924, 1275, 5348

924
No se convierte a las páginas de códigos 858, 861, 871, 1051, 1149, 1252, 1275, 5348

1149
No se convierte a las páginas de códigos 924, 1051, 1275

HP-UX

Página de códigos:

1051
No se convierte a la página de códigos 861

Windows

Página de códigos:

861
No se convierte a las páginas de códigos 1051, 1275

Lenguas de Europa oriental

Detalles de CCSID y conversión de CCSID para idiomas de Europa oriental. Los idiomas típicos que utilizan estos CCSID son el albanés, el croata, el checo, el húngaro, el polaco, el rumano, el serbio, el eslovaco y el esloveno.

La tabla siguiente muestra los CCSID nativos para idiomas de Europa oriental en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	870, 1153
Windows	852, 1250, 5346, 9044
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux	912
Cliente de Apple de Europa oriental	1282
Cliente rumano de Apple	1285
Cliente de Apple croata	1284

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

z/OS

Página de códigos:

870
No se convierte a las páginas de códigos 1284, 1285

1153
No se convierte a las páginas de códigos 1250, 1284, 1285

IBM i

Página de códigos:

870
No se convierte a las páginas de códigos 1284, 1285, 5346, 9044

1153
No se convierte a las páginas de códigos 1282, 1284, 1285, 5346, 9044

HP-UX, Solaris, Linux

Página de códigos:

912
No se convierte a las páginas de códigos 1284, 1285

HP Integrity NonStop Server

Página de códigos:

912
No se convierte a las páginas de códigos 1153, 1284, 1285, 9044

Windows

Página de códigos:

852
No se convierte a las páginas de códigos 1284, 1285

1250
No se convierte a las páginas de códigos 1284, 1285

9044
No se convierte a las páginas de códigos 912, 1282, 1284, 1285

Cirílico
Detalles de CCSID y conversión de CCSID para cirílico. Los idiomas típicos que utilizan estos CCSID son el bielorruso, el búlgaro, el macedonio, el ruso y el serbio.

La tabla siguiente muestra los CCSID nativos para cirílico en plataformas soportadas:

Plataforma	CCSID nativos
z/OS	1025
IBM i	880, 1025
Windows	855, 866, 1131, 1251, 5347
Solaris	878, 915
AIX, HP-UX, Linux, HP Integrity NonStop Server	915
Cliente Apple	1283

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

880
No se convierte a las páginas de códigos 855, 866, 878, 1131, 5347

1025
No se convierte a las páginas de códigos 878, 5347

Windows

Página de códigos:

855
No se convierte a la página de códigos 1131

866
No se convierte a la página de códigos 1131

1131
No se convierte a las páginas de códigos 855, 866, 880, 1283

Estonio
Detalles de CCSID y conversión de CCSID para estonio.

La tabla siguiente muestra los CCSID nativos para estonio en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	1122, 1157
Windows	902, 922, 1257, 5353, 9449
AIX, HP-UX, Solaris, Linux	902, 922
HP Integrity NonStop Server	922

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

Página de códigos:

1122
No se convierte a las páginas de códigos 902, 1157, 9449

1157
No se convierte a las páginas de códigos 922, 1122, 1257, 9449

IBM i

Página de códigos:

1122
No se convierte a las páginas de códigos 902, 5353, 9449

1157
No se convierte a las páginas de códigos 922, 5353, 9449

HP-UX, Solaris, Linux

Página de códigos:

902
No se convierte a las páginas de códigos 922, 1122, 9449

922
No se convierte a las páginas de códigos 902, 1157, 9449

Windows

Página de códigos:

5353
No se convierte a la página de códigos 9449

9449
No se convierte a las páginas de códigos 902, 922, 1122, 1157, 1257, 5353

902
No se convierte a las páginas de códigos 922, 1122, 9449

HP Integrity NonStop Server

Página de códigos:

922
No se convierte a las páginas de códigos 902, 1157, 9449

Letón y lituano

Detalles de CCSID y conversión de CCSID para letón y lituano.

La tabla siguiente muestra los CCSID nativos para letón y lituano en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	1112, 1156
Windows	901, 921, 1257, 5353, 9449
AIX, HP-UX, Solaris, Linux	901, 921

Plataforma	CCSID nativos
HP Integrity NonStop Server	921

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

Página de códigos:

1112
No se convierte a las páginas de códigos 901, 1156, 9449

1156
No se convierte a las páginas de códigos 901, 1156, 9449

IBM i

Página de códigos:

1112
No se convierte a la página de códigos 5353

1153
No se convierte a las páginas de códigos 921, 5353, 9449

HP-UX, Solaris, Linux

Página de códigos:

902
No se convierte a las páginas de códigos 921, 1112, 1257, 9449

921
No se convierte a las páginas de códigos 901, 1156, 9449

Windows

Página de códigos:

901
No se convierte a las páginas de códigos 921, 1112, 1257, 9449

5355
No se convierte a la página de códigos 9449

9449
No se convierte a las páginas de códigos 901, 921, 1112, 1156, 1257

HP Integrity NonStop Server

Página de códigos:

921
No se convierte a las páginas de códigos 901, 1156, 9449

Ucraniano

Detalles de CCSID y conversión de CCSID para ucraniano.

La tabla siguiente muestra los CCSID nativos para Ukranian en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	1123

Plataforma	CCSID nativos
Windows	1124, 1125, 1251, 5347
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux	1124

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

1123
No se convierte a la página de códigos 5347

HP-UX

Página de códigos:

1124
No se convierte a la página de códigos 5347

Windows

Página de códigos:

1125
No se convierte a la página de códigos 1123

Griego

Detalles de CCSID y conversión de CCSID para griego.

La tabla siguiente muestra los CCSID nativos para griego en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	875
HP-UX	813 (véase nota)
Windows	869, 1253, 5349
AIX, NCR, HP Integrity NonStop Server, Solaris, Linux	813
Cliente Apple	1280
Cliente DOS	737
Nota: Solo se da soporte al conjunto de códigos ISO en HP-UX. El conjunto de códigos HP-UX propietario greek8 no tiene ningún CCSID registrado y no está soportado.	

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos, los CCSID nativos de las otras plataformas con las excepciones siguientes.

IBM i

Página de códigos:

875
No se convierte a la página de códigos 5349

Windows

Página de códigos:

1253
No se convierte a la página de códigos 737

5349
No se convierte a la página de códigos 737

Turco

Detalles de CCSID y conversión de CCSID para turco.

La tabla siguiente muestra los CCSID nativos para turco en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	1026
HP-UX	920 (véase nota)
Windows	857, 1254, 5350
AIX, HP Integrity NonStop Server, Solaris, Linux	920
Cliente Apple	1281
Nota: Solo se da soporte al conjunto de códigos ISO en HP-UX. El conjunto de códigos turkish8 propiedad de HP-UX no tiene ningún CCSID registrado y no está soportado.	

Todas las plataformas que no son de cliente soportan la conversión entre sus CCSID nativos y los CCSID nativos de las otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

1026
No se convierte a la página de códigos 5350

Hebreo

Detalles de CCSID y conversión de CCSID para hebreo.

La tabla siguiente muestra los CCSID nativos para hebreo en plataformas soportadas:

Plataforma	CCSID nativos
z/OS	424, 803, 4899, 12712
IBM i	424
AIX	916, 9048
HP-UX	916 (ver nota)
Windows	1255, 5351
HP Integrity NonStop Server, Solaris, Linux	916
Nota: Solo se da soporte al conjunto de códigos ISO en HP-UX. El conjunto de códigos HP-UX propietario greek8 no tiene ningún CCSID registrado y no está soportado.	

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

Página de códigos:

424
No se convierte a las páginas de códigos 867, 4899, 9048, 12712

803
No se convierte a las páginas de códigos 867, 4899, 5351, 9048, 12712

4899
No se convierte a las páginas de códigos 424, 803, 856, 862, 916, 1255

12712
No se convierte a las páginas de códigos 424, 803, 856, 916, 1255

IBM i

Página de códigos:

424
No se convierte a las páginas de códigos 803, 867, 4899, 5351, 9048, 12712

La página de códigos 424 también se convierte a y desde CCSID 4952, que es una variante de 856.

AIX

Página de códigos:

916
No se convierte a las páginas de códigos 867, 4899, 9048, 12712

9048
No se convierte a las páginas de códigos 424, 803, 856, 862, 916, 1255

Windows

Página de códigos:

1255
No se convierte a las páginas de códigos 867, 4899, 9048, 12712

5351
No se convierte a la página de códigos 803

Árabe

Detalles de CCSID y conversión de CCSID para árabe

La tabla siguiente muestra los CCSID nativos para el árabe en las plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	420
AIX	1046, 1089
HP-UX	1089 (véase nota)
Windows	720, 864, 1256, 5352
HP Integrity NonStop Server, Solaris, Linux	1089
Nota: Solo se da soporte al conjunto de códigos ISO en HP-UX. El conjunto de códigos HP-UX propietario arabic8 no tiene ningún CCSID registrado y no está soportado.	

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

420
No se convierte a la página de códigos 5352

HP-UX, Solaris, Linux, HP Integrity NonStop Server, Tru64

Página de códigos:

1089
No se convierte a la página de códigos 720

Windows

Página de códigos:

720
No se convierte a las páginas de códigos 1089, 5352

5352
No se convierte a la página de códigos 720

Farsi

Detalles de CCSID y conversión de CCSID para farsi.

La tabla siguiente muestra los CCSID nativos para Farsi en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	1097
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux Windows	1098 (véase nota)
Nota: El CCSID nativo para estas plataformas no se ha estandarizado y puede cambiar.	

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas.

Urdú

Detalles de CCSID y conversión de CCSID para Urdu.

La tabla siguiente muestra los CCSID nativos para Urdu en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	918
Windows	868
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux	1006

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

918
No se convierte a la página de códigos 1006

Tailandés

Detalles de CCSID y conversión de CCSID para tailandés.

La tabla siguiente muestra los CCSID nativos para tailandés en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	838
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux Windows	874 (véase nota)
Nota: El CCSID nativo para estas plataformas no se ha estandarizado y puede cambiar.	

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas.

lao

Detalles de CCSID y conversión de CCSID para Lao.

La tabla siguiente muestra los CCSID nativos para Lao en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	1132
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux Windows	1133

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas.

Vietnamita

Detalles de CCSID y conversión de CCSID para vietnamita.

La tabla siguiente muestra los CCSID nativos para vietnamita en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	1130
Windows	1258, 5354
AIX, HP-UX, HP Integrity NonStop Server, Solaris, Linux	1129

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

IBM i

Página de códigos:

1130
No se convierte a las páginas de códigos 1129, 5354

Japonés latín SBCS

Detalles de CCSID y conversión de CCSID para SBCS latino japonés.

La tabla siguiente muestra los CCSID nativos para SBCS latino japonés en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	1027

Plataforma	CCSID nativos
AIX	932, 5050, 33722 (véase la nota 1)
Windows	932, 943 (véanse las notas 2 y 3)
Linux, HP Integrity NonStop Server, Solaris	943, 5050
HP-UX	Desconocido

Nota:

1. 5050 y 33722 son CCSID relacionados con la página de códigos base 954 en AIX. El CCSID notificado por el sistema operativo es 33722.
2. Windows NT utiliza la página de códigos 932, pero esto se representa mejor mediante el CCSID 943. Sin embargo, no todas las plataformas de WebSphere MQ dan soporte a este CCSID.

   En WebSphere MQ para Windows el CCSID 932 se utiliza para representar la página de códigos 932, pero se puede realizar un cambio en el archivo `../conv/table/ccsid.tbl` que cambia el CCSID utilizado a 943.
3. WebSphere MQ no da soporte a páginas de códigos basadas en el estándar JIS X 0213 (JIS2004).

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

Página de códigos:

1027
No se convierte a las páginas de códigos 932, 942, 943, 954, 5050, 33722

IBM i

Página de códigos:

1027
No se convierte a la página de códigos 932

AIX

Página de códigos:

932
No se convierte a la página de códigos 1027

5050
No se convierte a la página de códigos 1027

33722
No se convierte a la página de códigos 1027

Linux

Página de códigos:

943
No se convierte a la página de códigos 1027

5050
No se convierte a la página de códigos 1027

Solaris

Página de códigos:

943
No se convierte a la página de códigos 1027

5050
No se convierte a la página de códigos 1027

HP Integrity NonStop Server

Página de códigos:

943
No se convierte a la página de códigos 1027

5050
No se convierte a la página de códigos 1027

Japonés Katakana SBCS

Detalles de CCSID y conversión de CCSID para japonés Katakana SBCS.

La tabla siguiente muestra los CCSID nativos para japonés Katakana SBCS en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	290
HP-UX	897
AIX	932, 5050, 33722 (véase la nota 1)
Windows	932, 943 (véanse las notas 2 y 3)
Linux, HP Integrity NonStop Server, Solaris	943, 5050

Nota:

1. 5050 y 33722 son CCSID relacionados con la página de códigos base 954 en AIX. El CCSID notificado por el sistema operativo es 33722.
2. Windows NT utiliza la página de códigos 932, pero esto se representa mejor mediante el CCSID 943. Sin embargo, no todas las plataformas de WebSphere MQ dan soporte a este CCSID.

   En WebSphere MQ para Windows el CCSID 932 se utiliza para representar la página de códigos 932, pero se puede realizar un cambio en el archivo `../conv/table/ccsid.tbl` que cambia el CCSID utilizado a 943.
3. WebSphere MQ no da soporte a páginas de códigos basadas en el estándar JIS X 0213 (JIS2004).
4. Además de las conversiones anteriores, los productos WebSphere MQ en AIX, HP-UX, Solaris, Linux y Tru64 dan soporte a la conversión de CCSID 897 a CCSID 37, 273, 277, 278, 280, 284, 285, 290, 297, 437, 500, 819, 850, 1027 y 1252.

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

Página de códigos:

290
No se convierte a las páginas de códigos 932, 943, 954, 5050, 33722

IBM i

Página de códigos:

290
No se convierte a la página de códigos 932

AIX

Página de códigos:

932
No se convierte a las páginas de códigos 290, 897

5050
No se convierte a las páginas de códigos 290, 897

33722
No se convierte a las páginas de códigos 290, 897

HP-UX

Página de códigos:

897
No se convierte a las páginas de códigos 932, 943, 954, 5050, 33722

Linux

Página de códigos:

943
No se convierte a las páginas de códigos 290, 897

5050
No se convierte a las páginas de códigos 290, 897

Solaris

Página de códigos:

943
No se convierte a las páginas de códigos 290, 897

5050
No se convierte a las páginas de códigos 290, 897

HP Integrity NonStop Server

Página de códigos:

943
No se convierte a las páginas de códigos 290, 897

5050
No se convierte a las páginas de códigos 290, 897

Japonés Kanji/latín mixto

Detalles de CCSID y conversión de CCSID para japonés Kanji/Latin Mixed.

La tabla siguiente muestra los CCSID nativos para Kanji japonés/Latin Mixtos en plataformas soportadas:

Plataforma	CCSID nativos
IBM i, z/OS	1399, 5035 (véase la nota 1)

Plataforma	CCSID nativos
AIX	932, 5050, 33722 (véase la nota 2)
HP-UX	932, 954, 5039 (véase la nota 3)
Windows	932, 943 (véanse las notas 4 y 5)
Linux, HP Integrity NonStop Server, Solaris	943, 5050

Nota:

1. 5035 es un CCSID relacionado con la página de códigos 939
2. 5050 y 33722 son CCSID relacionados con la página de códigos base 954 en AIX. El CCSID notificado por el sistema operativo es 33722.
3. Los conjuntos de códigos `japan15` y SJIS en HP-UX se representan mediante CCSID 932. Estos tienen unos pocos caracteres DBCS que tienen representaciones diferentes en SJIS, por lo que 932 se puede convertir incorrectamente si la conversión no se realiza en un sistema HP-UX . WebSphere MQ para HP-UX da soporte a 5039, el CCSID correcto para HP SJIS. Se puede realizar un cambio en el archivo `/var/mqm/conv/ccsid.tbl` para cambiar el CCSID utilizado de 932 a 5039.
4. Windows NT utiliza la página de códigos 932, pero esto se representa mejor mediante el CCSID 943. Sin embargo, no todas las plataformas de WebSphere MQ dan soporte a este CCSID.

   En WebSphere MQ para Windows el CCSID 932 se utiliza para representar la página de códigos 932, pero se puede realizar un cambio en el archivo `../conv/table/ccsid.tbl` que cambia el CCSID utilizado a 943.
5. WebSphere MQ no da soporte a páginas de códigos basadas en el estándar JIS X 0213 (JIS2004).

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

Página de códigos:

1399
No se convierte a las páginas de códigos 954, 5035, 5050, 33722

5035
No se convierte a las páginas de códigos 954, 1399, 5050, 33722

IBM i

Página de códigos:

1399
No se convierte a la página de códigos 5039

5035
No se convierte a la página de códigos 5039

HP-UX

Página de códigos:

932
No se convierte a las páginas de códigos 942, 943, 1399

954
No se convierte a las páginas de códigos 942, 943, 1399

5039
No se convierte a las páginas de códigos 942, 943, 1399

HP Integrity NonStop Server

Página de códigos:

943
No se convierte a la página de códigos 1399

5050
No se convierte a la página de códigos 1399

Japonés Kanji/Katakana mixto

Detalles de CCSID y conversión de CCSID para japonés Kanji/Katakana mixto.

Tabla 582. CCSID nativos para Kanji/Katakana japonés mezclados en plataformas soportadas

Plataforma	CCSID nativos
z/OS	1390, 5026 (véase la nota 1)
IBM i	5026 (véase la nota 1)
AIX	932, 5050, 33722 (véase la nota 2)
HP-UX	932, 954, 5039 (véase la nota 3)
Windows	932, 943 (véanse las notas 4 y 5)
Linux, HP Integrity NonStop Server, Solaris	943, 5050

Nota:

1. CCSID 1390 no acepta caracteres en minúsculas. 5026 es un CCSID relacionado con la página de códigos 930. CCSID 5026 es el CCSID notificado en IBM i cuando se selecciona la característica Katakana japonesa (DBCS).
2. 5050 y 33722 son CCSID relacionados con la página de códigos base 954 en AIX. El CCSID notificado por el sistema operativo es 33722.
3. Los conjuntos de códigos `japan15` y SJIS en HP-UX se representan mediante CCSID 932. Estos tienen unos pocos caracteres DBCS que tienen representaciones diferentes en SJIS, por lo que 932 se puede convertir incorrectamente si la conversión no se realiza en un sistema HP-UX . WebSphere MQ para HP-UX da soporte a 5039, el CCSID correcto para HP SJIS. Se puede realizar un cambio en el archivo `/var/mqm/conv/ccsid.tbl` para cambiar el CCSID utilizado de 932 a 5039.
4. Windows NT utiliza la página de códigos 932, pero esto se representa mejor mediante el CCSID 943. Sin embargo, no todas las plataformas de WebSphere MQ dan soporte a este CCSID.

   En WebSphere MQ para Windows, el CCSID 932 se utiliza para representar la página de códigos 932, pero se puede realizar un cambio en el archivo `../conv/table/ccsid.tbl` que cambia el CCSID utilizado a 943.
5. WebSphere MQ no da soporte a páginas de códigos basadas en el estándar JIS X 0213 (JIS2004).

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

Página de códigos:

1390
No convierte a páginas de códigos 954, 5026, 5050, 33722

No acepta caracteres en minúsculas.

5026
No se convierte a las páginas de códigos 954, 1390, 5050, 33722

IBM i

Página de códigos:

5026
No se convierte a las páginas de códigos 1390, 5039

HP-UX

Página de códigos:

932
No se convierte a las páginas de códigos 942, 943, 1390

954
No se convierte a las páginas de códigos 942, 943, 1390

5039
No se convierte a las páginas de códigos 942, 943, 1390

HP Integrity NonStop Server

Página de códigos:

943
No se convierte a la página de códigos 1390

5050
No se convierte a la página de códigos 1390

Coreano

Detalles de CCSID y conversión de CCSID para coreano.

La tabla siguiente muestra los CCSID nativos para coreano en plataformas soportadas:

Plataforma	CCSID nativos
z/OS, IBM i	933, 1364
AIX, HP-UX, Linux, HP Integrity NonStop Server, Solaris	970
Windows	949, 1363

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

Página de códigos:

933
No se convierte a la página de códigos 970

1364
No se convierte a la página de códigos 970

HP-UX

Página de códigos:

970
No se convierte a las páginas de códigos 949, 1363, 1364

Chino simplificado

Detalles de CCSID y conversión de CCSID para chino simplificado.

La tabla siguiente muestra los CCSID nativos para chino simplificado en plataformas soportadas:

Plataforma	CCSID nativos
z/OS	935, 1388
IBM i	935, 1388
AIX	1383, 1386
HP-UX	1381 (véase la nota 1)
Windows	1381, 1386 (véase nota 2)
Linux, HP Integrity NonStop Server, Solaris	1383

Nota:

1. Los conjuntos de códigos `prc15` y `hp15CN` en HP-UX se representan mediante CCSID 1381.
2. Windows utiliza la página de códigos 936, pero esto se representa mejor mediante el CCSID 1386. Sin embargo, no todas las plataformas de WebSphere MQ dan soporte a este CCSID.

   En WebSphere MQ para Windows el CCSID 1381 se utiliza para representar la página de códigos 936, pero se puede realizar un cambio en el archivo `../conv/table/ccsid.tbl` que cambia el CCSID utilizado a 1386.
3. WebSphere MQ da soporte a la fase uno del estándar chino GB18030 .

   En z/OS, Linux, Windowsy Solaris, el soporte de conversión se proporciona entre Unicode (UTF-8 y UCS-2) y CCSID 1388 (EBCDIC con extensiones GB18030), Unicode (UTF-8 y UCS-2) y CCSID 5488 (GB18030 fase uno), y entre CCSID 1388 y CCSID 5488.

   Nota:

   En IBM i, el sistema operativo proporciona soporte para la conversión entre Unicode (UTF-8 y UCS-2) y CCSID 1388 (EBCDIC con extensiones GB18030).

   En HP-UX actualmente no hay soporte disponible en el sistema operativo HP11 para GB18030. En HP11i, el parche PHCO_26456 proporciona soporte de conversión entre GB18030 (CCSID 5488) y Unicode. No se proporciona soporte para la conversión entre GB18030 y 1388 (EBCDIC).

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

Página de códigos:

935
No se convierte a la página de códigos 1383

1388
No se convierte a la página de códigos 1383

HP-UX

Página de códigos:

1381
No se convierte a las páginas de códigos 1383, 1386, 1388

Chino tradicional

Detalles de CCSID y conversión de CCSID para chino tradicional.

La tabla siguiente muestra los CCSID nativos para chino tradicional en plataformas soportadas:

Plataforma	CCSID nativos
z/OS, IBM i	937
HP-UX	938, 950, 964 (véase la nota)
Windows	950
AIX, HP Integrity NonStop Server, Solaris, Linux	950, 964
Nota: El conjunto de códigos `roc15` en HP-UX está representado por el CCSID 938.	

Todas las plataformas soportan la conversión entre sus CCSID nativos y los CCSID nativos de otras plataformas, con las excepciones siguientes.

z/OS

Página de códigos:

937
No se convierte a la página de códigos 964

1388
No se convierte a la página de códigos 1383

HP-UX

Página de códigos:

938
No se convierte a la página de códigos 948

950
No se convierte a la página de códigos 948

964
No se convierte a la página de códigos 948

Linux, Solaris

Página de códigos:

964
No se convierte a la página de códigos 938

Soporte de conversión Unicode

Algunas plataformas dan soporte a la conversión de datos de usuario a o desde la codificación Unicode. Las dos formas de codificación Unicode soportadas son UCS-2 (CCSID 1200, 13488 y 17584) y UTF-8 (CCSID 1208).

El término *UCS-2* se utiliza a menudo de forma intercambiable pero incorrecta con *UTF-16*. UCS-2 es una codificación de anchura fija donde cada carácter ocupa 2 bytes. UTF-16 es una codificación de anchura variable que es un superconjunto de UCS-2. Además de los caracteres UCS-2 de 2 bytes, UTF-16 contiene caracteres, conocidos como pares suplentes, que tienen 4 bytes de longitud. WebSphere MQ no da soporte a pares suplentes. Por lo tanto, el soporte para UTF-16 y UTF-8 en WebSphere MQ está limitado a los caracteres Unicode que se pueden codificar en UCS-2.

Nota: WebSphere MQ no da soporte a los CCSID del gestor de colas UCS-2 , por lo que los datos de cabecera de mensaje no se pueden codificar en UCS-2.

WebSphere MQ AIX para Unicode

En WebSphere MQ para AIX la conversión a y desde CCSID Unicode está soportada para los CCSID de la tabla siguiente.

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	1200
1153	1156	1157	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

WebSphere MQ HP-UX para Unicode

En WebSphere MQ para HP-UX , la conversión a y desde CCSID Unicode está soportada para los CCSID listados en la tabla siguiente.

437	737	813	819	850	852
855	857	861	864	865	866
869	874	912	915	916	920
932	938	950	954	964	970
1051	1089	1140	1141	1142	1143
1144	1145	1146	1147	1148	1149
1200	1208	1250	1251	1252	1253
1254	1255	1256	1257	1258	1381
5050	5488	13488	33722		

Soporte de WebSphere MQ para Windows, Solaris y Linux para Unicode

En WebSphere MQ para Windows, WebSphere MQ para Solaris y WebSphere MQ para Linux conversión a y desde CCSID Unicode está soportado para los CCSID de la tabla siguiente.

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	903
904	912	913 (5)	915	916	918
920	921	922	923	924	927
928	930	931 (1)	932 (2)	933	935
937	938 (3)	939	941	942	943
947	948	949	950	951	954 (4)
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	1380	1381
1383	1386	1388	4899	5050	5346
5347	5348	5349	5350	5351	5352
5353	5354	5488 (5)	9044	9048	9449
12712	13488	17584	33722 (4)		

Notas:

1. 931 utiliza 939 para la conversión.
2. 932 utiliza 942 para la conversión.
3. 938 utiliza 948 para la conversión.
4. 954 y 33722 utilizan 5050 para la conversión.
5. Sólo en Windows, Linuxy Solaris.

Soporte de IBM i para Unicode

Para obtener detalles sobre el soporte de UNICODE, consulte la publicación correspondiente de IBM i relacionada con el sistema operativo.

Soporte de WebSphere MQ para z/OS para Unicode

En WebSphere MQ para z/OS la conversión a y desde los CCSID Unicode está soportada para los CCSID siguientes:

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	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	1.012	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	5346	5347	5348	5349	5350
5351	5352	5353	5354	5488	8482
8612	9027	9030	9044	9048	9049

9056	9061	9066	9238	9449	12712
13121	13218	13488	16684	16804	17248
17584	21427	28709			

Estándares de codificación en plataformas de 64 bits

Utilice esta información para obtener información sobre los estándares de codificación en plataformas de 64 bits y los tipos de datos preferidos.

Tipos de datos preferidos

Estos tipos nunca cambian de tamaño y están disponibles en plataformas WebSphere MQ de 32 bits y 64 bits:

Nombre	Longitud
MQLONG	4 bytes
MQULONG	4 bytes
MQINT32	4 bytes
MQUINT32	4 bytes
MQINT64	8 bytes
MQUINT64	8 bytes

Tipos de datos estándar

Conozca los tipos de datos estándar en aplicaciones UNIXde 32 bits, UNIXde 64 bits y Windows de 64 bits.

Aplicaciones UNIX de 32 bits

Esta sección se incluye para la comparación y se basa en Solaris. Las diferencias con otras plataformas UNIX se anotan:

Nombre	Longitud
carácter	1 byte
corto	2 bytes
int	4 bytes
largo	4 bytes
flotante	4 bytes
doble	8 bytes
doble largo	16 bytes Tenga en cuenta que en AIX y Linux PPC un doble largo es de 8 bytes.
puntero	4 bytes
ptrdiff_t	4 bytes
tamaño_t	4 bytes
tiempo_t	4 bytes
Clock_t	4 bytes

Nombre	Longitud
wchar_t	4 bytes
	Tenga en cuenta que en AIX un wchar_t es de 2 bytes.

Aplicaciones UNIX de 64 bits

Esta sección se basa en Solaris. Las diferencias con otras plataformas UNIX se anotan:

Nombre	Longitud
carácter	1 byte
corto	2 bytes
int	4 bytes
largo	8 bytes
flotante	4 bytes
doble	8 bytes
doble largo	16 bytes
	Tenga en cuenta que en AIX y Linux PPC un doble largo es de 8 bytes.
puntero	8 bytes
ptrdiff_t	8 bytes
tamaño_t	8 bytes
tiempo_t	8 bytes
Clock_t	8 bytes
	Tenga en cuenta que en las otras plataformas UNIX un clock_t es de 4 bytes.
wchar_t	4 bytes
	Tenga en cuenta que en AIX un wchar_t es de 2 bytes.

Aplicaciones Windows de 64 bits

Nombre	Longitud
carácter	1 byte
corto	2 bytes
int	4 bytes
largo	4 bytes
flotante	4 bytes
doble	8 bytes
doble largo	8 bytes
puntero	8 bytes
	Tenga en cuenta que todos los punteros son de 8 bytes.
ptrdiff_t	8 bytes
tamaño_t	8 bytes

Nombre	Longitud
tiempo_t	8 bytes
Clock_t	4 bytes
wchar_t	2 bytes
Word	2 bytes
DWORD	4 bytes
HANDLE	8 bytes
HFile	4 bytes

Consideraciones de codificación en Windows

HANDLE hf;

Uso

```
hf = CreateFile((LPCTSTR) FileName,
                Access,
                ShareMode,
                xihSecAttsNTRestrict,
                Create,
                AttrAndFlags,
                NULL);
```

No utilizar

```
HFILE hf;
hf = (HFILE) CreateFile((LPCTSTR) FileName,
                        Access,
                        ShareMode,
                        xihSecAttsNTRestrict,
                        Create,
                        AttrAndFlags,
                        NULL);
```

ya que esto produce un error.

size_t len fgets

Uso

```
size_t len
while (fgets(string1, (int) len, fp) != NULL)
len = strlen(buffer);
```

No utilizar

```
int len;

while (fgets(string1, len, fp) != NULL)
len = strlen(buffer);
```

printf

Uso

```
printf("My struc pointer: %p", pMyStruc);
```

No utilizar

```
printf("My struc pointer: %x", pMyStruc);
```

Si necesita salida hexadecimal, tiene que imprimir los 4 bytes superiores e inferiores por separado.

char * ptr

Uso

```
char * ptr1;
char * ptr2;
size_t bufLen;

bufLen = ptr2 - ptr1;
```

No utilizar

```
char *ptr1;
char *ptr2;
UINT32 bufLen;

bufLen = ptr2 - ptr1;
```

alignBytes

Uso

```
alignBytes = (unsigned short) ((size_t) address % 16);
```

No utilizar

```
void *address;
unsigned short alignBytes;

alignBytes = (unsigned short) ((UINT32) address % 16);
```

lon

Uso

```
len = (UINT32) ((char *) address2 - (char *) address1);
```

No utilizar

```
void *address1;
void *address2;
UINT32 len;

len = (UINT32) ((char *) address2 - (char *) address1);
```

sscanf

Uso

```
MQLONG SBCSprt;

sscanf(line, "%d", &SBCSprt);
```

No utilizar

```
MQLONG SBCSprt;

sscanf(line, "%1d", &SBCSprt);
```

`%ld` intenta poner un tipo de 8 bytes en un tipo de 4 bytes; solo utilice `%l` si está tratando con un tipo de datos `long` real. MQLONG, UINT32 y INT32 se definen como de cuatro bytes, los mismos que un `int` en todas las plataformas WebSphere MQ :

Referencia SOAP

Transporte de WebSphere MQ para información de referencia SOAP ordenada alfabéticamente.

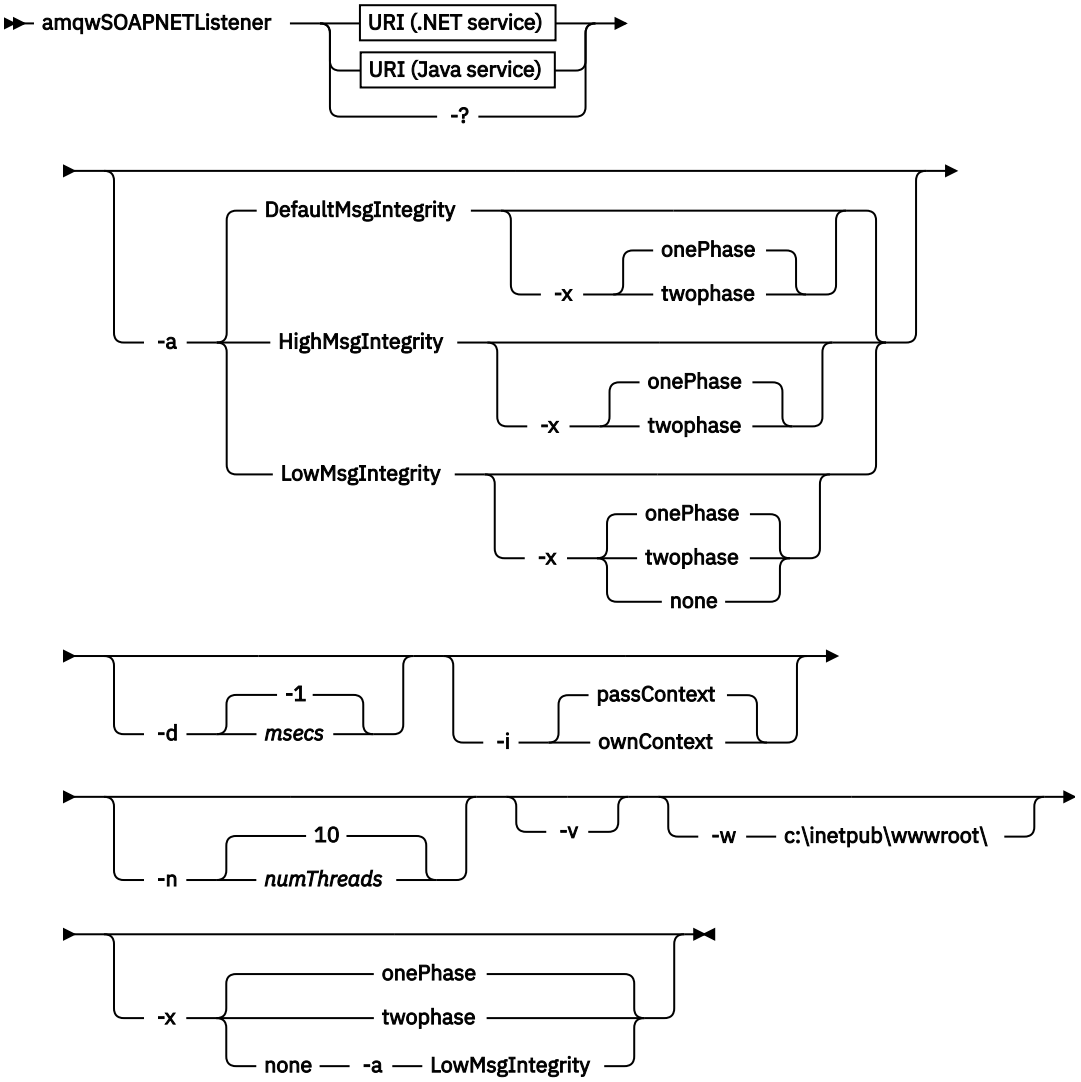
amqwSOAPNETListener: IBM WebSphere MQ escucha SOAP para .NET Framework 1 o 2

Sintaxis y parámetros para el escucha SOAP de WebSphere MQ para .NET Framework 1 o 2.

Finalidad

Inicia el escucha SOAP de IBM WebSphere MQ para .NET Framework 1 o 2.

.NET



Parámetros necesarios

URI ***plataforma***
Consulte “Sintaxis de URI y parámetros para el despliegue de servicios web” en la página 993.

-?
Imprima el texto de ayuda que describe cómo se utiliza el mandato.

Parámetros opcionales

-a ***integrityOption***
integrityOption especifica el comportamiento de los escuchas SOAP de WebSphere MQ si no es posible colocar un mensaje de solicitud fallida en la cola de mensajes no entregados. *integrityOption* puede tomar uno de los valores siguientes:

DefaultMsgIntegrity
Para los mensajes no persistentes, el escucha muestra un mensaje de aviso y continúa ejecutándose con el mensaje original que se descarta. Para los mensajes persistentes, muestra un mensaje de error, restituye el mensaje de solicitud para que permanezca en la cola de solicitudes y salga. DefaultMsgIntegrity se aplica si se omite la opción -a o si no se especifica *integrityOption* .

LowMsgIntegrity
Para los mensajes persistentes y no persistentes, el escucha muestra un aviso y continúa ejecutándose, descartando el mensaje.

HighMsgIntegrity
Para los mensajes persistentes y no persistentes, el escucha muestra un mensaje de error, restituye el mensaje de solicitud para que permanezca en la cola de solicitudes y salga.

El programa de utilidad de despliegue comprueba la compatibilidad de los distintivos -x y -a . Si se especifica -x none , se debe especificar -a LowMsgIntegrity . Si los distintivos son incompatibles, el programa de utilidad de despliegue sale con un mensaje de error y no se han realizado pasos de despliegue.

-d ***msegs***
msegs especifica el número de milisegundos para que el escucha SOAP de WebSphere MQ permanezca activo si se han recibido mensajes de solicitud en cualquier hebra. Si *msegs* se establece en -1, el escucha permanece activo indefinidamente.

-i ***Contexto***
Contexto especifica si los escuchas pasan el contexto de identidad. *Contexto* toma los valores siguientes:

passContext
Establezca el contexto de identidad del mensaje de solicitud original en el mensaje de respuesta. El escucha SOAP comprueba que tiene autorización para guardar el contexto de la cola de solicitudes y pasarlo a la cola de respuestas. Realiza las comprobaciones en tiempo de ejecución al abrir la cola de solicitudes para guardar el contexto y la cola de respuestas para pasar el contexto. Si no tiene la autorización necesaria, o la llamada MQOPEN falla, y el mensaje de respuesta no se procesa. El mensaje de respuesta se coloca en la cola de mensajes no entregados con la cabecera de mensaje no entregado que contiene el código de retorno del MQOPENanómalo. A continuación, el escucha continúa procesando los mensajes entrantes subsiguientes de forma normal.

ownContext
El escucha SOAP no pasa contexto. El contexto devuelto refleja el ID de usuario bajo el que se ejecuta el escucha en lugar del ID de usuario que ha creado el mensaje de solicitud original.

Los campos en el contexto de origen los establece el gestor de colas y no el escucha SOAP.

-n ***numThreads***
numThreads especifica el número de hebras en los scripts de inicio generados para el escucha SOAP de WebSphere MQ . El valor predeterminado es 10. Considere la posibilidad de aumentar este número si tiene un alto rendimiento de mensajes.

-v
-v establece la salida detallada de los mandatos externos. Los mensajes de error siempre se visualizan. Utilice -v para generar mandatos que puede adaptar para crear scripts de despliegue personalizados.

-w ***serviceDirectory***
serviceDirectory es el directorio que contiene el servicio web.

-x ***transaccionalidad***
transaccionalidad especifica el tipo de control transaccional para el escucha. La *transaccionalidad* se puede establecer en uno de los valores siguientes:

onePhase
Se utiliza el soporte de una fase de IBM WebSphere MQ . Si el sistema falla durante el proceso, el mensaje de solicitud se vuelve a entregar a la aplicación. Las transacciones de WebSphere MQ garantizan que los mensajes de respuesta se escriban exactamente una vez.

twoPhase
Se utiliza el soporte de dos fases. Si el servicio se escribe correctamente, el mensaje se entrega exactamente una vez, coordinado con otros recursos, dentro de una única ejecución confirmada del servicio. Esta opción sólo se aplica a las conexiones de enlaces de servidor.

none
Sin soporte transaccional. Si el sistema falla durante el proceso, el mensaje de solicitud se puede perder, aunque sea persistente. Es posible que el servicio se haya ejecutado o no, y que se hayan escrito mensajes de respuesta, de informe o de mensajes no entregados.

El programa de utilidad de despliegue comprueba la compatibilidad de los distintivos -x y -a . Consulte la descripción del distintivo -a para obtener detalles.

Ejemplo de .NET

```
amqwSOAPNETlistener
-u "jms:/queue?destination=myQ&connectionFactory=()
&targetService=myService&initialContextFactory=com.ibm.mq.jms.Nojndi"
-w C:/wmqsoap/demos
-n 20
```

amqswsdl: generar WSDL para el servicio .NET Framework 1 o 2

amqswsdl toma un servicio web escrito para .NET Framework 1 o 2, y genera el WSDL para la clase, insertando el URI que proporciona para el transporte WebSphere MQ para SOAP en el WSDL generado.

Finalidad

Utilice **amqswsdl** para generar WSDL que contenga el URI del servicio desplegado en WebSphere MQ. Utilice el WSDL para generar proxies de cliente.

Parámetros

escapedUri **(Entrada)**
El URI del servicio, con todos los caracteres de escape "&" en "&amp.". Por ejemplo:

```
"jms:/queue?destination=REQUESTDOTNET
&amp.initialContextFactory=com.ibm.mq.jms.Nojndi
&amp.connectionFactory=(connectQueueManager(QM1)binding(server))
&amp.targetService=Quote.asmx"
```

className**.asmx (Entrada)**
La clase de servicio.

className**.wsdl (Salida)**
El WSDL de servicio.

Descripción

Si la clase se implementa utilizando el modelo de programación de código posterior, debe crear *className*`.dll` y almacenarlo en `./bin`.

`amqwclientconfig`: crear descriptor de despliegue de cliente de servicios web de Axis 1.4 para el transporte WebSphere MQ para SOAP

`amqwclientconfig` crea el archivo de descriptor de despliegue del cliente `client-config.wsdd` Axis 1.4 .

Finalidad

Añade el transporte `jms:/` al descriptor y registra `java:com.ibm.mq.soap.transport.jms.WMQSender` como la clase para manejar solicitudes SOAP para el transporte `jms:` .

Sintaxis

amqwclientconfig

Descripción

`amqwclientconfig` llama a **`amqwsetcp`** para establecer la CLASSPATH y ejecuta el mandato:

```
java org.apache.axis.utils.Admin client "%WMQSOAP_HOME%\bin\amqwclientTransport.wsdd"
```

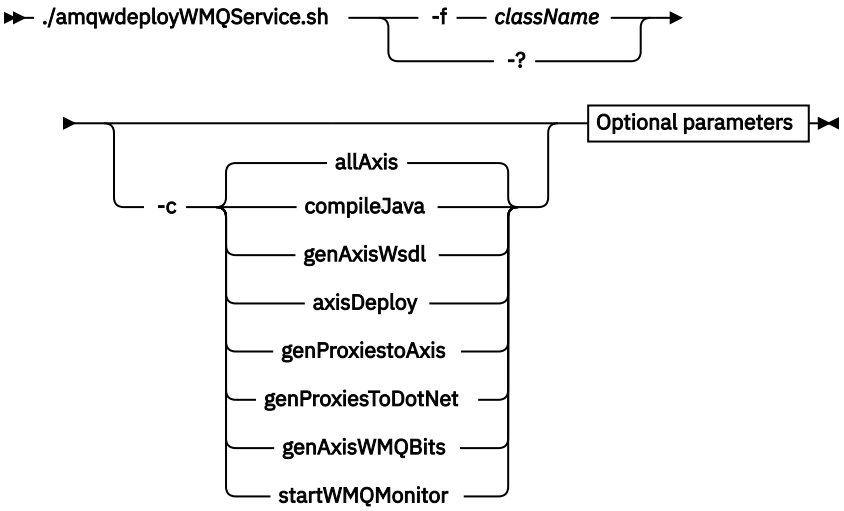
`amqwdeployWMQService`: programa de utilidad de despliegue de servicio web

El programa de utilidad de despliegue prepara una clase de servicio para utilizarla como servicio web utilizando WebSphere MQ como transporte.

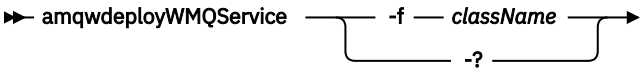
Finalidad

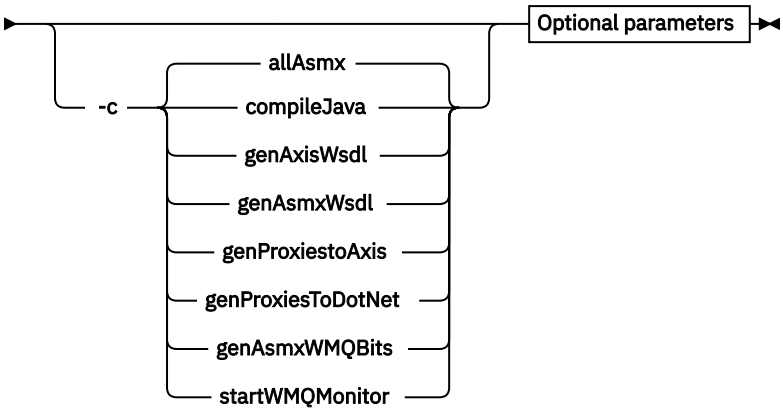
Utilice el programa de utilidad de despliegue para generar los archivos necesarios para desplegar un servicio Axis 1.4, .NET Framework 1 o .NET Framework 2. Utilice los archivos para desplegar un servicio invocado por IBM WebSphere MQ. Los archivos generados por **`amqwdeployWMQService`** se muestran en “Archivos de salida de amqwdeployWMQService” en la página 962.

Syntax diagram
UNIX and Linux systems

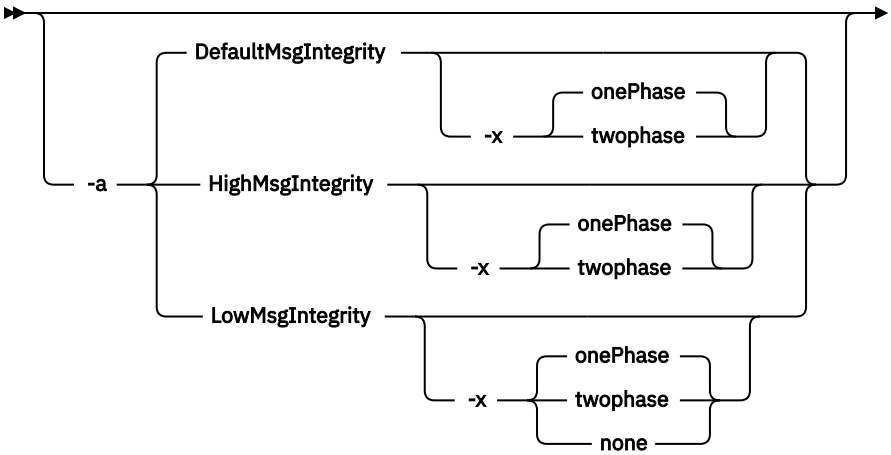


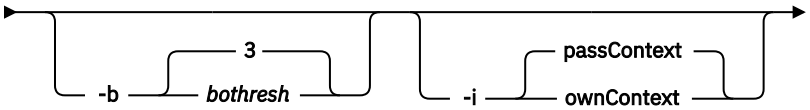
Windows

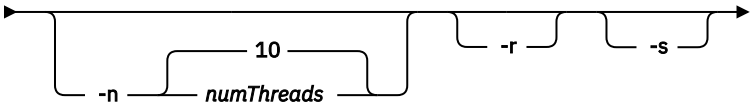


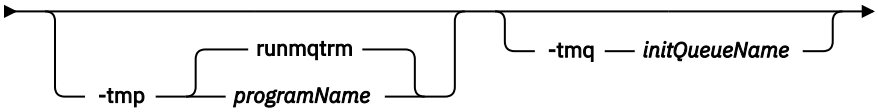


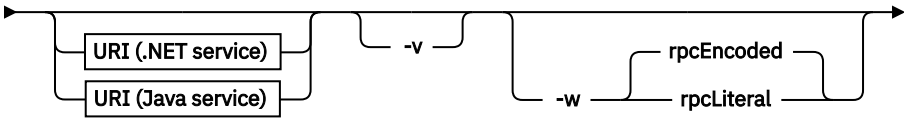
Optional parameters

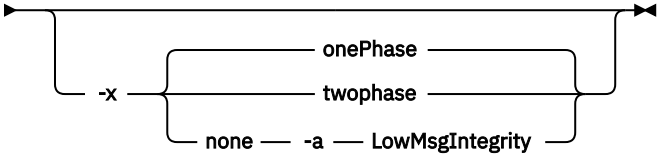












Parámetros necesarios

-f ***className***
className es el nombre de la clase que se va a desplegar. Para los servicios de Axis, *className* es el archivo de origen Java y, para los servicios .NET, el archivo `.asmx` . Figura 11 en la página 959 ilustra el despliegue de un servicio Axis y Figura 12 en la página 959 de un servicio .NET.

```
amqwdeployWMQService -f javaDemos/service/StockQuoteAxis.java
```

Figura 11. Ejemplo de despliegue del servicio Axis

```
amqwdeployWMQService -f StockQuoteDotNet.asmx
```

Figura 12. Ejemplo de despliegue del servicio .NET

Para Java, *className* debe estar totalmente calificado por el nombre de paquete. Se puede especificar como un nombre de vía de acceso con separadores de directorio o como un nombre de clase con separadores de punto. La clase generada se encuentra en `./generated/client/remote/`*`path name`*. Para un servicio .NET, aunque se puede especificar el directorio, los proxies Java generados siempre se encuentran en `./generated/client/remote/dotNetService`.

Si especifica un URI con la opción -u y dentro del URI especifica *targetService*, el programa de utilidad de despliegue comprueba el *className*. *className* debe coincidir con *targetService*. Si la clase y el servicio no coinciden, el programa de utilidad de despliegue muestra un mensaje de error y sale.

-?
Imprima el texto de ayuda que describe cómo se utiliza el mandato.

Parámetros opcionales

-a ***integrityOption***
integrityOption especifica el comportamiento de los escuchas SOAP de WebSphere MQ si no es posible colocar un mensaje de solicitud fallida en la cola de mensajes no entregados. *integrityOption* puede tomar uno de los valores siguientes:

DefaultMsgIntegrity
Para los mensajes no persistentes, el escucha muestra un mensaje de aviso y continúa ejecutándose con el mensaje original que se descarta. Para los mensajes persistentes, muestra un mensaje de error, restituye el mensaje de solicitud para que permanezca en la cola de solicitudes y salga. DefaultMsgIntegrity se aplica si se omite la opción -a o si no se especifica *integrityOption* .

LowMsgIntegrity
Para los mensajes persistentes y no persistentes, el escucha muestra un aviso y continúa ejecutándose, descartando el mensaje.

HighMsgIntegrity
Para los mensajes persistentes y no persistentes, el escucha muestra un mensaje de error, restituye el mensaje de solicitud para que permanezca en la cola de solicitudes y salga.

El programa de utilidad de despliegue comprueba la compatibilidad de los distintivos -x y -a . Si se especifica -x none , se debe especificar -a LowMsgIntegrity . Si los distintivos son incompatibles, el programa de utilidad de despliegue sale con un mensaje de error y no se han realizado pasos de despliegue.

-b ***umbral***
bothresh especifica el valor de umbral de restitución para la cola de solicitudes. El valor predeterminado es 3.

-c ***operación***
operation especifica qué parte del proceso de despliegue se debe ejecutar. *operation* es una de las opciones siguientes:

allAxis
Realizar todos los pasos de compilación y configuración para un servicio Axis o Java[4].

compileJava
Compile el servicio Java: `.java` en `.class`.

genAxisWsdl
Generar WSDL: `.class` en `.wsdl`.

axisDeploy
Despliegue el archivo de clase: `.wsdl` en `.wsdd`, aplique `.wsdd`.

genProxiestoAxis
Generar proxies: `.wsdl` en `.java` y `.class`.

genAxisWMQBits
Configure colas de IBM WebSphere MQ , escuchas SOAP de IBM WebSphere MQ y desencadenantes para un servicio Axis.

allAsmx
Realizar todos los pasos de configuración para un servicio .NET[5].

genAsmxWsdl
Generar WSDL: `.asmx` en `.wsdl`.

genProxiesToDotNet
Generar proxies: `.wsdl` en `.java`, `.class`, `.cs` y `.vb`.

genAsmxWMQBits
Configurar colas de IBM WebSphere MQ , escuchas SOAP de IBM WebSphere MQ y desencadenantes

startWMQMonitor
Inicie el supervisor desencadenante para los servicios SOAP de WebSphere MQ .

Nota: runmqtrm se ejecuta bajo el ID de usuario `mqm` . Si la seguridad es un problema, debe asegurarse de que los escuchas se inician con los ID de usuario adecuados.

-i *Contexto*
Contexto especifica si los escuchas pasan el contexto de identidad. *Contexto* toma los valores siguientes:

passContext
Establezca el contexto de identidad del mensaje de solicitud original en el mensaje de respuesta. El escucha SOAP comprueba que tiene autorización para guardar el contexto de la cola de solicitudes y pasarlo a la cola de respuestas. Realiza las comprobaciones en tiempo de ejecución al abrir la cola de solicitudes para guardar el contexto y la cola de respuestas para pasar el contexto. Si no tiene la autorización necesaria, o la llamada `MQOPEN` falla, y el mensaje de respuesta no se procesa. El mensaje de respuesta se coloca en la cola de mensajes no entregados con la cabecera de mensaje no entregado que contiene el código de retorno del `MQOPEN`anómalo. A continuación, el escucha continúa procesando los mensajes entrantes subsiguientes de forma normal.

ownContext
El escucha SOAP no pasa contexto. El contexto devuelto refleja el ID de usuario bajo el que se ejecuta el escucha en lugar del ID de usuario que ha creado el mensaje de solicitud original.

Los campos en el contexto de origen los establece el gestor de colas y no el escucha SOAP.

-n *numThreads*
numThreads especifica el número de hebras en los scripts de inicio generados para el escucha SOAP de WebSphere MQ . El valor predeterminado es 10. Considere la posibilidad de aumentar este número si tiene un alto rendimiento de mensajes.

-r
-r especifica que se sustituirán las definiciones de cola de supervisor desencadenante o de solicitud existentes. Las colas de supervisor desencadenante sólo se sustituyen si también se especifica -tmq . Las colas se vuelven a crear con atributos predeterminados estándar y los mensajes existentes en

[4] Valor predeterminado si *className* tiene una extensión .java
[5] Valor predeterminado si *className* tiene una extensión .asmx.

las colas se suprimen. Si no se utiliza la opción -r , las definiciones de cola existentes no se alteran y los mensajes existentes no se suprimen. Al no especificar -r, se asegura de que se conserven los atributos de cola personalizados.

-s
Configure el escucha para que se ejecute como un servicio de WebSphere MQ . Si se especifican -s y -tmq , el programa de utilidad de despliegue muestra un mensaje de error y sale.

-tmp ***programName***
programName especifica el nombre de un programa supervisor desencadenante. Utilice `-tmp` *programName* en un entorno UNIX o Linux como alternativa al uso de **runmqtrm**. Los programas que inicia se ejecutan bajo la autorización `mqm` .

Por ejemplo:

```
amqwdeployWMQService -f javaDemos/service/StockQuoteAxis.java
-tmq trigger.monitor.queue -tmp trigmon
```

-tmq ***queueName***
queueName especifica un nombre de cola de supervisor desencadenante. Las definiciones de proceso de IBM WebSphere MQ se crean para configurar el desencadenamiento automático de escuchas SOAP de WebSphere MQ con el nombre de cola de supervisor desencadenante asociado. Si no se especifica la opción, el programa de utilidad de despliegue no define ninguna configuración de desencadenamiento. Si se especifican -s y -tmq , el programa de utilidad de despliegue muestra un mensaje de error y sale.

URI ***plataforma***
Consulte “Sintaxis de URI y parámetros para el despliegue de servicios web” en la página 993.

-v
-v establece la salida detallada de los mandatos externos. Los mensajes de error siempre se visualizan. Utilice -v para generar mandatos que puede adaptar para crear scripts de despliegue personalizados.

-w
-w controla el estilo de WSDL que se va a generar. El valor predeterminado es rpcEncloded, por compatibilidad con releases anteriores de WebSphere MQ Transport para SOAP. Utilice rpcLiteral para crear WSDL compatible con la generación de proxy de cliente Axis2 . rpcEncoded no es compatible con las recomendaciones WS-I.

-x ***transaccionalidad***
transaccionalidad especifica el tipo de control transaccional para el escucha. La *transaccionalidad* se puede establecer en uno de los valores siguientes:

onePhase
Se utiliza el soporte de una fase de IBM WebSphere MQ . Si el sistema falla durante el proceso, el mensaje de solicitud se vuelve a entregar a la aplicación. Las transacciones de WebSphere MQ garantizan que los mensajes de respuesta se escriban exactamente una vez.

twoPhase
Se utiliza el soporte de dos fases. Si el servicio se escribe correctamente, el mensaje se entrega exactamente una vez, coordinado con otros recursos, dentro de una única ejecución confirmada del servicio. Esta opción sólo se aplica a las conexiones de enlaces de servidor.

none
Sin soporte transaccional. Si el sistema falla durante el proceso, el mensaje de solicitud se puede perder, aunque sea persistente. Es posible que el servicio se haya ejecutado o no, y que se hayan escrito mensajes de respuesta, de informe o de mensajes no entregados.

El programa de utilidad de despliegue comprueba la compatibilidad de los distintivos -x y -a . Consulte la descripción del distintivo -a para obtener detalles.

Errores

En Windows, si se notifican errores desde **amqswsdl**, intente emitir el mandato siguiente para registrar los archivos `.asmx` como servicios.

```
%windir%/Microsoft.NET/Framework/version number/aspnet_regiis.exe -ir
```

El problema normalmente se produce en sistemas en los que no se ha instalado IIS o en los que se ha instalado IIS después de NET. El problema se encuentra cuando **amqswsdl** genera los archivos `.wsdl` .

Nota: Las claves de registro también son necesarias para permitir que el escucha invoque los servicios. Si utiliza sus propios procedimientos de despliegue personalizados, es posible que no encuentre el problema hasta el tiempo de ejecución.

Archivos de salida de amqwdeployWMQService

Una lista de los directorios y archivos de salida de **amqwdeployWMQService**

*Tabla 583. Archivos de salida de **amqwdeployWMQService***

Resultados	Descripción	Directorio de salida	Nombre de archivo
`.class`	Archivo fuente Java compilado	`./generated/server/`*server package*	*classname*`.class`
`.wsdl`	Descripción del servicio	`./generated`	*className*`Axis_Wmq.wsdl` *className*`DotNet_Wmq.wsdl`
`.wsdd`	Archivos de despliegue de cliente y servicio de Axis	`./`	`client-config.wsdd` `server-config.wsdd`
		`./generated/server/`*server package*	*className*`_deploy.wsdd` *className*`_undeploy.wsdd`
Origen de cliente (`.vb`, `.cs`, `.java`)	Apéndices de cliente .Net para el servicio Axis	`./generated/client`	*classname*`AxisService.cs` *classname*`AxisService.vb`
	Servicio .Net de apéndices de cliente a. Net	`./generated/client`	*classname*`DotNet.cs` *classname*`DotNet.vb`

*Tabla 583. Archivos de salida de **amqwdeployWMQService** (continuación)*			
Resultados	**Descripción**	**Directorio de salida**	**Nombre de archivo**
Ayudante de cliente (.java y .class)	Proxy de cliente Java al servicio .Net	./generated/server/soap/client/remote/dotnetService	*className*DotNet.class *className*DotNet.java *className*DotNetLocator.class *className*DotNetLocator.java *className*DotNetSoap12Stub.class *className*DotNetSoap12Stub.java *className*DotNetSoap_BindingStub.class *className*DotNetSoap_BindingStub.java *className*DotNetSoap_PortType.class *className*DotNetSoap_PortType.java
	Proxy de cliente Java para el servicio Axis	./generated/server/soap/client/remote/*client package*	SoapServer*className*AxisBindingSoapStub.class SoapServer*className*AxisBindingSoapStub.java *className*Axis.class *className*Axis.java *className*AxisService.class *className*AxisService.java *className*AxisServiceLocator.class *className*AxisServiceLocator.java
Scripts (.cmd y .sh)	Scripts de escucha	/generated/server	startWMQJListener.cmd startWMQJListener.sh startWMQNListener.cmd endWMQJListener.cmd endWMQJListener.sh endWMQNListener.cmd

Notas de uso para amqwdeployWMQService

Describe las tareas realizadas por **amqwdeployWMQService**.

El programa de utilidad de despliegue realiza las acciones siguientes.

1. Comprueba las vías de acceso a los archivos siguientes:
   - axis.jar.
   - *WMQSOAP_HOME*/java/lib/com.ibm.mq.soap.jar.
   - En Windows, csc.exe
2. En Windows, utiliza %SystemRoot%\Microsoft.NET\Framework\v1.1.432 o, si el compilador C# está instalado, la vía de acceso a csc.exe como vía de acceso a .NET Framework.

   Nota: Si tiene instalado Microsoft Visual Studio 2008 (Versión 9), wsdl.exe no está en la vía de acceso a csc.exe. Debe añadir la vía de acceso a .NET Framework a la variable Path; por ejemplo:

   ```
   Set Path=C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727;%Path%
   ```
3. Crea el directorio ./generated y los subdirectorios necesarios, si no existen.
4. Para servicios Java, compila el origen en *className*.class.
5. Genera WSDL.

6. Para los servicios Java, crea los archivos de descriptor de despliegue *className*_deploy.wsdd y *className*_undeploy.wsdd
7. Para servicios Java, crea o actualiza el archivo de descriptor de despliegue de Axis, server-config.wsdd.
8. Genera los proxies de cliente para Java, C# y Visual Basic a partir del WSDL.

   Nota: En Windows, el programa de utilidad de despliegue genera proxies para Visual Basic y C# independientemente del idioma en el que se escriba el servicio. El WSDL y los proxies generados a partir del mismo incluyen el URI adecuado para llamar al servicio:

   a.
   ```
   jms:/queue?destination=SOAPN.demos@WMQSOAP.DEMO.QM
   &connectionFactory=(connectQueueManager(WMQSOAP.DEMO.QM))
   &initialContextFactory=com.ibm.mq.jms.Nojndi
   &targetService=StockQuoteDotNet.asmx
   &replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE
   ```

   Figura 13. URI de ejemplo en el cliente .NET generado para llamar al servicio .NET

   b.
   ```
   jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM
   &connectionFactory=(connectQueueManager(WMQSOAP.DEMO.QM))
   &initialContextFactory=com.ibm.mq.jms.Nojndi
   &targetService=soap.server.StockQuoteAxis.java
   &replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE
   ```

   Figura 14. URI de ejemplo en el cliente .NET generado para llamar al servicio Axis 1
9. Compila los proxies Java.
10. Crea una cola de WebSphere MQ , *requestQueue* para contener las solicitudes para el servicio. El nombre de cola predeterminado tiene el formato SOAPJ.*directory*, o puede especificar *requestQueue* en la opción de URI -u .
11. Crea archivos de mandatos y de scripts de shell para iniciar los escuchas SOAP de WebSphere MQ que procesan la cola de solicitudes.
12. Si se ha utilizado la opción -tmq , el programa de utilidad de despliegue crea definiciones de WebSphere MQ para desencadenar automáticamente los procesos de escucha SOAP de WebSphere MQ .

    - El programa de utilidad de despliegue utiliza el atributo APPLICID del mandato **runmqsc** DEFINE PROCESS para contener un mandato para iniciar el escucha. El mandato tiene el nombre del directorio de despliegue incorporado. El campo APPLICID tiene una longitud máxima de 256, lo que limita la longitud máxima del directorio de despliegue. El límite de directorio para los servicios Java es el siguiente:
      – Sistemas UNIX and Linux : 218
      – Windows: 197 menos la longitud del nombre de cola de solicitudes.

      Para los servicios .NET, el límite de directorio es el siguiente:
      – Windows: 209 menos la longitud del nombre de servicio, menos la extensión .asmx .
    - El programa de utilidad de despliegue comprueba si se ha superado el límite para APPLICID . Si se supera el límite, el programa de utilidad no intenta definir el proceso desencadenante. Muestra un mensaje de error y el proceso de despliegue falla sin realizar ningún paso de despliegue.

    Los ejemplos siguientes muestran los mandatos de configuración e inicio generados por el programa de utilidad de despliegue para iniciar un escucha SOAP de WebSphere MQ .

    ```
    DEFINE PROCESS(requestQueue) APPLICID(applicIDStr) REPLACE
    ALTER  QLOCAL (requestQueue) TRIGTYPE(FIRST) TRIGGER
           PROCESS(requestQueue) INITQ(initQueueName) TRIGMPRI(0)
    ```

    Figura 15. Mandatos de configuración de WebSphere MQ para desencadenar un escucha SOAP.

```
applicIDStr = start "Java WMQSoapListener -requestQueue"
                    /min .\generated\server\startWMQJListener.cmd;
```

Figura 16. Inicio del escucha SOAP de Axis en Windows

```
applicIDStr = start "WMQAsmxListener -className\
                    /min .\generated\server\startWMQNListener.cmd;
```

Figura 17. Inicio del escucha .NET SOAP en Windows

```
applicIDStr = xterm -iconic -T \"Java WMQSoapListener_requestQueue\"
                   -e ./generated/server/startWMQJListener.sh & #
```

Figura 18. Inicio del escucha SOAP de Axis en sistemas UNIX and Linux

amqwRegisterdotNet: registrar transporte de IBM WebSphere MQ para SOAP a .NET

Registre el transporte IBM WebSphere MQ para SOAP a la memoria caché de ensamblaje global en .NET.

Finalidad

amqwRegisterdotNet registra el emisor SOAP, el escucha SOAP y el procesador WSDL de WebSphere MQ con .NET Framework 1 o 2.

Sintaxis

Descripción

amqwRegisterdotNet se ejecuta automáticamente durante la instalación. No es necesario ejecutarlo de nuevo si .NET Framework que está utilizando se ha instalado antes del transporte WebSphere MQ para SOAP. Puede ejecutarlo tantas veces como desee. Utilícelo para volver a registrar el transporte WebSphere MQ para SOAP con distintas versiones de .NET Framework.

Nota: En Windows 2003 Server, también debe ejecutar el programa de utilidad **aspnet_regiis** , incluso si no está desplegando en Internet Information Server (IIS). La ubicación del programa de utilidad `aspnet_regiis.exe` puede variar con distintas versiones de Microsoft .NET Framework, pero normalmente se encuentra en: `%SystemRoot%/Microsoft.NET/Framework/version number/aspnet_regiis`. Si hay varias versiones instaladas, utilice **aspnet_regiis** para la versión de .NET Framework que está utilizando.

Licencia de software de Apache

Licencia de Apache Versión 2.0, enero de 2004 http://www.apache.org/licenses/

http://www.apache.org/licenses/

Licencia de Apache
Versión 2.0, enero de 2004
http://www.apache.org/licenses/

TÉRMINOS Y CONDICIONES DE USO, REPRODUCCIÓN Y DISTRIBUCIÓN

1. Definiciones.

"Licencia": los términos y condiciones de uso, reproducción,
y distribución tal como se define en las secciones 1 a 9 de este documento.

"Licenciante": el propietario o entidad de copyright autorizado por
el propietario del copyright que otorga la licencia.

"Entidad jurídica": la unión de la entidad actora y todos los
otras entidades que controlan, están controladas por, o están bajo
control con esa entidad. A efectos de la presente definición,
"control": i) la potencia, directa o indirecta, que provoca la
dirección o gestión de dicha entidad, ya sea por contrato o
de lo contrario, o (ii) la propiedad del cincuenta por ciento (50%) o más de la
acciones en circulación, o iii) titularidad real de dicha entidad.

"Usted" (o "Su") se refiere a una persona física o jurídica
ejerciendo los permisos otorgados por esta licencia.

Forma "fuente": la forma preferida para realizar modificaciones,
incluyendo pero no limitado a código fuente de software, documentación
y los archivos de configuración.

Forma de "objeto": cualquier forma resultante de una
transformación o traducción de un formulario de origen, incluyendo pero
no limitado a código de objeto compilado, documentación generada,
y conversiones a otros tipos de medios.

Por "trabajo" se entenderá el trabajo de autoría, ya sea en origen o
Formulario de objeto, disponible bajo la licencia, según lo indicado por un
aviso de copyright que se incluye o se adjunta a la obra
(en el apéndice figura un ejemplo).

"Trabajos derivados": cualquier trabajo, ya sea en origen u objeto.
forma, que se basa en (o se deriva de) el trabajo y para el que el
revisiones editoriales, anotaciones, elaboraciones u otras modificaciones
representar, en su conjunto, una obra original de autoría. A los efectos
de esta Licencia, los Trabajos Derivados no incluirán los trabajos que permanezcan
separable de, o simplemente enlace (o enlace por nombre) a las interfaces de,
los trabajos y trabajos derivados de los mismos.

"Contribución": cualquier obra de autor, en particular:
la versión original de la Obra y cualquier modificación o adición
a ese Trabajo o Trabajos Derivados del mismo, que es intencionadamente
enviado al Licenciante para su inclusión en el trabajo por el propietario del copyright
o por una persona física o jurídica autorizada a presentar en nombre de
el propietario del copyright. A efectos de la presente definición, "presentado"
Cualquier forma de comunicación electrónica, verbal o escrita enviada
al Licenciante o a sus representantes, incluidos, a título enunciativo y no limitativo,
comunicación en listas electrónicas de correo, sistemas de control de código fuente,
y sistemas de seguimiento de problemas gestionados por, o en nombre de,
Licenciante con el fin de discutir y mejorar el trabajo, pero
excluyendo la comunicación que está marcada de forma visible o de otro modo
designado por escrito por el propietario de los derechos de autor como "No una contribución".

"Colaborador": el Licenciante y cualquier persona física o jurídica
en nombre de los cuales el Licenciante ha recibido una contribución y
posteriormente incorporados en el trabajo.

2. Concesión de licencia de copyright. Sujeto a los términos y condiciones de
esta Licencia, cada Colaborador por el presente le otorga un perpetuo,
mundial, no exclusivo, gratuito, libre de regalías, irrevocable
licencia de copyright para reproducir, preparar Trabajos Derivados de,

mostrar públicamente, realizar públicamente, sublicenciar y distribuir el Trabajos y dichos trabajos derivados en forma de fuente u objeto.

3. Concesión de licencia de patente. Sujeto a los términos y condiciones de esta Licencia, cada Colaborador por el presente le otorga un perpetuo, mundial, no exclusivo, gratuito, libre de regalías, irrevocable (excepto como se indica en esta sección) licencia de patente para hacer, han hecho, utilizar, ofrecer para vender, vender, importar y, de lo contrario, transferir el trabajo, donde dicha licencia se aplica únicamente a las solicitudes de patente licenciables por dicho Colaborador que sean necesariamente infringidos por su Contribución (s) sola (s) o por combinación de su (s) contribución (s) con el trabajo al que se han enviado dichas contribuciones. Si instituir un litigio de patente contra cualquier entidad (incluida una reclamación o reconvención en una demanda) alegando que el Trabajo o una contribución incorporada en el trabajo constituye una contribución directa o infracción de patente contributiva, entonces cualquier licencia de patente otorgado al Cliente en virtud de esta Licencia para que el Trabajo termine a partir de la fecha en que se presente dicho litigio.

4. Redistribución. Usted puede reproducir y distribuir copias de la Trabajos o trabajos derivados de los mismos en cualquier soporte, con o sin modificaciones, y en el formulario de origen u objeto, siempre que cumplen las condiciones siguientes:

(a) Usted debe dar cualquier otro receptor de la Obra o Trabajos Derivados una copia de esta Licencia; y

(b) Usted debe hacer que cualquier archivo modificado lleve avisos prominentes indicando que ha cambiado los archivos; y

(c) Usted debe retener, en la forma de Fuente de cualquier Trabajos Derivados que usted distribuye, todos los derechos de autor, patente, marca registrada, y avisos de atribución de la forma Fuente del Trabajo, excluyendo los avisos que no pertenecen a ninguna parte de los trabajos derivados; y

(d) Si el trabajo incluye un archivo de texto "AVISO" como parte de su , a continuación, cualquier trabajo derivado que distribuya debe incluir una copia legible de los avisos de atribución contenidos dentro de dicho archivo NOTICE, excluyendo los avisos que no pertenecen a cualquier parte de los trabajos derivados, en al menos uno de los siguientes lugares: dentro de un archivo de texto NOTICE distribuido como parte de los Trabajos Derivados; dentro de la forma Fuente o documentación, si se proporciona junto con los trabajos derivados; o, dentro de una pantalla generada por los trabajos derivados, si y donde normalmente aparecen dichos avisos de terceros. El contenido del archivo NOTICE son sólo para fines informativos y no modifique la licencia. Puede añadir su propia atribución avisos dentro de Trabajos Derivados que distribuya, junto con o como adenda al texto de la NOTIFICACIÓN de la Obra, siempre que que tales avisos de atribución adicionales no pueden interpretarse como modificar la licencia.

Puede añadir su propia declaración de copyright a sus modificaciones y puede proporcionar términos y condiciones de licencia adicionales o diferentes para uso, reproducción o distribución de sus modificaciones, o para cualquiera de dichos Trabajos Derivados en su conjunto, siempre que su uso, reproducción y distribución de la Obra de otro modo cumple con

las condiciones establecidas en esta licencia.

5. Presentación de contribuciones. A menos que indique explícitamente lo contrario,
cualquier contribución enviada intencionadamente para su inclusión en el trabajo
por usted al Licenciante estará bajo los términos y condiciones de
esta licencia, sin ningún término o condición adicional.
Sin perjuicio de lo anterior, nada en este documento reemplazará o modificará
los términos de cualquier acuerdo de licencia independiente que pueda haber ejecutado
con el Licenciante en relación con tales Contribuciones.

6. Marcas registradas. Esta licencia no otorga permiso para utilizar el comercio
nombres, marcas registradas, marcas de servicio o nombres de productos del Licenciante,
excepto cuando sea necesario para un uso razonable y habitual en la descripción de la
origen de la Obra y reproducción del contenido del fichero NOTICE.

7. Declaración de limitación de responsabilidad de garantía. A menos que lo exija la legislación aplicable o
acordada por escrito, el Licenciante proporciona el Trabajo (y cada
Contribuidor proporciona sus contribuciones) en un "TAL CUAL" BASIS,
SIN GARANTÍAS NI CONDICIONES DE NINGÚN TIPO, ni expresas ni
implícita, incluidas, a título enunciativo y no limitativo, las garantías o condiciones
de TÍTULO, NO INFRACCIÓN, COMERCIABILIDAD o APTITUD PARA UN
UN PROPÓSITO DETERMINADO. Usted es el único responsable de determinar el
idoneidad de utilizar o redistribuir el trabajo y asumir cualquier
riesgos asociados con el ejercicio de los permisos bajo esta Licencia.

8. LIMITACIÓN DE RESPONSABILIDAD En ningún caso y bajo ninguna teoría jurídica,
ya sea en agravio (incluyendo negligencia), contrato, o de otra manera,
a menos que lo exija la legislación aplicable (por ejemplo, deliberada y groseramente
actos negligentes) o acordados por escrito, será cualquier contribuyente
responsabilidad ante Usted por daños, incluyendo cualquier directo, indirecto, especial,
daños accidentales, o consecuentes de cualquier carácter que surja como un
resultado de esta licencia o fuera del uso o de la incapacidad de utilizar el
Trabajo (incluyendo pero no limitado a daños por pérdida de buena voluntad,
parada de trabajo, anomalía o mal funcionamiento del sistema, o cualquiera y todos
otros daños comerciales o pérdidas), incluso si tal Colaborador
ha sido informado de la posibilidad de tales daños.

9. Aceptación de garantía o responsabilidad adicional. Al redistribuir
el Trabajo o Trabajos Derivados del mismo, Usted puede optar por ofrecer,
y cobrar una tarifa por, aceptación de soporte, garantía, indemnización,
u otras obligaciones de responsabilidad y/o derechos coherentes con este
Licencia. Sin embargo, al aceptar tales obligaciones, Usted sólo puede actuar
en su propio nombre y bajo su exclusiva responsabilidad, no en nombre
de cualquier otro Colaborador, y solo si el Cliente acepta indemnizar,
defender, y mantener a cada Colaborador inofensivo para cualquier responsabilidad
incurridas por, o reclamaciones declaradas contra, dicho Colaborador por razón
de su aceptación de cualquier garantía o responsabilidad adicional.

FIN DE TÉRMINOS Y CONDICIONES

APÉNDICE: Cómo aplicar la licencia de Apache a su trabajo.

Para aplicar la licencia de Apache a su trabajo, adjunte lo siguiente
aviso de placa modelo, con los campos entre corchetes "[]"
con su propia información de identificación. (No incluir
los corchetes!) El texto debe incluirse en el
sintaxis de comentario para el formato de archivo. También se recomienda que un

nombre de archivo o clase y descripción de la finalidad que debe incluirse en el
la misma "página impresa" que el aviso de copyright para facilitar
identificación dentro de archivos de terceros.

Copyright [aaaa] [nombre del propietario del copyright]

Con licencia bajo Apache License, Versión 2.0 (la "Licencia");
no puede utilizar este archivo excepto en conformidad con la licencia.
Usted puede obtener una copia de la licencia en

http://www.apache.org/licenses/LICENSE-2.0

A menos que lo exija la legislación aplicable o se acuerde por escrito, el software
distribuido bajo la licencia se distribuye en un "TAL CUAL" BASIS,
SIN GARANTÍAS NI CONDICIONES DE NINGÚN TIPO, ya sean expresas o implícitas.
Consulte la licencia para obtener los permisos de control de idioma específicos y
limitaciones bajo la licencia.

Valores SOAP de MQMD

El emisor SOAP de IBM WebSphere MQ y el escucha SOAP de IBM WebSphere MQ crean un descriptor de mensaje (**MQMD**). El tema describe los campos que debe establecer en MQMD si crea su propio emisor o escucha SOAP.

Finalidad

Los valores establecidos en **MQMD** controlan el intercambio de mensajes entre el remitente SOAP de IBM WebSphere MQ , el escucha SOAP de IBM WebSphere MQ y el programa cliente SOAP. Si crea su propio emisor o escucha SOAP, siga las reglas de Tabla 584 en la página 969.

Descripción

Tabla 584 en la página 969 describe cómo se establecen los campos **MQMD** mediante el emisor SOAP de IBM WebSphere MQ y el escucha SOAP de IBM WebSphere MQ . Si escribe su propio remitente o escucha, debe establecer estos campos de acuerdo con las reglas para intercambiar mensajes. El escucha SOAP de IBM WebSphere MQ se ajusta a los protocolos de intercambio de mensajes típicos de IBM WebSphere MQ . Si escribe su propio remitente para trabajar con los escuchas SOAP de IBM WebSphere MQ , puede establecer distintos valores **MQMD** .

En Tabla 584 en la página 969, los valores de la columna Valor se organizan de la forma siguiente:

Solicitud, unidireccional
Valores realizados por el remitente SOAP de IBM WebSphere MQ .

Respuesta, Informe
Valores realizados por el escucha SOAP de IBM WebSphere MQ en respuesta a la solicitud del remitente SOAP de IBM WebSphere MQ .

ALL
Valores realizados por el emisor SOAP de IBM WebSphere MQ y el escucha SOAP de IBM WebSphere MQ .

Remitente personalizado
Puede escribir su propio remitente. Normalmente, un remitente personalizado altera temporalmente las opciones de informe estándar.

Tabla 584. Valores SOAP de MQMD

Nombre de campo	Valor	Valores
StrucId	**ALL** MQMD_STRUC_ID	'MD¬¬'[1]

Tabla 584. Valores SOAP de MQMD (continuación)		
Nombre de campo	**Valor**	**Valores**
Version	**ALL** MQMD_VERSION_2	2
Report	**ALL** MQRO_NONE + MQRO_NEW_MSG_ID + MQRO_COPY_MSG_ID_TO_CORREL_ID + MQRO_EXCEPTION + MQRO_EXPIRY + MQRO_DISCARD **Remitente personalizado** Consulte “Opciones de informe personalizadas” en la página 974.	52428800
MsgType	**Solicitud** MQMT_REQUEST **Respuesta** MQMT_REPLY **Informe** MQMT_REPORT **Unidireccional** MQMT_DATAGRAM	**MQMT_REQUEST** 1 **MQMT_REPLY** 2 **MQMT_REPORT** 4 **MQMT_DATAGRAM** 8
Expiry	**Solicitud, unidireccional** Especificado por la opción Caducidad en el URI. El valor predeterminado es MQEI_UNLIMITED. **Respuesta** Valor de Caducidad en el mensaje de solicitud **Informe** MQEI_UNLIMITED	**MQEI_UNLIMITED** -1

Tabla 584. Valores SOAP de MQMD (continuación)

Nombre de campo	Valor	Valores
Feedback	**Solicitud, respuesta, unidireccional** MQFB_NONE. **Informe** • Generado por el gestor de colas-valor establecido de acuerdo con las reglas normales. • Generado por el escucha SOAP de IBM WebSphere MQ : **MQRC_BACKOUT_THRESHOLD_REACHED** Se ha superado el umbral de restitución para varios intentos. **MQRCCF_MD_FORMAT_ERROR** No se reconoce que el mensaje tenga una cabecera MQRFH2 . **MQRC_RFH_PARM_MISSING** Falta un parámetro necesario, por ejemplo, SoapAction, en **MQRFH2** . **MQRC_RFH_FORMAT_ERROR** Una comprobación de integridad básica de **MQRFH2** ha fallado, por ejemplo, las longitudes internas están dañadas. **MQRC_RFH_ERROR** **MQRFH2** ha pasado una comprobación de integridad, pero el cuerpo del mensaje no está establecido en MQFMT_NONE.	**MQFB_NONE** 0 **MQRC_BACKOUT_THRESHOLD_REACHED** 2362 **MQRCCF_MD_FORMAT_ERROR** 3023 **MQRC_RFH_PARM_MISSING** 2339 **MQRC_RFH_FORMAT_ERROR** 2421 **MQRC_RFH_ERROR** 2334
Encoding	**ALL** MQENC_NATIVE	Depende del entorno
CodedCharSetId	**ALL** Establézcalo en UTF-8	1208
Format	**Solicitud, respuesta, unidireccional** MQFMT_RF_HEADER_2 **Informe** **Informes de gestor de colas** Sigue las reglas de IBM WebSphere MQ **Informes de escucha SOAP de IBM WebSphere MQ** Formato del mensaje de solicitud original.	**MQFMT_RF_HEADER_2** "MQRFH2 "

Tabla 584. Valores SOAP de MQMD (continuación)		
Nombre de campo	**Valor**	**Valores**
Priority	**Solicitud, unidireccional** Especificado por la opción Prioridad en el URI. El valor predeterminado es MQPRI_PRIORITY_AS_Q_DEF. **Respuesta, Informe** Valor de Prioridad en el mensaje de solicitud.	**MQPRI_PRIORITY_AS_Q_DEF** -1
Persistence	**Solicitud, unidireccional** MQPER_PERSISTENCE_AS_Q_DEF. **Respuesta, Informe** Valor de Persistence en el mensaje de solicitud.	**MQPER_PERSISTENCE_AS_Q_DEF** 2
MsgId	**Solicitud, unidireccional** Generado por el gestor de colas. **Respuesta, Informe** Se generan los IBM WebSphere MQ conjuntos de remitentes SOAP MQRO_NEW_MSG_ID y *MsgId*	**Generado** Valor exclusivo generado por el gestor de colas
CorrelId	**Solicitud, unidireccional, Informe** MQCI_NONE **Respuesta, Informe** El emisor SOAP de IBM WebSphere MQ establece MQRO_COPY_MSG_ID_TO_CORREL_ID y el escucha copia *MsgId* del mensaje de solicitud.	**MQCI_NONE** 0
BackoutCount	**ALL** No se utiliza	0
ReplyToQ	**Solicitud** Especificado por la opción replyDestination en URI. El valor predeterminado es SYSTEM.SOAP.RESPONSE.QUEUE. **Respuesta, unidireccional, Informe** Se ha dejado en blanco	
ReplyToQMgr	**ALL** Campo dejado en blanco	Generado por el gestor de colas; consulte Cola de respuesta y gestor de colas .

Tabla 584. Valores SOAP de MQMD (continuación)		
Nombre de campo	**Valor**	**Valores**
UserIdentifier	**Solicitud, unidireccional, Informe** Se ha dejado en blanco **Respuesta** Depende de la opción -i *passContext* proporcionada al escucha y de la autorización con la que se ejecuta el escucha.	**Solicitud, unidireccional, Informe** Generado por el gestor de colas; consulte “UserIdentifier (MQCHAR12)” en la página 439 **Respuesta** *variable*
AccountingToken	**ALL** MQACT_NONE	**MQACT_NONE** Serie nula o espacios en blanco Establecido por el gestor de colas; consulte “AccountingToken (MQBYTE32)” en la página 396
ApplIdentityData	**ALL** Ninguna	Serie nula o blancos[2]
PutApplType	**ALL** MQAT_NO_CONTEXT	**MQAT_NO_CONTEXT** 0 Valor generado por el gestor de colas; consulte “PutApplType (MQLONG)” en la página 425.
PutApplName	**ALL** Ninguna	Valor generado por el gestor de colas; consulte “PutApplNombre (MQCHAR28)” en la página 423.
PutDate	**ALL** Ninguna	Valor generado por el gestor de colas; consulte “PutDate (MQCHAR8)” en la página 426.
PutTime	**ALL** Ninguna	Valor generado por el gestor de colas; consulte “PutTime (MQCHAR8)” en la página 427.
ApplOriginData	**ALL** Ninguna	Serie nula o blancos[2]
GroupId	**Solicitud, unidireccional, Informe** MQGI_NONE **Respuesta** El campo se copia del mensaje de solicitud	Nulos

Tabla 584. Valores SOAP de MQMD (continuación)		
Nombre de campo	**Valor**	**Valores**
MsgSeqNumber	**Solicitud, unidireccional, Informe** No se utiliza **Respuesta** El campo se copia del mensaje de solicitud	Generado por el gestor de colas; consulte Orden físico en una cola.
Offset	**Solicitud, unidireccional, Informe** No se utiliza **Respuesta** El campo se copia del mensaje de solicitud	0
MsgFlags	**Solicitud, unidireccional, Informe** MQMF_NONE **Respuesta** El campo se copia del mensaje de solicitud	**MQMF_NONE** 0 Consulte "MsgFlags (MQLONG)" en la página 413.
OriginalLength	**Solicitud, unidireccional, respuesta** MQOL_UNDEFINED **Informe** Longitud del mensaje de solicitud original	**MQOL_UNDEFINED** -1
Notas: 1. El símbolo ¬ representa un único carácter en blanco. 2. El valor Serie nula o espacios en blanco indica la serie nula en C y los caracteres en blanco en otros lenguajes de programación.		

Opciones de informe personalizadas

Puede escribir su propio remitente SOAP y utilizarlo con los escuchas proporcionados. Normalmente, puede escribir un remitente para cambiar la elección de las opciones de informe. Los escuchas SOAP de IBM WebSphere MQ dan soporte a la mayoría de las combinaciones de opciones de informe, tal como se describe en las listas siguientes.

- Opciones de informe soportadas por escuchas SOAP de IBM WebSphere MQ :
  - MQRO_EXCEPTION
  - MQRO_EXCEPTION_WITH_DATA
  - MQRO_EXCEPTION_WITH_FULL_DATA
  - MQRO_DEAD_LETTER_Q
  - MQRO_DISCARD_MSG
  - MQRO_NONE
  - MQRO_NEW_MSG_ID
  - MQRO_PASS_MSG_ID
  - MQRO_COPY_MSG_ID_TO_CORREL_ID
  - MQRO_PASS_CORREL_ID
- Opciones de informe soportadas por el gestor de colas:

  - MQRO_COA
  - MQRO_COA_WITH_DATA
  - MQRO_COA_WITH_FULL_DATA
  - MQRO_COD
  - MQRO_COD_WITH_DATA
  - MQRO_COD_WITH_FULL_DATA
  - MQRO_EXPIRATION
  - MQRO_EXPIRATION_WITH_DATA
  - MQRO_EXPIRATION_WITH_FULL_DATA
- Las siguientes opciones de informe no están soportadas por los escuchas SOAP de IBM WebSphere MQ .
  - MQRO_PAN
  - MQRO_NAN

El comportamiento de los escuchas SOAP de IBM WebSphere MQ en respuesta a combinaciones de MQRO_EXCEPTION_* y MQRO_DISCARD se describe en Tabla 585 en la página 975.

La notación MQRO_EXCEPTION_* indica el uso de MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA o MQRO_EXCEPTION_WITH_FULL_DATA.

Tabla 585. Comportamiento de escucha resultante de los valores MQRO_EXCEPTION_* y MQRO_DISCARD

	MQRO_DISCARD habilitado	**MQRO_DISCARD No habilitado**
MQRO_EXCEPTION_* habilitado	Comportamiento predeterminado. Los mensajes de informe se generan automáticamente si es necesario y se descarta la solicitud original. Si no se ha podido devolver un mensaje de informe a la cola de respuestas, el mensaje de informe se envía a la cola de mensajes no entregados.	Los mensajes de informe se generan automáticamente si es necesario y el mensaje original se envía a la cola de mensajes no entregados. Si el mensaje de informe no se ha podido devolver a la cola de respuestas, también se envía a la cola de mensajes no entregados. En este caso, por lo tanto, hay dos entradas de cola de mensajes no entregados para la solicitud fallida.
MQRO_EXCEPTION_* No habilitado	Los mensajes de informe no se generan automáticamente cuando no se reconoce el formato de entrada o cuando se supera el número de intentos de restitución. El mensaje no se envía a la cola de mensajes no entregados. No se devuelve ninguna notificación que el cliente pueda inspeccionar y se pierde el mensaje de solicitud original.	Los mensajes de informe no se generan automáticamente cuando no se reconoce el formato de entrada o cuando se supera el número de intentos de restitución. Sin embargo, el mensaje de solicitud original se graba en la cola de mensajes no entregados cuando de lo contrario se habría generado un informe.

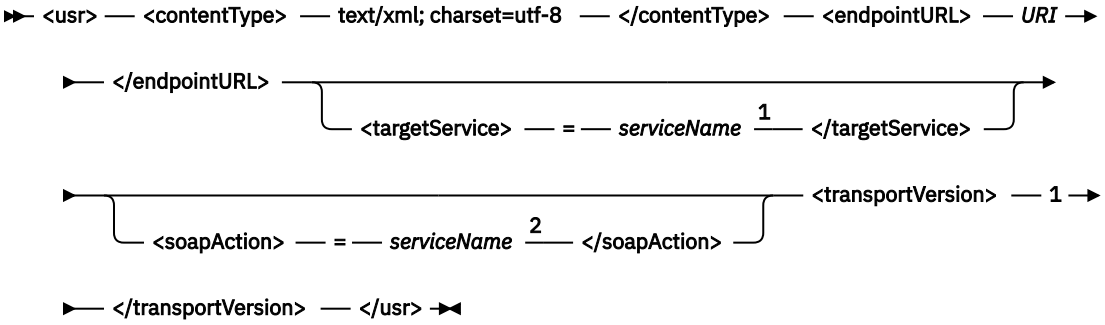
Valores SOAP de MQRFH2

Los remitentes y escuchas SOAP de IBM WebSphere MQ crean o esperan recibir un MQRFH2 con los valores siguientes.

Finalidad

Los remitentes SOAP de WebSphere MQ añaden propiedades a la carpeta <usr> creada por JMS de WebSphere MQ . Las propiedades contienen información necesaria para el contenedor SOAP en el entorno de destino. "Sintaxis de propiedad" en la página 976 describe la sintaxis de las propiedades cuando se añaden a un MQRFH2. Para ver la descripción de una cabecera MQRFH2 , consulte MQRFH2 -Cabecera de reglas y formateo 2.

Sintaxis de propiedad



Notas:

[1] targetService es necesario para .NET Framework 1 o 2 y no se utiliza en Axis 1.4.
[2] soapAction es opcional para .NET Framework 1 o 2 y no se utiliza en Axis 1.4.

Parámetros

contentType
contentType siempre contiene la serie `text/xml; charset=utf-8`.

endpointURL
Consulte "Sintaxis de URI y parámetros para el despliegue de servicios web" en la página 993.

targetService
[6]En Axis, *serviceName* es el nombre completo de un servicio de Java , por ejemplo: `targetService=javaDemos.service.StockQuoteAxis`. Si no se especifica targetService , se carga un servicio utilizando el mecanismo Axis predeterminado.

[7]En .NET, *serviceName* es el nombre de un servicio .NET ubicado en el directorio de despliegue, por ejemplo: `targetService=mySservice.asmx`. En el entorno .NET, el parámetro targetService hace posible que un único escucha SOAP de WebSphere MQ pueda procesar solicitudes de varios servicios. Estos servicios deben desplegarse desde el mismo directorio.

soapAction

transportVersion
transportVersion siempre se establece en 1.

Ejemplo

El ejemplo muestra un MQRFH2 y el siguiente mensaje SOAP. Las longitudes de las carpetas se muestran en decimal.

Nota: & en el URI se codifica como &

```
52464820 00000002 000002B0 00000001 RFH 0002 1208 0001
000004B8 20202020 20202020 00000000 1208 0000
000004B8                           1208
32  <mcd>
        <Msd>jms_bytes</Msd>
```

[6] Sólo servicio Java
[7] Sólo servicio .NET

```
    </mcd>?
208 <jms>
        <Dst>queue://queue://SOAPJ.demos</Dst>
        <Rto>queue://WMQSOAP.DEMO.QM/SYSTEM.SOAP.RESPONSE.QUEUE</Rto>
        <Tms>1157388516465</Tms>
        <Cid>ID:000000000000000000000000000000000000000000000000</Cid>
        <Dlv>1</Dlv>
    </jms>
400 <usr>
        <contentType>text/xml; charset=utf-8</contentType>
        <transportVersion>1</transportVersion>
        <endpointURL>
            jms:/queue?destination=SOAPJ.demos@WMQSOAP.DEMO.QM
            &amp;connectionFactory=connectQueueManager(WMQSOAP.DEMO.QM)
                clientConnection(localhost%25289414%2529)
                clientChannel(TESTCHANNEL)
            &amp;replyDestination=SYSTEM.SOAP.RESPONSE.QUEUE
            &amp;initialContextFactory=com.ibm.mq.jms.Nojndi
        </endpointURL>
    </usr>
<?xml version="1.0" encoding="UTF-8"?>
    <soapenv:Envelope
            xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
            xmlns:xsd="https://www.w3.org/2001/XMLSchema"
            xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
        <soapenv:Body>
            <ns1:getQuote
                    soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    xmlns:ns1="soap.server.StockQuoteAxis_Wmq">
                <in0 xsi:type="xsd:string">XXX</in0>
            </ns1:getQuote>
        </soapenv:Body>
    </soapenv:Envelope>
```

runivt: prueba de verificación de instalación de WebSphere MQ para SOAP

Se proporciona una suite de pruebas de verificación de instalación (IVT) con el transporte de IBM WebSphere MQ para SOAP. **runivt** ejecuta una serie de aplicaciones de demostración y garantiza que el entorno se ha configurado correctamente después de la instalación.

Finalidad

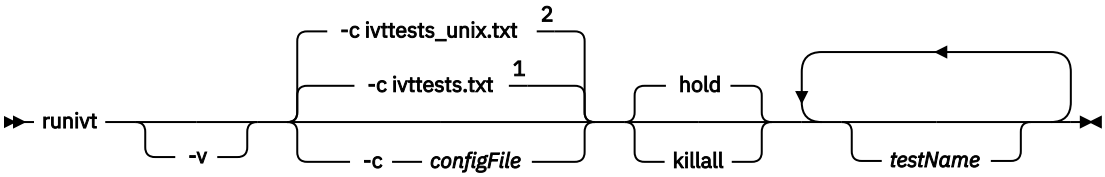
El mandato **runivt** utiliza los programas de ejemplo proporcionados con el transporte de WebSphere MQ para SOAP para enviar solicitudes de servicio web de clientes a servicios. Ejecuta pruebas para Axis 1.4, .NET Framework 1 y .NET Framework 2. Las pruebas se configuran en un archivo de script de prueba. El archivo de script de prueba predeterminado para Windows ejecuta una combinación de pruebas entre clientes y servicios Java y .NET.

Descripción

runivt debe ejecutarse desde su propio directorio.

El mandato inicia escuchas en una ventana de mandatos diferente. Por este motivo, debe ejecutar el mandato desde una sesión del sistema X Window en sistemas UNIX and Linux .

runivt syntax



Notas:

[1] Default on Windows

[2] Default on UNIX and Linux systems

Parámetros `runivt`

`-v`
Modalidad detallada. Escriba mensajes de error más completos en la consola.

`-c` ***configFile***
Un archivo de configuración que define las pruebas que se van a ejecutar. El archivo de configuración predeterminado que se proporciona con los sistemas Windows, UNIX o Linux se utiliza de forma predeterminada.

`hold`
Dejar el escucha en ejecución después de que se completen las pruebas

`killall`
Finalizar el escucha cuando finalicen las pruebas

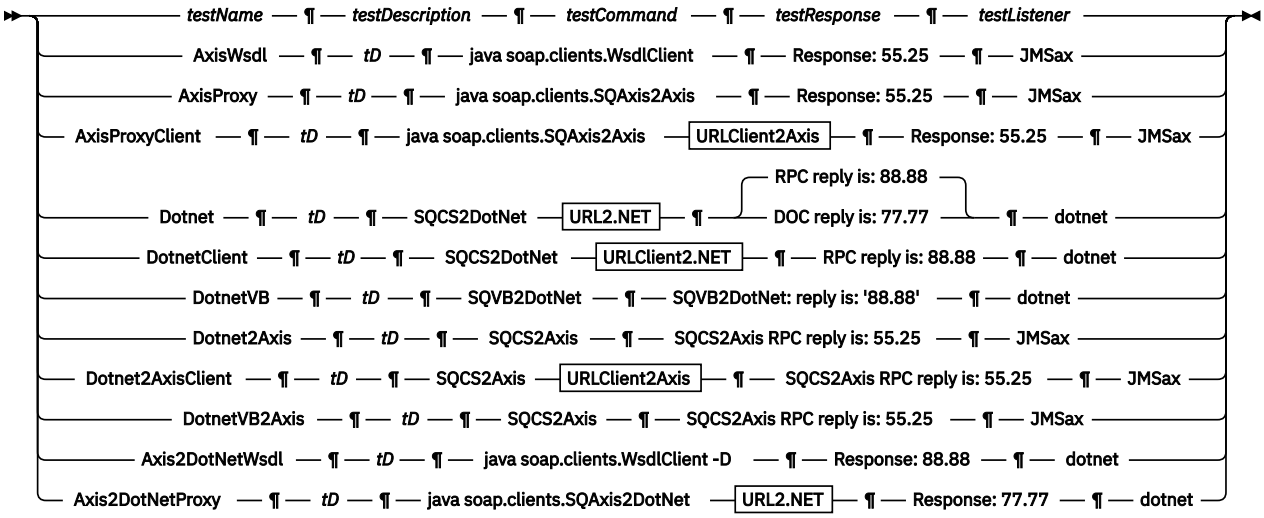
testName
Una lista separada por espacios de las pruebas que se van a ejecutar. Los nombres de prueba se seleccionan en el archivo de configuración. Si no se especifican nombres, se ejecutan todas las pruebas del archivo de configuración.

Configuration file

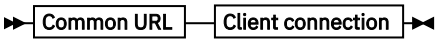
Each configuration file parameter is a separate line of the file. Leave a blank line between each group of parameters.

The parameters in the `ivttests.txt` parameter file are listed.

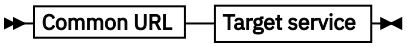
***configFile* syntax**



URLClient2Axis

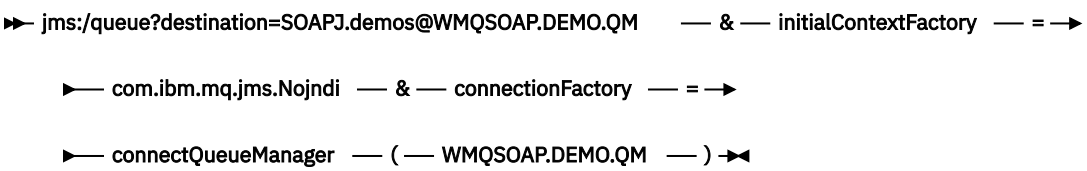


URL2.NET

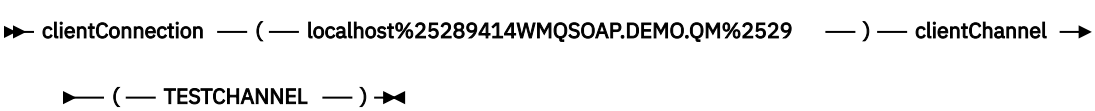


URLClient2.NET

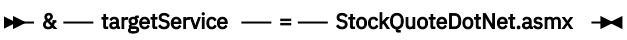
Common URL



Client connection



Target service



Parámetros de *configFile*

testName
El nombre de la prueba. Utilice *testName* en el mandato **runivt**

testDescription
Documentation sobre la prueba

testCommand
El mandato ejecutado por el mandato **runivt** para realizar la solicitud de cliente.

testResponse
La serie de respuesta exacta devuelta por la solicitud de cliente a la consola. Para que la prueba se realice correctamente, *testResponse* debe coincidir con la respuesta real.

testListener
El nombre del escucha SOAP de WebSphere MQ iniciado por **runivt** para procesar la solicitud SOAP. dotnet y JMSax son sinónimos para los escuchas proporcionados, **amqwSOAPNETlistener** y **SimpleJavaListener**.

Ejemplos

```
runivt
```

Figura 19. ejecutar todas las pruebas predeterminadas

```
runivt dotnet
```

Figura 20. ejecutar una prueba específica desde las pruebas predeterminadas

```
runivt -c mytests.txt
```

Figura 21. ejecutar un conjunto de pruebas personalizadas

Información relacionada

Verificación del transporte de WebSphere MQ para SOAP

Servicios web seguros a través del transporte IBM WebSphere MQ para SOAP

Puede proteger los servicios web que utilizan el transporte IBM WebSphere MQ para SOAP de una de dos maneras. Cree un canal SSL entre el cliente y el servidor o utilice la seguridad de servicios web.

SSL y el transporte WebSphere MQ para SOAP

El transporte WebSphere MQ para SOAP proporciona varias opciones SSL que se pueden especificar para su uso con el canal de cliente configurado para ejecutarse en modalidad SSL. Las opciones difieren entre los entornos .NET y Java. Los remitentes y escuchas SOAP de WebSphere MQ sólo procesan las opciones SSL que son aplicables a su entorno concreto. Ignoran las opciones que no son aplicables.

La presencia o ausencia de la opción `sslCipherSpec` para clientes .NET y la opción `sslCipherSuite` para clientes Java determina si se utiliza SSL o no. Si la opción no se especifica en el URI, de forma predeterminada no se utiliza SSL y se ignoran todas las demás opciones SSL. Todas las opciones SSL son opcionales excepto donde se indica.

Para clientes WebSphere MQ , establezca los atributos SSL en la tabla de definición de URI o canal. En el servidor, establezca los atributos utilizando los recursos de WebSphere MQ.

De forma predeterminada, la opción SSL estándar de WebSphere MQ , SSLCAUTH, se establece al habilitar SSL en el canal. Los clientes deben autenticarse a sí mismos antes de que pueda comenzar la comunicación SSL. Si SSLCAUTH no está establecido, las comunicaciones SSL se establecen sin autenticación de cliente.

Para autenticarse a sí mismos, los clientes deben tener un certificado asignado en su repositorio de claves que sea aceptable para el gestor de colas. Para obtener seguridad adicional, los canales de WebSphere MQ se pueden configurar para aceptar sólo certificados de una lista restringida. La lista está restringida comparando el nombre distinguido del certificado con el atributo de nombre de igual del canal.

Si utiliza Java, la primera conexión SSL de un cliente SOAP de WebSphere MQ hace que se arreglen los siguientes parámetros SSL. Los mismos valores se utilizan en conexiones posteriores utilizando el mismo proceso de cliente:

- `sslKeyAlmacén`
- `sslKeyStorePassword`
- `sslTrustAlmacén`
- `sslTrustStorePassword`
- `sslFipsRequired`
- `sslLDAPCRLservers`

El efecto de variar estos parámetros en las conexiones posteriores de este cliente no está definido.

Si utiliza .NET, la primera conexión SSL de un cliente SOAP de WebSphere MQ hace que se arreglen los siguientes parámetros SSL. Los mismos valores se utilizan en conexiones posteriores utilizando el mismo proceso de cliente:

- `SSLKeyRepository`
- `SSLCryptoHardware`
- `sslFipsRequired`
- `sslLDAPCRLservers`

El efecto de variar estos parámetros en las conexiones posteriores de este cliente no está definido. Estos parámetros se restablecen si todas las conexiones SSL pasan a estar inactivas y se realiza una nueva conexión SSL.

Las propiedades siguientes también se pueden especificar como propiedades del sistema:

- `sslKeyAlmacén`
- `sslKeyStorePassword`
- `sslTrustAlmacén`
- `sslTrustStorePassword`

Si se especifican como propiedades del sistema y en el URI, y los valores difieren, el programa de utilidad de despliegue muestra un aviso. Los valores de URI tienen prioridad.

Tareas relacionadas

Especificación de que sólo se utilizan CipherSpecs certificadas por FIPS en el tiempo de ejecución del cliente MQI

Referencia relacionada

Parámetros de fábrica de conexiones SSL en el URI de servicios web de WebSphere MQ

Añada opciones SSL a la lista de opciones de fábrica de conexiones en el URI de servicios web de IBM WebSphere MQ .

Federal Information Processing Standards (FIPS) para UNIX, Linux y Windows

Parámetros de fábrica de conexiones SSL en el URI de servicios web de WebSphere MQ

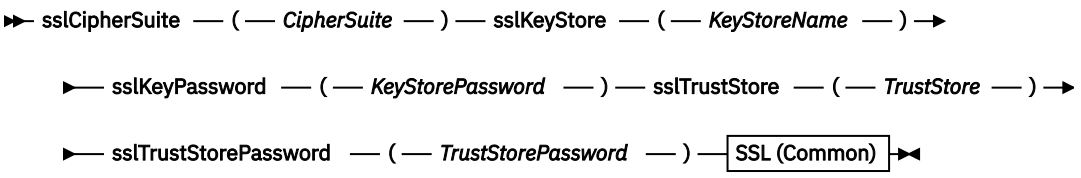
Añada opciones SSL a la lista de opciones de fábrica de conexiones en el URI de servicios web de IBM WebSphere MQ .

Finalidad

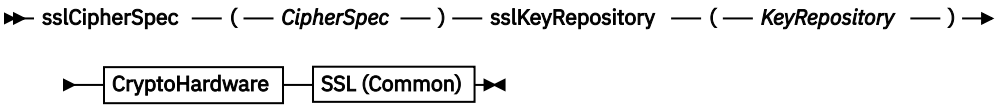
Puede utilizar una conexión segura entre un cliente de servicios web de IBM WebSphere MQ y el gestor de colas que aloja el servicio web. Las opciones SSL controlan cómo se configura SSL en la conexión de canal cliente-servidor MQI de IBM WebSphere MQ .

Syntax diagram

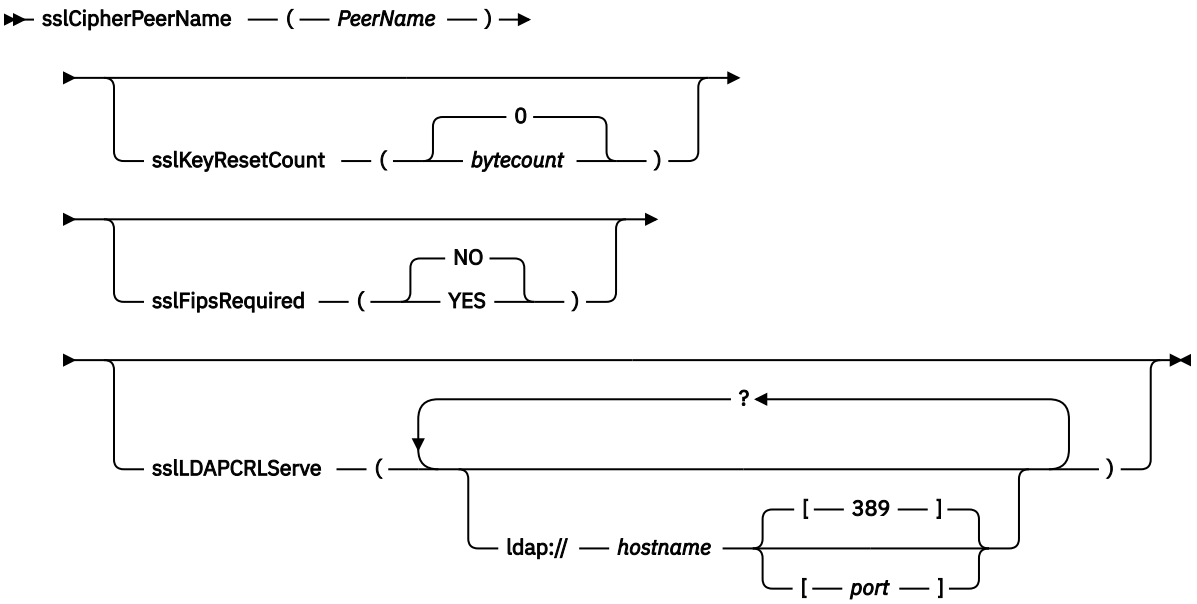
SSL (Java)



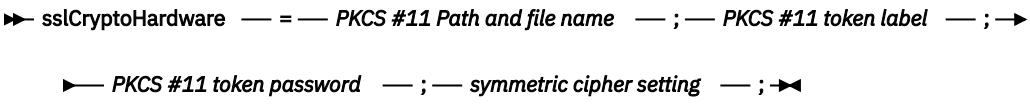
SSL (.NET)



SSL (Common)



CryptoHardware



Parámetros SSL necesarios (común)

sslPeerName(*peerName*)
peerName especifica el sslPeerNombre utilizado en el canal.

Parámetros SSL necesarios (Java)

sslCipherSuite(*CipherSuite*)
CipherSuite especifica la sslCipherSuite utilizada en el canal. La CipherSuite especificada por el cliente debe coincidir con la CipherSuite especificada en el canal de conexión del servidor.

sslKeyStore(*KeyStoreNombre*)
KeyStoreName especifica la sslKeyStoreName utilizada en el canal. El almacén de claves contiene la clave privada del cliente utilizada para autenticar el cliente en el servidor. El almacén de claves es opcional si la conexión SSL está configurada para aceptar conexiones de cliente anónimas.

sslKeyStorePassword(*KeyStoreContraseña*)
KeyStoreContraseña especifica la sslKeyStorePassword utilizada en el canal.

sslTrustStore(*TrustStoreNombre*)
TrustStoreNombre especifica el sslTrustStoreName utilizado en el canal. El almacén de confianza contiene el certificado público del servidor, o su cadena de claves, para autenticar el servidor en el cliente. El almacén de confianza es opcional si se utiliza el certificado raíz de una entidad emisora de certificados para autenticar el servidor. En Java, los certificados raíz se conservan en el almacén de certificados JRE, `cacerts`.

sslTrustStorePassword(*TrustStore*)
TrustStoreContraseña especifica el sslTrustStorePassword utilizado en el canal.

Parámetros SSL necesarios (.NET)

sslCipherSpec(*CipherSpec*)
CipherSpec especifica la sslCipherSpec utilizada en el canal. Si se especifica la opción, se utiliza SSL en el canal de cliente.

sslKeyRepository(*KeyRepository*)
KeyRepository especifica la especificaciónsslCipher utilizada en el canal donde se almacenan las claves SSL y los certificados. *KeyRepository* se especifica en formato raíz, es decir, una vía de acceso completa con el nombre de archivo pero con la extensión de archivo omitida. El efecto de establecer sslKeyRepository es el mismo que establecer el campo `KeyRepository` en la estructura **MQSCO** en una llamada `MQCONNX` .

Parámetros SSL opcionales (.NET)

sslCryptoHardware(*CryptoHardware*)
CryptoHardware especifica el sslCryptoHardware utilizado en el canal. Los valores posibles para este campo, y el efecto de establecerlo, son los mismos que para el campo `CryptoHardware` de la estructura **MQSCO** en un `MQCONNX`.

Parámetros SSL opcionales (comunes)

sslKeyResetCount(*bytecount*)
bytecount especifica el número de bytes pasados a través de un canal SSL antes de que se deba renegociar la clave secreta SSL. Para inhabilitar la renegociación de claves SSL omita el campo o establézcalo en cero. Cero es el único valor soportado en algunos entornos, consulte Renegociación

de la clave secreta en WebSphere MQ classes for Java. El efecto de establecer sslKeyResetCount es el mismo que establecer el campo `KeyResetCount` en la estructura **MQSCO** en una llamada `MQCONNX` .

sslFipsRequired(*fipsCertified*)
fipsCertified especifica si *CipherSpec* o *CipherSuite* debe utilizar la criptografía certificada por FIPS en IBM WebSphere MQ en el canal. El efecto de establecer *fipsCertified* es el mismo que establecer el campo `FipsRequired` de la estructura **MQSCO** en una llamada `MQCONNX` .

sslLDAPCRLServers(*LDAPServerList*)

LDAPServerList especifica una lista de servidores LDAP que se utilizarán para la comprobación de la lista de revocación de certificados.

Para las conexiones de cliente habilitadas para SSL, *LDAPServerList* es una lista de servidores LDAP que se van a utilizar para la comprobación de la lista de revocación de certificados (CRL). El certificado proporcionado por el gestor de colas se compara con uno de los servidores CRL LDAP listados; si se encuentra, la conexión falla. Cada servidor LDAP se intenta a su vez hasta que se establece la conectividad con uno de ellos. Si es imposible conectarse a alguno de los servidores, el certificado se rechaza. Una vez que se ha establecido correctamente una conexión con uno de ellos, el certificado se acepta o se rechaza en función de las CRL presentes en ese servidor LDAP.

Si *LDAPServerList* está en blanco, el certificado que pertenece al gestor de colas no se compara con una lista de revocación de certificados. Se visualiza un mensaje de error si la lista proporcionada de URI de LDAP no es válida. El efecto de establecer este campo es el mismo que el de incluir registros `MQAIR` y acceder a ellos desde una estructura **MQSCO** en un `MQCONNX`.

Tareas relacionadas

Especificación de que sólo se utilizan CipherSpecs certificadas por FIPS en el tiempo de ejecución del cliente MQI

Referencia relacionada

SSL y el transporte WebSphere MQ para SOAP
El transporte WebSphere MQ para SOAP proporciona varias opciones SSL que se pueden especificar para su uso con el canal de cliente configurado para ejecutarse en modalidad SSL. Las opciones difieren entre los entornos .NET y Java. Los remitentes y escuchas SOAP de WebSphere MQ sólo procesan las opciones SSL que son aplicables a su entorno concreto. Ignoran las opciones que no son aplicables.

Federal Information Processing Standards (FIPS) para UNIX, Linux y Windows

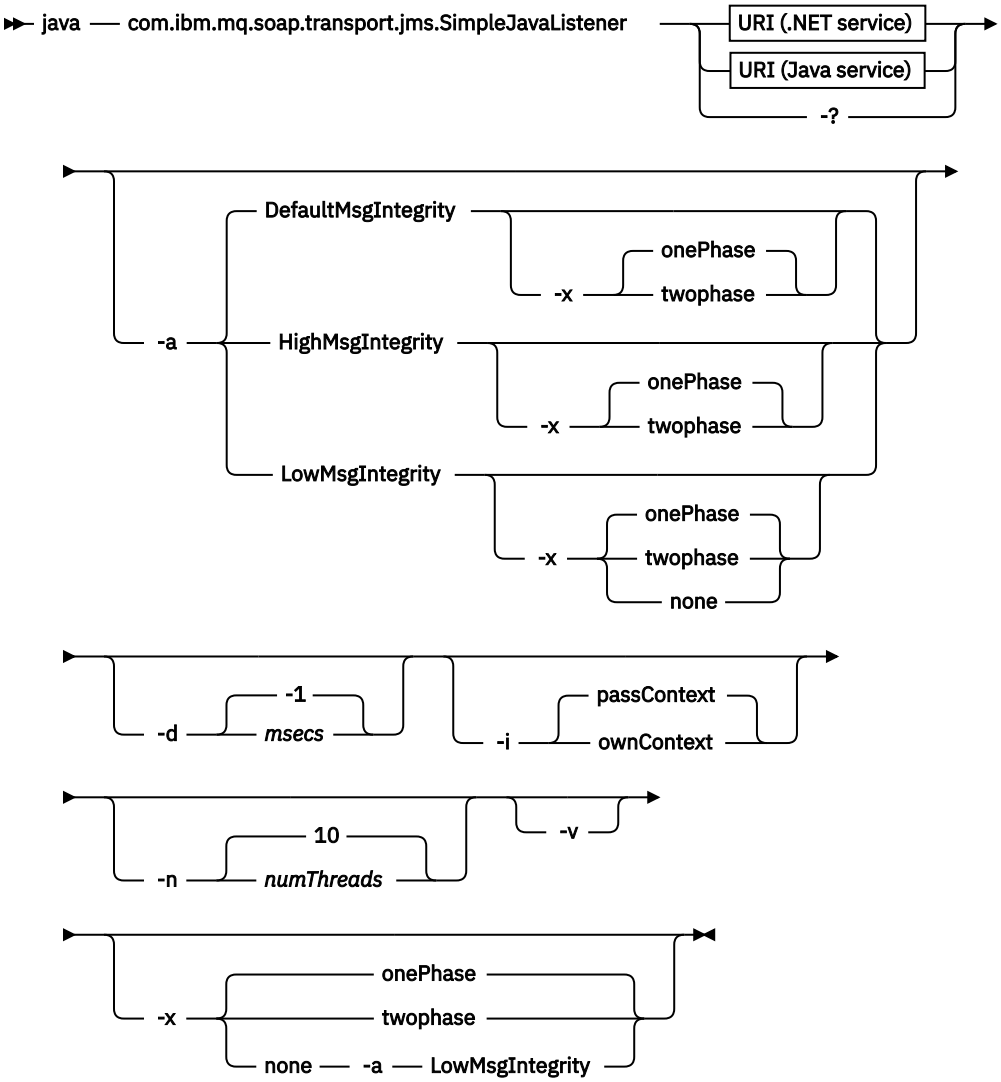
`SimpleJavaListener`: IBM WebSphere MQ Escucha SOAP para Axis 1.4

Sintaxis y parámetros para el escucha SOAP de IBM WebSphere MQ para Axis 1.4.

Finalidad

Inicia el escucha SOAP de IBM WebSphere MQ para Axis 1.4.

Java



Parámetros necesarios

URI ***plataforma***
Consulte “Sintaxis de URI y parámetros para el despliegue de servicios web” en la página 993.

-?
Imprima el texto de ayuda que describe cómo se utiliza el mandato.

Parámetros opcionales

-a ***integrityOption***
integrityOption especifica el comportamiento de los escuchas SOAP de WebSphere MQ si no es posible colocar un mensaje de solicitud fallida en la cola de mensajes no entregados. *integrityOption* puede tomar uno de los valores siguientes:

DefaultMsgIntegrity
Para los mensajes no persistentes, el escucha muestra un mensaje de aviso y continúa ejecutándose con el mensaje original que se descarta. Para los mensajes persistentes, muestra un mensaje de error, restituye el mensaje de solicitud para que permanezca en la cola de solicitudes y salga. DefaultMsgIntegrity se aplica si se omite la opción -a o si no se especifica *integrityOption* .

LowMsgIntegrity
Para los mensajes persistentes y no persistentes, el escucha muestra un aviso y continúa ejecutándose, descartando el mensaje.

HighMsgIntegrity
Para los mensajes persistentes y no persistentes, el escucha muestra un mensaje de error, restituye el mensaje de solicitud para que permanezca en la cola de solicitudes y salga.

El programa de utilidad de despliegue comprueba la compatibilidad de los distintivos -x y -a . Si se especifica -x none , se debe especificar -a LowMsgIntegrity . Si los distintivos son incompatibles, el programa de utilidad de despliegue sale con un mensaje de error y no se han realizado pasos de despliegue.

-d ***msegs***
msegs especifica el número de milisegundos para que el escucha SOAP de WebSphere MQ permanezca activo si se han recibido mensajes de solicitud en cualquier hebra. Si *msegs* se establece en -1, el escucha permanece activo indefinidamente.

-i ***Contexto***
Contexto especifica si los escuchas pasan el contexto de identidad. *Contexto* toma los valores siguientes:

passContext
Establezca el contexto de identidad del mensaje de solicitud original en el mensaje de respuesta. El escucha SOAP comprueba que tiene autorización para guardar el contexto de la cola de solicitudes y pasarlo a la cola de respuestas. Realiza las comprobaciones en tiempo de ejecución al abrir la cola de solicitudes para guardar el contexto y la cola de respuestas para pasar el contexto. Si no tiene la autorización necesaria, o la llamada MQOPEN falla, y el mensaje de respuesta no se procesa. El mensaje de respuesta se coloca en la cola de mensajes no entregados con la cabecera de mensaje no entregado que contiene el código de retorno del MQOPENanómalo. A continuación, el escucha continúa procesando los mensajes entrantes subsiguientes de forma normal.

ownContext
El escucha SOAP no pasa contexto. El contexto devuelto refleja el ID de usuario bajo el que se ejecuta el escucha en lugar del ID de usuario que ha creado el mensaje de solicitud original.

Los campos en el contexto de origen los establece el gestor de colas y no el escucha SOAP.

-n ***numThreads***
numThreads especifica el número de hebras en los scripts de inicio generados para el escucha SOAP de WebSphere MQ . El valor predeterminado es 10. Considere la posibilidad de aumentar este número si tiene un alto rendimiento de mensajes.

-v
-v establece la salida detallada de los mandatos externos. Los mensajes de error siempre se visualizan. Utilice -v para generar mandatos que puede adaptar para crear scripts de despliegue personalizados.

-w ***serviceDirectory***
serviceDirectory es el directorio que contiene el servicio web.

-x ***transaccionalidad***
transaccionalidad especifica el tipo de control transaccional para el escucha. La *transaccionalidad* se puede establecer en uno de los valores siguientes:

onePhase
Se utiliza el soporte de una fase de IBM WebSphere MQ . Si el sistema falla durante el proceso, el mensaje de solicitud se vuelve a entregar a la aplicación. Las transacciones de WebSphere MQ garantizan que los mensajes de respuesta se escriban exactamente una vez.

twoPhase
Se utiliza el soporte de dos fases. Si el servicio se escribe correctamente, el mensaje se entrega exactamente una vez, coordinado con otros recursos, dentro de una única ejecución confirmada del servicio. Esta opción sólo se aplica a las conexiones de enlaces de servidor.

none
Sin soporte transaccional. Si el sistema falla durante el proceso, el mensaje de solicitud se puede perder, aunque sea persistente. Es posible que el servicio se haya ejecutado o no, y que se hayan escrito mensajes de respuesta, de informe o de mensajes no entregados.

El programa de utilidad de despliegue comprueba la compatibilidad de los distintivos -x y -a . Consulte la descripción del distintivo -a para obtener detalles.

Ejemplo de Java

```
java com.ibm.mq.soap.transport.jms.SimpleJavaListener
-u "jms:/queue?destination=myQ&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi"
-n 20
```

Escuchas SOAP de WebSphere MQ

Un escucha SOAP de WebSphere MQ lee una solicitud SOAP de entrada de la cola especificada como destino en el URI. Comprueba el formato del mensaje de solicitud y, a continuación, invoca un servicio web utilizando la infraestructura de servicios web. Un escucha SOAP de WebSphere MQ devuelve cualquier respuesta o error de un servicio web utilizando la cola de destino de respuesta en el URI. Devuelve informes de WebSphere MQ a la cola de respuestas.

El término escucha se utiliza aquí en su sentido de servicios web estándar. Es distinto del escucha estándar de WebSphere MQ invocado por el mandato **runmqlsr** .

Descripción

El escucha SOAP de Java se implementa como una clase Java y ejecuta servicios utilizando Axis 1.4. El escucha .NET es una aplicación de consola y ejecuta los servicios .NET Framework 1 o .NET Framework 2. Para los servicios .NET Framework 3, utilice el canal personalizado WebSphere MQ para Microsoft Windows Communication Foundation (WCF).

El programa de utilidad de despliegue crea scripts para iniciar automáticamente escuchas SOAP de Java o .NET. Un escucha SOAP se puede iniciar manualmente utilizando el mandato **amqSOAPNETListener** o llamando a la clase `SimpleJavaListener` . Puede configurar el escucha SOAP de WebSphere MQ para que se inicie como un servicio de WebSphere MQ estableciendo la opción -s en el programa de utilidad de despliegue. De forma alternativa, inicie los escuchas utilizando el desencadenamiento, o utilice los scripts de escucha de inicio y finalización generados por el programa de utilidad de despliegue. Puede configurar el desencadenamiento manualmente o utilizar las opciones de despliegue -tmq y -tmp para configurar el desencadenamiento automáticamente. Puede finalizar un escucha estableciendo la cola de solicitudes en `GET(DISABLED)`.

Tabla 586. Scripts de mandatos generados por el programa de utilidad de despliegue

Infraestructura de servicio web	Sistemas UNIX and Linux	Windows Java	Windows .NET
Iniciar escucha	**startWMQJListener.sh**	**startWMQJListener.cmd**	**startWMQNListener.cmd**
Detener escucha	**endWMQJListener.sh**	**endWMQJListener.cmd**	**endWMQNListener.cmd**
Definir servicio de escucha	**defineWMQJListener.sh**	**defineWMQJListener.cmd**	**defineWMQNListener.cmd**

El escucha SOAP de WebSphere MQ pasa los campos `endpointURL` y `soapAction` del mensaje SOAP a la infraestructura SOAP. El escucha invoca el servicio a través de la infraestructura de servicios web y espera la respuesta. El escucha no valida `endpointURL` y `soapAction`. Los campos los establece el remitente SOAP de WebSphere MQ a partir de los datos proporcionados en el URI establecido por un cliente SOAP.

El escucha crea el mensaje de respuesta y lo envía al destino de respuesta proporcionado en el URI de mensaje de solicitud. Además, el escucha establece el ID de correlación en el mensaje de respuesta de acuerdo con la opción de informe del mensaje de solicitud. Devuelve los valores de caducidad,

persistencia y prioridad del mensaje de solicitud. El escucha también envía mensajes de informe a los clientes en algunas circunstancias.

Si hay errores de formato en la solicitud SOAP, el escucha devuelve un mensaje de informe al cliente utilizando la cola de destino de respuesta. El gestor de colas también devuelve mensajes de informe al cliente utilizando la cola de destino de respuesta, si se ha solicitado un informe. Los mensajes de informe completos se graban en la cola de respuestas, en respuesta a una serie de sucesos:

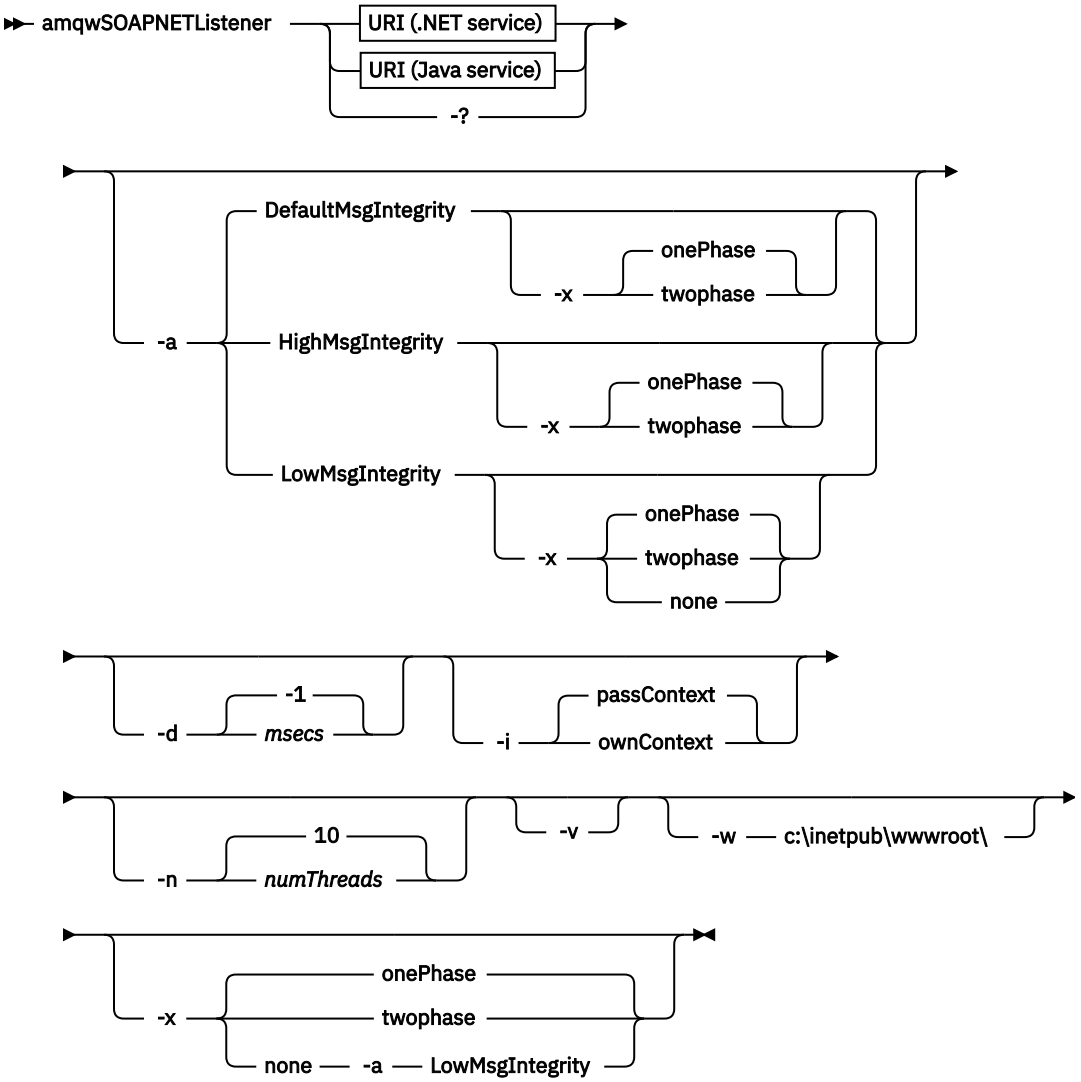
- Una excepción.
- Caducidad del mensaje.
- El formato del mensaje de solicitud no se reconoce.
- Anomalía de la comprobación de integridad de la cabecera **MQRFH2** .
- El formato del cuerpo del mensaje principal no es MQFMT_NONE.
- Se ha excedido el umbral de reintentos/reintentos mientras el escucha SOAP de WebSphere MQ está procesando la solicitud.

El emisor SOAP de WebSphere MQ establece las opciones de informe MQRO_EXCEPTION_WITH_FULL_DATA y MQRO_EXPIRATION_WITH_FULL_DATA . Como resultado de las opciones de informe establecidas por el remitente SOAP de WebSphere MQ , el mensaje de informe contiene todo el mensaje de solicitud de origen. El remitente SOAP de WebSphere MQ también establece la opción MQRO_DISCARD , que hace que el mensaje se descarte después de que se haya devuelto un mensaje de informe. Si las opciones de informe no cumplen sus requisitos, escriba sus propios remitentes para utilizar distintas opciones de informe MQRO_EXCEPTION y MQRO_DISCARD . Si la solicitud SOAP la envía un remitente diferente que no ha establecido MQRO_DISCARD, el mensaje anómalo se graba en la cola de mensajes no entregados (DLQ).

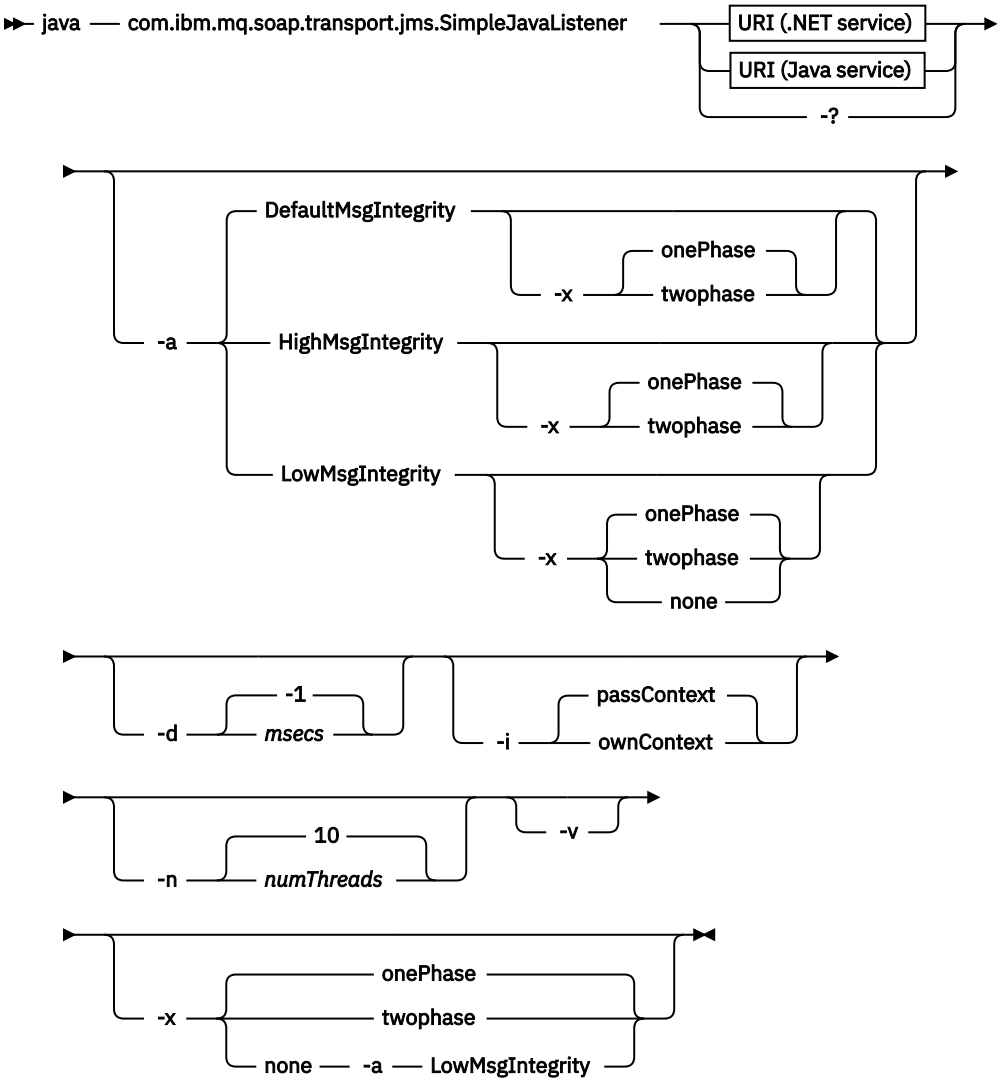
Si el escucha genera un mensaje de informe pero falla en el proceso de enviar el informe, el mensaje de informe se envía a la DLQ. Asegúrese de que el manejador DLQ maneja estos mensajes correctamente.

Si se produce un error al intentar grabar en la cola de mensajes no entregados, se graba un mensaje en el registro de errores de WebSphere MQ . Si el escucha continúa procesando más mensajes depende de la persistencia de mensajes y de las opciones transaccionales que se seleccionen. Si el escucha se ejecuta en modalidad transaccional de una fase y está procesando un mensaje de solicitud no persistente, el mensaje original se descarta. El escucha SOAP de WebSphere MQ continúa ejecutándose. Si el mensaje de solicitud es persistente, el mensaje de solicitud se restituye en la cola de solicitudes y el escucha sale. La cola de solicitudes se establece en get-inhibido para evitar un reinicio desencadenado accidentalmente.

Syntax diagram
.NET


amqwSOAPNETListener
URI (.NET service)
URI (Java service)
-?
-a
DefaultMsgIntegrity
-x
onePhase
twophase
HighMsgIntegrity
-x
onePhase
twophase
LowMsgIntegrity
-x
onePhase
twophase
none
-d
-1
msecs
-i
passContext
ownContext
-n
10
numThreads
-v
-w
c:\inetpub\wwwroot\
-x
onePhase
twophase
none
-a
LowMsgIntegrity

Java



Parámetros necesarios

URI ***plataforma***
Consulte “Sintaxis de URI y parámetros para el despliegue de servicios web” en la página 993.

-?
Imprima el texto de ayuda que describe cómo se utiliza el mandato.

Parámetros opcionales

-a ***integrityOption***
integrityOption especifica el comportamiento de los escuchas SOAP de WebSphere MQ si no es posible colocar un mensaje de solicitud fallida en la cola de mensajes no entregados. *integrityOption* puede tomar uno de los valores siguientes:

DefaultMsgIntegrity
Para los mensajes no persistentes, el escucha muestra un mensaje de aviso y continúa ejecutándose con el mensaje original que se descarta. Para los mensajes persistentes, muestra un mensaje de error, restituye el mensaje de solicitud para que permanezca en la cola de solicitudes y salga. DefaultMsgIntegrity se aplica si se omite la opción -a o si no se especifica *integrityOption* .

LowMsgIntegrity
Para los mensajes persistentes y no persistentes, el escucha muestra un aviso y continúa ejecutándose, descartando el mensaje.

HighMsgIntegrity
Para los mensajes persistentes y no persistentes, el escucha muestra un mensaje de error, restituye el mensaje de solicitud para que permanezca en la cola de solicitudes y salga.

El programa de utilidad de despliegue comprueba la compatibilidad de los distintivos -x y -a . Si se especifica -x none , se debe especificar -a LowMsgIntegrity . Si los distintivos son incompatibles, el programa de utilidad de despliegue sale con un mensaje de error y no se han realizado pasos de despliegue.

-d ***msegs***
msegs especifica el número de milisegundos para que el escucha SOAP de WebSphere MQ permanezca activo si se han recibido mensajes de solicitud en cualquier hebra. Si *msegs* se establece en -1, el escucha permanece activo indefinidamente.

-i ***Contexto***
Contexto especifica si los escuchas pasan el contexto de identidad. *Contexto* toma los valores siguientes:

passContext
Establezca el contexto de identidad del mensaje de solicitud original en el mensaje de respuesta. El escucha SOAP comprueba que tiene autorización para guardar el contexto de la cola de solicitudes y pasarlo a la cola de respuestas. Realiza las comprobaciones en tiempo de ejecución al abrir la cola de solicitudes para guardar el contexto y la cola de respuestas para pasar el contexto. Si no tiene la autorización necesaria, o la llamada MQOPEN falla, y el mensaje de respuesta no se procesa. El mensaje de respuesta se coloca en la cola de mensajes no entregados con la cabecera de mensaje no entregado que contiene el código de retorno del MQOPENanómalo. A continuación, el escucha continúa procesando los mensajes entrantes subsiguientes de forma normal.

ownContext
El escucha SOAP no pasa contexto. El contexto devuelto refleja el ID de usuario bajo el que se ejecuta el escucha en lugar del ID de usuario que ha creado el mensaje de solicitud original.

Los campos en el contexto de origen los establece el gestor de colas y no el escucha SOAP.

-n ***numThreads***
numThreads especifica el número de hebras en los scripts de inicio generados para el escucha SOAP de WebSphere MQ . El valor predeterminado es 10. Considere la posibilidad de aumentar este número si tiene un alto rendimiento de mensajes.

-v
-v establece la salida detallada de los mandatos externos. Los mensajes de error siempre se visualizan. Utilice -v para generar mandatos que puede adaptar para crear scripts de despliegue personalizados.

-w ***serviceDirectory***
serviceDirectory es el directorio que contiene el servicio web.

-x ***transaccionalidad***
transaccionalidad especifica el tipo de control transaccional para el escucha. La *transaccionalidad* se puede establecer en uno de los valores siguientes:

onePhase
Se utiliza el soporte de una fase de IBM WebSphere MQ . Si el sistema falla durante el proceso, el mensaje de solicitud se vuelve a entregar a la aplicación. Las transacciones de WebSphere MQ garantizan que los mensajes de respuesta se escriban exactamente una vez.

twoPhase
Se utiliza el soporte de dos fases. Si el servicio se escribe correctamente, el mensaje se entrega exactamente una vez, coordinado con otros recursos, dentro de una única ejecución confirmada del servicio. Esta opción sólo se aplica a las conexiones de enlaces de servidor.

none
Sin soporte transaccional. Si el sistema falla durante el proceso, el mensaje de solicitud se puede perder, aunque sea persistente. Es posible que el servicio se haya ejecutado o no, y que se hayan escrito mensajes de respuesta, de informe o de mensajes no entregados.

El programa de utilidad de despliegue comprueba la compatibilidad de los distintivos -x y -a . Consulte la descripción del distintivo -a para obtener detalles.

Ejemplo de .NET

```
amqwSOAPNETlistener
-u "jms:/queue?destination=myQ&connectionFactory=()
&targetService=myService&initialContextFactory=com.ibm.mq.jms.Nojndi"
-w C:/wmqsoap/demos
-n 20
```

Ejemplo de Java

```
java com.ibm.mq.soap.transport.jms.SimpleJavaListener
-u "jms:/queue?destination=myQ&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi"
-n 20
```

Transporte de IBM WebSphere MQ para remitente SOAP

Se proporcionan clases de remitente para Axis y .NET Framework 1 y .NET Framework 2. El emisor construye una solicitud SOAP y la coloca en una cola y, a continuación, se bloquea hasta que ha leído una respuesta de la cola de respuestas. Puede modificar el comportamiento de las clases pasando diferentes URI de un cliente SOAP. Para .NET Framework 3, utilice el canal personalizado WebSphere MQ para Microsoft Windows Communication Foundation (WCF).

Finalidad

El emisor SOAP de WebSphere MQ coloca una solicitud SOAP para invocar un servicio web en una cola de solicitudes WebSphere MQ . El remitente establece los campos en la cabecera **MQRFH2** de acuerdo con las opciones especificadas en el URI, o de acuerdo con los valores predeterminados.

Si necesita cambiar el comportamiento de un remitente más allá de lo que es posible utilizando las opciones de URI, escriba su propio remitente. El remitente puede trabajar con el transporte IBM WebSphere MQ para escuchas SOAP o con otros entornos SOAP. El remitente debe construir mensajes SOAP en el formato definido por WebSphere MQ. El formato está soportado por el escucha SOAP de IBM WebSphere MQ y también por los escuchas SOAP proporcionados por WebSphere Application Server y CICS. El remitente debe seguir las reglas de un solicitante de IBM WebSphere MQ . El escucha SOAP de IBM WebSphere MQ devuelve mensajes de respuesta y de informe. Consulte “Valores SOAP de MQMD” en la página 969 para obtener detalles sobre cómo establecer las opciones de informe en **MQMD**. Las opciones de informe controlan los mensajes de informe devueltos por el escucha SOAP de WebSphere MQ .

Descripción

El emisor Java SOAP de WebSphere MQ se registra con el entorno de host Axis para el prefijo URI `jms:` . El remitente se implementa en la clase `com.ibm.mq.soap.transport.jms.WMQSender`, que se deriva de `org.apache.axis.handlers.BasicHandler`. Si el entorno de host Axis detecta un prefijo de URI `jms:` , invoca la clase `com.ibm.mq.soap.transport.jms.WMQSender` . La clase se bloquea después de colocar el mensaje hasta que ha leído una respuesta de la cola de respuestas. Si no se recibe ninguna respuesta dentro de un intervalo de tiempo de espera, el remitente emite una excepción. Si se recibe una respuesta dentro del intervalo de tiempo de espera, el mensaje de respuesta se devuelve al cliente utilizando la infraestructura Axis. La aplicación cliente debe poder manejar estos mensajes de respuesta.

Para los servicios Microsoft .NET Framework 1 y .NET Framework 2, el remitente SOAP de WebSphere MQ se implementa en la clase `IBM.WMQSOAP.MQWebRequest`, que se deriva de `System.Net.WebRequest` y `System.Net.IwebRequestCreate`. Si .NET Framework 1 o .NET Framework 2 detecta un prefijo de URI de `jms:` , invoca la clase `IBM.WMQSOAP.MQWebRequest` . El emisor crea un objeto `MQWebResponse` para leer el mensaje de respuesta de la cola de respuestas y devolverlo al cliente.

`com.ibm.mq.soap.transport.jms.WMQSender` es una clase final y `IBM.WMQSOAP.MQWebRequest` está sellado. No puede modificar su comportamiento creando subclases.

Parámetros

Establezca el URI para controlar el comportamiento del remitente SOAP de IBM WebSphere MQ en un cliente SOAP de servicio web. El programa de utilidad de despliegue crea apéndices de cliente de servicio web que incorporan las opciones de URI proporcionadas al programa de utilidad de despliegue.

Utilice una tabla de definición de canal con el transporte SOAP de WebSphere MQ para el remitente SOAP

Una definición de canal de conexión de cliente es una alternativa a establecer propiedades de conexión en el atributo ConnectionFactory del URI de servicio web. Las propiedades de conexión son los parámetros clientChannel, clientConnectiony SSL .

Descripción

Cree la tabla de descripciones de canal de cliente definiendo las conexiones de cliente. Incluso si un cliente de servicios Web se conecta a distintos gestores de colas, cree todas las conexiones en la tabla de conexiones en un único gestor de colas. El nombre y la ubicación predeterminados de la tabla de conexiones es *queue manager directory*`/@ipcc/AMQCLCHL.TAB`.

Pase la ubicación de la tabla de conexiones a un cliente Java estableciendo la propiedad del sistema `com.ibm.mq.soap.transport.jms.mqchlurl` .

Pase la ubicación de la tabla de conexión a un cliente .NET estableciendo las variables de entorno `MQCHLLIB` y `MQCHLTAB` .

Puede proporcionar una tabla de conexiones de canal y parámetros de conexión de canal en el atributo ConnectionFactory del URI de servicio web. Los valores establecidos en ConnectionFactory tienen prioridad sobre los valores de la tabla de definiciones de canal.

Utilización de una tabla de definiciones de canal en Java

```
java -Dcom.ibm.msg.client.config.location=file:/C:/mydir/myjms.config MyAppClass
```

Figura 22. Inicio del cliente Java utilizando un archivo de configuración

```
com.ibm.mq.soap.transport.jms.mqchlurl=file:/C:/ibm/wmq/qmgrs/QM1/@ipcc/AMQCLCHL.TAB
```

Figura 23. `myjms.config`

Transacciones

Utilice la opción `-x` al iniciar el escucha, para ejecutar servicios web de forma transaccional. Seleccione la integridad de los mensajes estableciendo la opción `persistence` en el URI de servicio.

Servicios web

Utilice la opción `-x` al iniciar el escucha, para ejecutar servicios web de forma transaccional. En .NET Framework 1 y 2, el escucha SOAP utiliza Microsoft Transaction Coordinator (MTS). En el eje 1.4, el escucha SOAP utiliza transacciones coordinadas del gestor de colas.

Clientes de servicio web

Los remitentes SOAP no son transaccionales.

Enlaces de WebSphere MQ

Puede establecer el tipo de enlace para el remitente SOAP. Se puede conectar como una aplicación de servidor de WebSphere MQ o como una aplicación cliente. También puede enlazar el emisor SOAP como un cliente XA en .NET.

Persistencia de los mensajes

Seleccione el nivel de persistencia estableciendo la opción `Persistence` en el URI.

Transacciones de servicio web

Puede utilizar transacciones de servicio web, porque el remitente SOAP no es transaccional. Si escribe su propio remitente SOAP y tiene la intención de utilizar transacciones de servicio web, no cree un remitente SOAP transaccional. No puede enviar el mensaje de solicitud y recibir el mensaje de respuesta en la misma transacción. La transacción de servicio web no debe coordinar el envío y la recepción.

Sintaxis de URI y parámetros para el despliegue de servicios web

La sintaxis y los parámetros para desplegar un servicio web de IBM WebSphere MQ se definen en un URI. El programa de utilidad de despliegue genera un URI predeterminado basado en el nombre del servicio web. Puede alterar temporalmente los valores predeterminados definiendo su propio URI como parámetro para el programa de utilidad de despliegue. El programa de utilidad de despliegue incorpora el URI en los apéndices de cliente de servicio web generados.

Finalidad

Un servicio web se especifica utilizando un URI (identificador universal de recursos). El diagrama de sintaxis especifica el URI soportado en el transporte IBM WebSphere MQ para SOAP. El URI controla los parámetros SOAP específicos de IBM WebSphere MQy las opciones utilizadas para acceder a los servicios de destino. El URI es compatible con los servicios web alojados por .NET, Apache Axis 1, WebSphere Application Server, CICS.

Descripción

El URI se incorpora en las clases de cliente de servicio web generadas por el programa de utilidad de despliegue. El cliente pasa el URI a IBM WebSphere MQ SOAP Sender en un mensaje IBM WebSphere MQ . El URI controla el proceso realizado por el remitente SOAP de IBM WebSphere MQ y el escucha SOAP de IBM WebSphere MQ .

Syntax

The URI syntax is as follows:

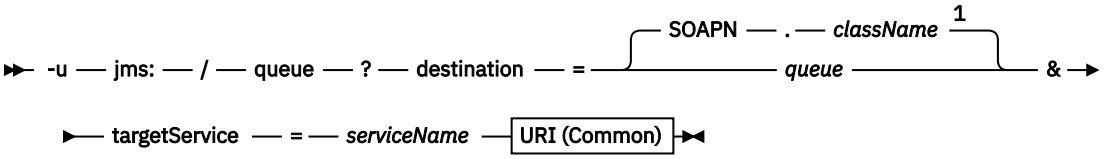
```
jms:/queue?name=value&name=value...
```

where `name` is a parameter name and *value* is an appropriate value, and the `name`=*value* element can be repeated any number of times with the second and subsequent occurrences being preceded by an ampersand (&).

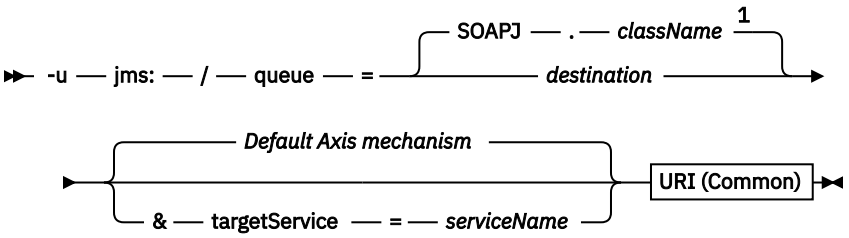
Parameter names are case-sensitive, as are names of IBM WebSphere MQ objects. If any parameter is specified more than once, the final occurrence of the parameter takes effect. Client applications can override a generated parameter by appending another copy of the parameter to the URI. If any additional unrecognized parameters are included, they are ignored.

If you store a URI in an XML string, you must represent the ampersand character as `&`. Similarly, if a URI is coded in a script, take care to escape characters such as & that would otherwise be interpreted by the shell.

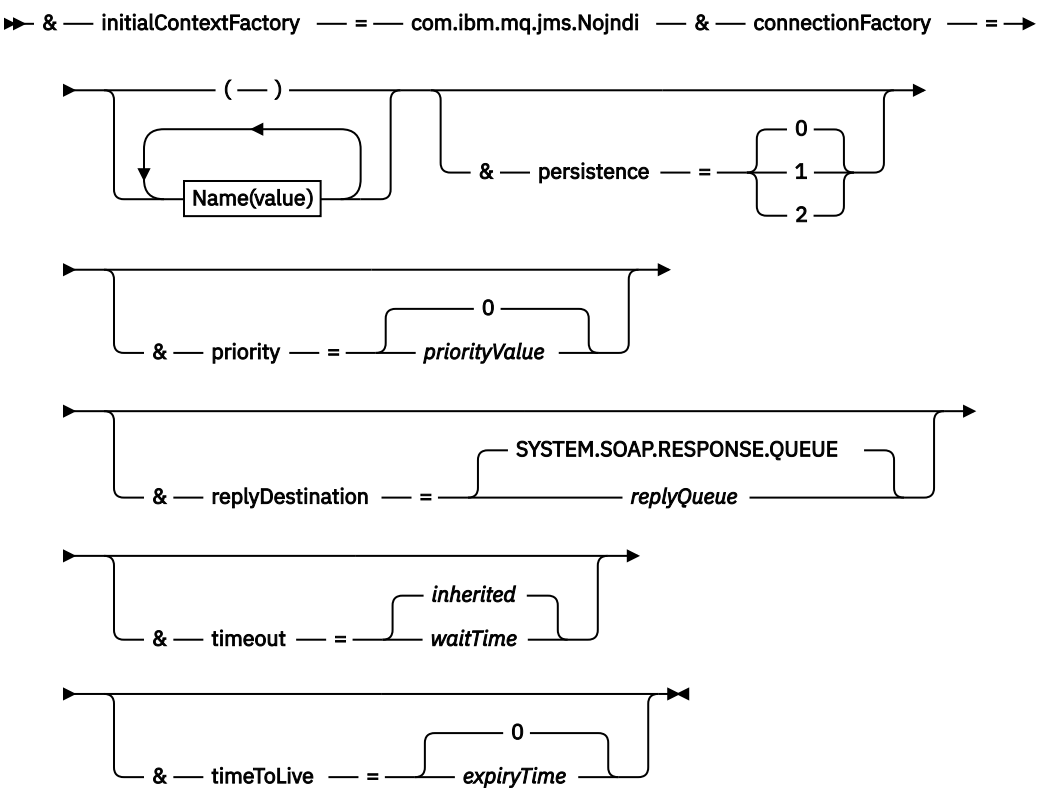
Syntax diagram
URI (.NET service)



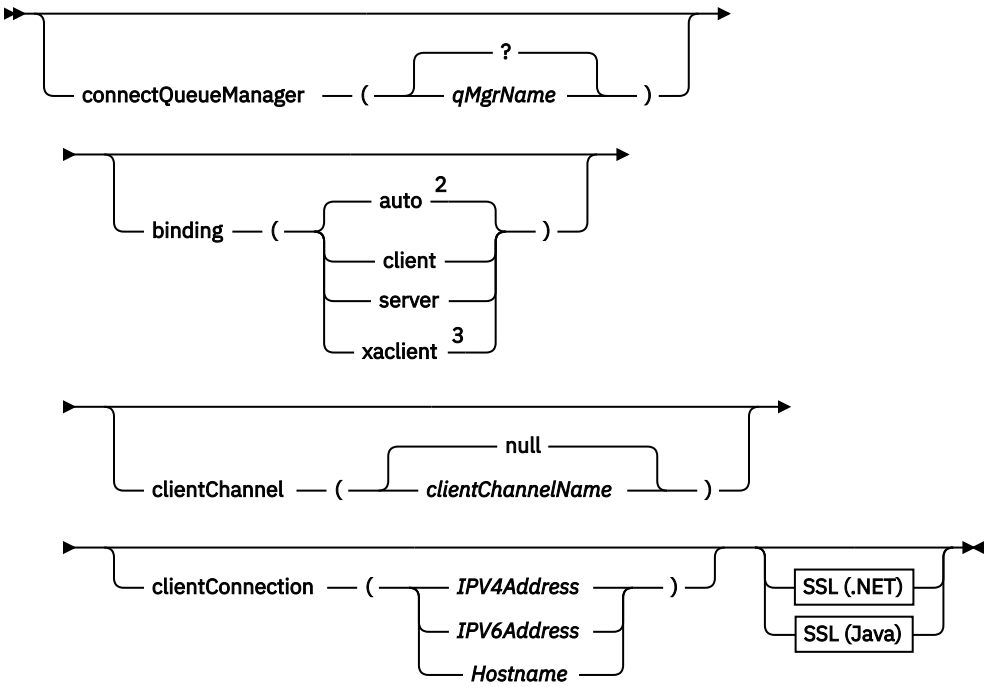
URI (Java service)



URI (Common)



***Name*(*value*)**



Notas:

[1] The queue manager transforms *className* to a queue name following the steps described in "Transformación de destino a nombre de cola" en la página 995

[2] `client` is the default if other options appropriate for a client are specified; for example `clientConnection`.

[3] `xaclient` applies to .NET only

Transformación de destino a nombre de cola

1. *className* tiene el prefijo `SOAPJ.` para los servicios de Java o `SOAPN.` para los servicios .NET.
2. La extensión de archivo se elimina del nombre completo de vía de acceso proporcionado en el parámetro *className* .
3. La serie resultante se trunca en no más de 48 caracteres
4. Los caracteres de separador de directorio se sustituyen por caracteres de punto.
5. Los espacios intercalados se sustituyen por caracteres de subrayado.
6. Los dos puntos que siguen a una letra de prefijo de unidad se sustituyen por un punto para un servicio .NET.

Nota: En algunos entornos, es posible que un nombre de cola generado por el programa de utilidad de despliegue no sea exclusivo. El programa de utilidad de despliegue comprueba si se debe crear la cola. Puede optar por alterar temporalmente el programa de utilidad de despliegue reestructurando la jerarquía de directorios de despliegue o personalizando el proceso de despliegue proporcionado.

Parámetros de URI necesarios

`destination=`*cola*

cola es el nombre del destino de solicitud. Puede ser una cola o un alias de cola. Si es alias de cola, el alias se puede resolver en un tema.

- Si se omite el parámetro -u , *cola* se genera a partir de *nombreclase* utilizando los pasos descritos en "Transformación de destino a nombre de cola" en la página 995.
- Si se especifica el parámetro-u , *queue* es necesario y debe ser el primer parámetro del URI después del `jms:/queue?` inicial serie. Especifique un nombre de cola IBM WebSphere MQ o un

nombre de cola y un nombre de gestor de colas conectados mediante un símbolo @, por ejemplo `SOAPN.trandemos@WMQSOAP.DEMO.QM`.

- El programa de utilidad de despliegue comprueba si el nombre de cola, generado o proporcionado, coincide con el nombre de una cola existente. La acción realizada se describe en Tabla 587 en la página 996.

Tabla 587. Validación de cola

¿Existe el script de escucha?	**El script de escucha existe en el directorio `./generated/server`**		**El script de escucha no existe en el directorio `./generated/server`**
¿La cola del script de escucha coincide con *cola*?	***cola* no coincide con la cola de solicitudes que se utiliza en el script de escucha**	***cola* coincide con la cola de solicitudes que se utiliza en el script de escucha**	
***cola* existe**	– *Salidas de despliegue con un error.* – El servicio ya se ha desplegado en `./generated/server`, pero utilizando una cola diferente.	– **El despliegue continúa normalmente.** – El servicio ya se ha desplegado en `./generated/server`	– *Salidas de despliegue con un error.* – El script de inicio de escucha no se encuentra en `./generated/server`, pero la *cola* está siendo utilizada por un servicio o aplicación diferente.
***cola* no existe**		– **El despliegue continúa con un aviso.** – Es posible que el despliegue anterior haya fallado, porque el inicio es válido, pero falta la *cola* .	– **El despliegue continúa normalmente.** – No se ha desplegado ningún servicio desde este directorio.

`&connectionFactory=`*Nombre (valor)*

Nombre es uno de los parámetros siguientes:

- connectQueueManager (qMgrNombre)
- binding (bindingType)
- clientChannel(canal)
- clientConnection(conexión)
- "Parámetros SSL necesarios (Java)" en la página 982

Consulte "Parámetros de fábrica de conexiones" en la página 998 para obtener una descripción de los valores de estos parámetros.

`&targetService=`*serviceName*

[8]En .NET, *serviceName* es el nombre de un servicio .NET ubicado en el directorio de despliegue, por ejemplo: `targetService=myService.asmx`. En el entorno .NET, el parámetro targetService hace posible que un único escucha SOAP de WebSphere MQ pueda procesar solicitudes de varios servicios. Estos servicios deben desplegarse desde el mismo directorio.

[8] Sólo servicio .NET

Parámetros de URI opcionales

&initialContextFactory=*contextFactory*
contextFactory es necesario y debe establecerse en `com.ibm.mq.jms.Nojndi`. Asegúrese de que `Nojndi.jar` esté en la vía de acceso de clases para un cliente de servicios web de WebSphere Application Server. `Nojndi.jar` devuelve objetos Java basados en el contenido de los parámetros `connectionFactory` y `destination` , en lugar de hacerlo por referencia a un directorio.

&targetService=*serviceName*
[9]En Axis, *serviceName* es el nombre completo de un servicio de Java , por ejemplo: `targetService=javaDemos.service.StockQuoteAxis`. Si no se especifica targetService , se carga un servicio utilizando el mecanismo Axis predeterminado.

&persistence=*messagePersistence*
messagePersistence toma uno de los valores siguientes:

0
La persistencia se hereda de la definición de cola.

1
El mensaje no es persistente.

2
El mensaje es persistente

&priority=*priorityValue*
priorityValue está en el rango de 0 a 9. 0 es de prioridad baja. El valor predeterminado es específico del entorno, que en el caso de IBM WebSphere MQ es 0.

&replyDestination=*replyToCola*
La cola en el lado del cliente que se utilizará para el mensaje de respuesta. La cola de respuestas predeterminada es `SYSTEM.SOAP.RESPONSE.QUEUE`.

- Ejecute el script `setupWMQSOAP` para crear los objetos SOAP predeterminados de WebSphere MQ .
- Especifique una cola modelo para *replyToQueue* para crear una cola de respuestas dinámica temporal o permanente. Para las colas de respuestas dinámicas temporales y permanentes, se crea una instancia separada de cola dinámica para cada solicitud. Si se produce alguno de los sucesos siguientes, se suprime la cola:
  - La respuesta llega y se procesa.
  - La solicitud excede el tiempo de espera.
  - El programa solicitante termina.

  Para obtener el mejor rendimiento, utilice colas dinámicas temporales en lugar de colas dinámicas permanentes. No envíe un mensaje de solicitud persistente a un URI con una cola dinámica temporal. El SOAP de escucha de IBM WebSphere MQ no puede procesar el mensaje y genera un error. El cliente excede el tiempo de espera de la respuesta.
- El script `setupWMQSOAP` crea una cola de modelo dinámico permanente predeterminada denominada `SYSTEM.SOAP.MODEL.RESPONSE.QUEUE`.

&timeout=*waitTime*
Tiempo, en milisegundos, que el cliente espera un mensaje de respuesta. *waitTime* altera temporalmente los valores establecidos por la infraestructura o la aplicación cliente. Si no se especifica el valor de aplicación, si se especifica, o se hereda el valor predeterminado de infraestructura.

Nota: No se aplica ninguna relación entre el tiempo de espera y timeToLive.

&timeToLive=*expiryTime*
expiryTime es el tiempo, especificado en milisegundos, antes de que caduque el mensaje. El valor predeterminado es cero, que indica un tiempo de vida ilimitado.

Nota: No se aplica ninguna relación entre el tiempo de espera y timeToLive.

[9] Sólo servicio Java

Parámetros de fábrica de conexiones

connectQueueManager(*qMgrNombre*)
qMgrNombre especifica el gestor de colas al que se conecta el cliente. El valor predeterminado es en blanco.

binding(*bindingType*)
bindingType especifica cómo se conecta el cliente a *qMgrNombre*. El valor predeterminado es auto. *bindingType* toma los valores siguientes:

auto
El remitente intenta los siguientes tipos de conexión, en orden:

1. Si se especifican otras opciones adecuadas para una conexión de cliente, el remitente utiliza un enlace de cliente. Las otras opciones son clientConnection o clientChannel.
2. Utilice una conexión de servidor.
3. Utilice una conexión de cliente.

Utilice `binding(auto)` en el *URI* si no hay ningún gestor de colas local en el cliente SOAP. Se crea una conexión de cliente para el cliente SOAP.

client
Utilice `binding(cliente)` en el *URI* para crear una configuración de cliente para el remitente SOAP.

server
Utilice `binding(servidor)` en el *URI* para crear una configuración de servidor para el remitente SOAP. Si la conexión tiene parámetros de tipo de cliente, la conexión falla y el remitente SOAP de IBM WebSphere MQ muestra un error. Los parámetros de tipo de cliente son clientConnection, clientChannelo parámetros SSL.

xaclient
xaclient sólo es aplicable en .NET y no para clientes Java. Utilice una conexión de cliente XA.

clientChannel(*canal*)

El cliente SOAP utiliza *channel* para realizar una conexión de cliente IBM WebSphere MQ . *channel* debe coincidir con el nombre de un canal de conexión de servidor, a menos que la definición automática de canal esté habilitada en el servidor. clientChannel es un parámetro necesario, a menos que haya proporcionado una tabla de definición de conexión de cliente (CCDT).

Proporcione una CCDT en Java estableciendo `com.ibm.mq.soap.transport.jms.mqchlurl`. En .NET, establezca las variables de entorno `MQCHLLIB` y `MQCHLTAB` ; consulte "Utilice una tabla de definición de canal con el transporte SOAP de WebSphere MQ para el remitente SOAP" en la página 992.

clientConnection(*conexión*)
El cliente SOAP utiliza *conexión* para realizar una conexión de cliente IBM WebSphere MQ . El nombre de host predeterminado es localhosty el puerto predeterminado es 1414. Si la *conexión* es una dirección TCP/IP, toma uno de los tres formatos y puede tener como sufijo un número de puerto.

Los clientes JMS pueden utilizar el formato: `hostname:port` o 'escapar' los corchetes utilizando el formato %X donde X es el valor hexadecimal que representa el carácter de corchete en la página de códigos del URI. Por ejemplo, en ASCII, %28 y %29 para `(` y `)` respectivamente.

Los clientes .Net pueden utilizar los corchetes explícitamente: `hostname(port)` o utilizar el formato 'escape'.

Dirección IPv4
Por ejemplo, `192.0.2.0`.

Dirección IPv6
Por ejemplo, `2001:DB8:0:0:0:0:0:0`.

Nombre de host
Por ejemplo, `www.example.com%281687%29`, `www.example.com:1687` o `www.example.com(1687)`.

SSL ***plataforma***
Consulte “Parámetros SSL necesarios (Java)” en la página 982

URI de ejemplo

Nota:

1. & en el URI se codifica como &
2. Todos los parámetros listados anteriormente son aplicables a los clientes.
3. Solo **destination**, **connectionFactory** y **initialContextFactory** son aplicables al servicio WCF.

```
jms:/queue?
destination=myQ&amp;amp;connectionFactory=()&amp;amp;initialContextFactory=com.ibm.mq.jms.Nojndi
```

Figura 24. URI para un servicio Axis, que proporciona sólo los parámetros necesarios

```
jms:/queue?destination=myQ&amp;amp;connectionFactory=()&amp;amp;targetService=MyService.asmx
&amp;amp;initialContextFactory=com.ibm.mq.jms.Nojndi
```

Figura 25. URI para un servicio .NET, proporcionando sólo los parámetros necesarios

```
jms:/queue?destination=myQ@myRQM&amp;amp;connectionFactory=connectQueueManager(myconnQM)
binding(client)clientChannel(myChannel)clientConnection(myConnection)
&amp;amp;initialContextFactory=com.ibm.mq.jms.Nojndi
```

Figura 26. URI para un servicio Axis, que proporciona algunos parámetros opcionales de connectionFactory

```
jms:/queue?destination=myQ@myRQM&amp;amp;connectionFactory=connectQueueManager(myconnQM)
binding(client)clientChannel(myChannel)clientConnection(myConnection)
sslPeerName(CN=MQ Test 1,O=IBM,S=Hampshire,C=GB)
&amp;amp;initialContextFactory=com.ibm.mq.jms.Nojndi
```

Figura 27. URI para un servicio Axis, que proporciona la opción sslPeerName del parámetro connectionFactory

El mecanismo `Nojndi`

El mecanismo `Nojndi` permite que los programas JMS, que utilizan interfaces JNDI, utilicen el mismo URI que los programas WebSphere MQ , que no utilizan JNDI.

Puede utilizar el transporte WebSphere MQ para SOAP para invocar servicios web en WebSphere Application Server. WebSphere Application Server SOAP sobre JMS busca los recursos JMS utilizando JNDI. El cliente de servicio web puede estar en ejecución en .NET, o utilizando Axis 1.4, para invocar el servicio web y no utilizar JNDI. Para utilizar el mismo URL para el cliente y el servidor, debe proporcionar la misma información si el entorno utiliza JNDI o no.

El URI pasado al transporte WebSphere MQ para SOAP por un cliente de servicio web contiene un gestor de colas y nombres de cola específicos de WebSphere MQ . Estos nombres son analizados y utilizados directamente por el soporte SOAP de WebSphere MQ .

El mecanismo `Nojndi` dirige el initialContextFactory utilizado por un programa JMS a `com.ibm.mq.jms.Nojndi`. La clase `com.ibm.mq.jms.Nojndi` es una implementación de la interfaz JNDI que devuelve connectionFactory y destino del URL como ConnectionFactory y objetos Java Queue . Si la implementación JMS es WebSphere MQ, MQConnectionFactory y MQQueue heredan de las clases ConnectionFactory y Queue .

Utilizando el mecanismo `Nojndi` , puede proporcionar la misma información de conexión a WebSphere Application Server y .NET utilizando el mismo URL.

W3C URI de SOAP sobre JMS para el cliente WebSphere MQ Axis 2

Defina un URI de SOAP sobre JMS W3C para llamar a un servicio web desde un cliente Axis 2 utilizando WebSphere MQ JMS como transporte SOAP. El servicio web debe proporcionarlo un servidor que soporte WebSphere MQ JMS y la recomendación de candidato SOAP sobre JMS W3C para el enlace SOAP/JMS.

Descripción

La recomendación candidata de W3C define el enlace SOAP sobre JMS; SOAP sobre Java Message Service 1.0. También es útil para sus ejemplos URI Scheme for Java (tm) Message Service 1.0[10].

Utilice el diagrama de sintaxis para crear los URI de SOAP sobre JMS de W3C que sean sintácticamente correctos y que sean aceptados por el cliente de WebSphere MQ Axis 2. Se limita a definir el URI aceptado por el cliente de WebSphere MQ Axis 2. Es un subconjunto de la recomendación W3C en dos aspectos:

1. El tema jms-variante tema no está soportado y no se debe especificar en un URI pasado al cliente WebSphere MQ Axis 2.
2. Las propiedades siguientes se omiten del diagrama de sintaxis porque son propiedades JMS y no forman parte del URI.
   a. bindingVersion
   b. contentType
   c. soapAction
   d. requestURI
   e. isFault

Las propiedades JMS las establece el cliente o el servidor Axis 2.

El diagrama amplía la recomendación W3C definiendo un parámetro personalizado, connectionFactory. connectionFactory se utiliza como alternativa a JNDI para especificar cómo se conecta el cliente Axis 2 a un gestor de colas utilizando una cola.

El cliente WebSphere MQ Axis 2 sólo acepta propiedades como parte del URI pasado al cliente por la aplicación cliente o como variables de entorno. El cliente WebSphere MQ Axis 2 no tiene capacidad para procesar un documento WSDL. La aplicación cliente o una herramienta de desarrollo puede procesar el WSDL y crear el URI para pasarlo al cliente Axis 2. Una aplicación cliente WebSphere MQ Axis 2 no puede establecer las propiedades del mensaje JMS directamente.

Syntax

In accordance with the W3C recommendation, all the parameters can be obtained from environment variables. The environment variable names are formed by prefacing the parameter name with `soapjms_`. The syntax is: `soapjms_`*parameterName*; for example,

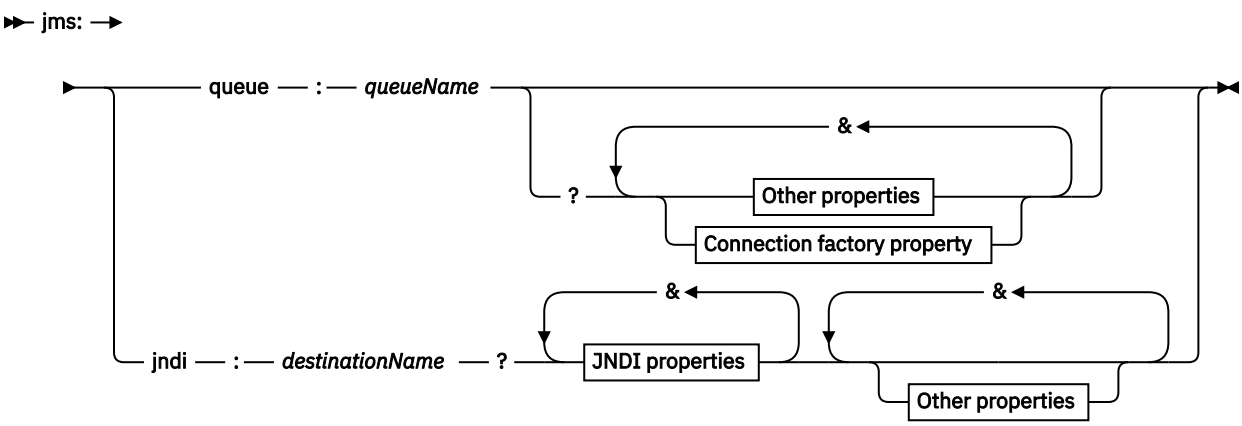
```
set soapjms_targetServer=com.example.org.stockquote
```

If a parameter is set using an environment variable it overrides the value set in the URI.

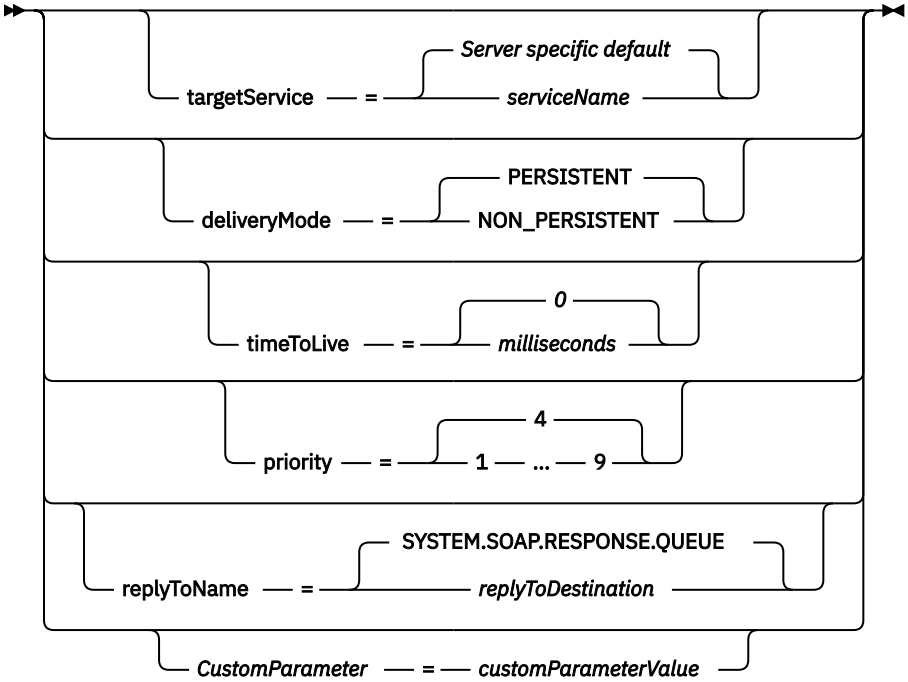
In accordance with the W3C recommendation, all the parameters can be repeated. The last instance of a parameter is used, unless overridden by an environment variable.

[10] Busque *URI Scheme for JMS*, en las referencias de especificación de W3C, para obtener el borrador más reciente.

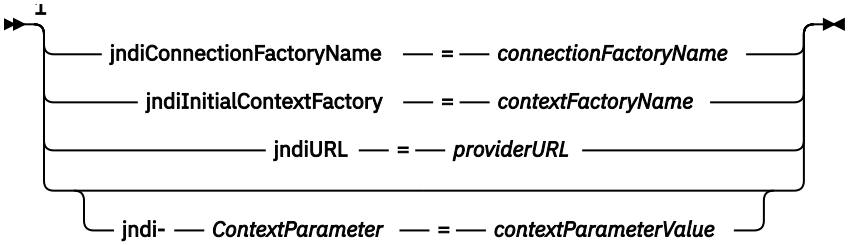
jms-uri



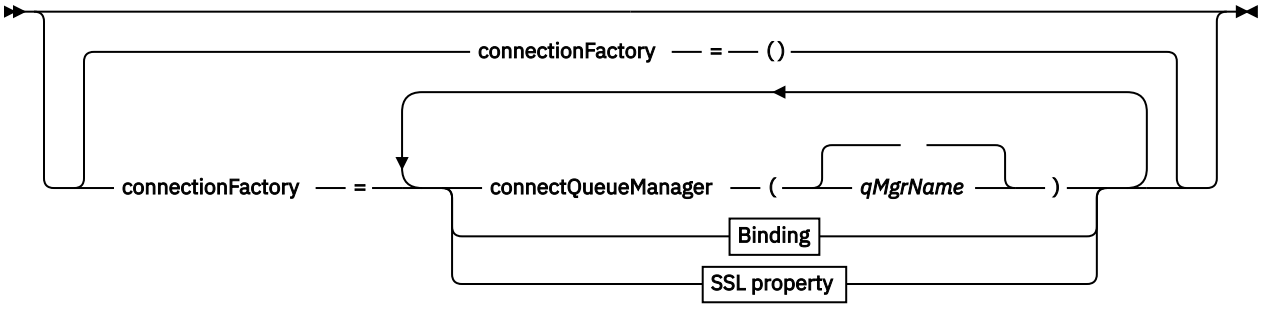
Other properties



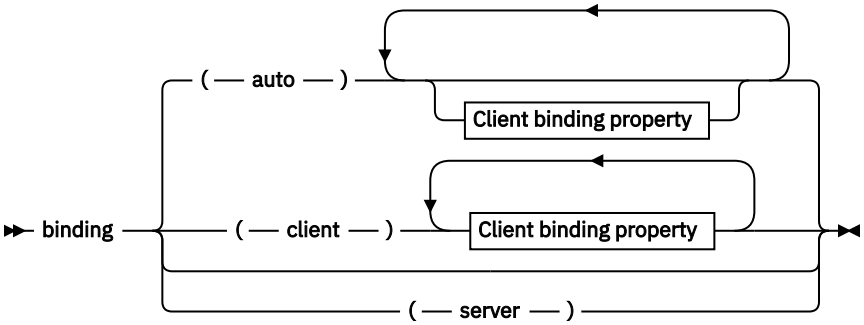
JNDI properties



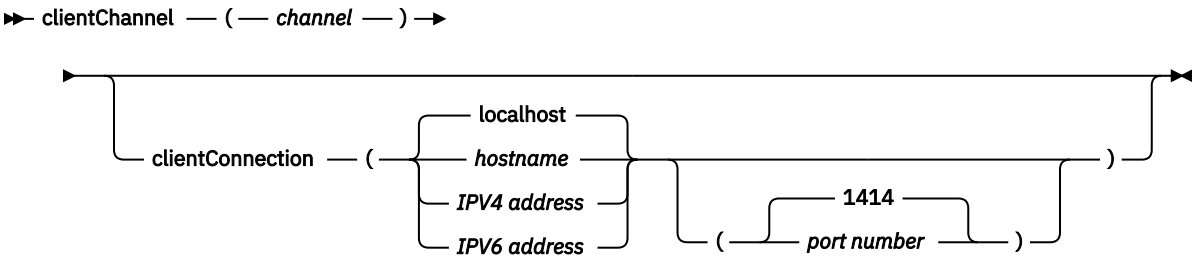
Connection factory property



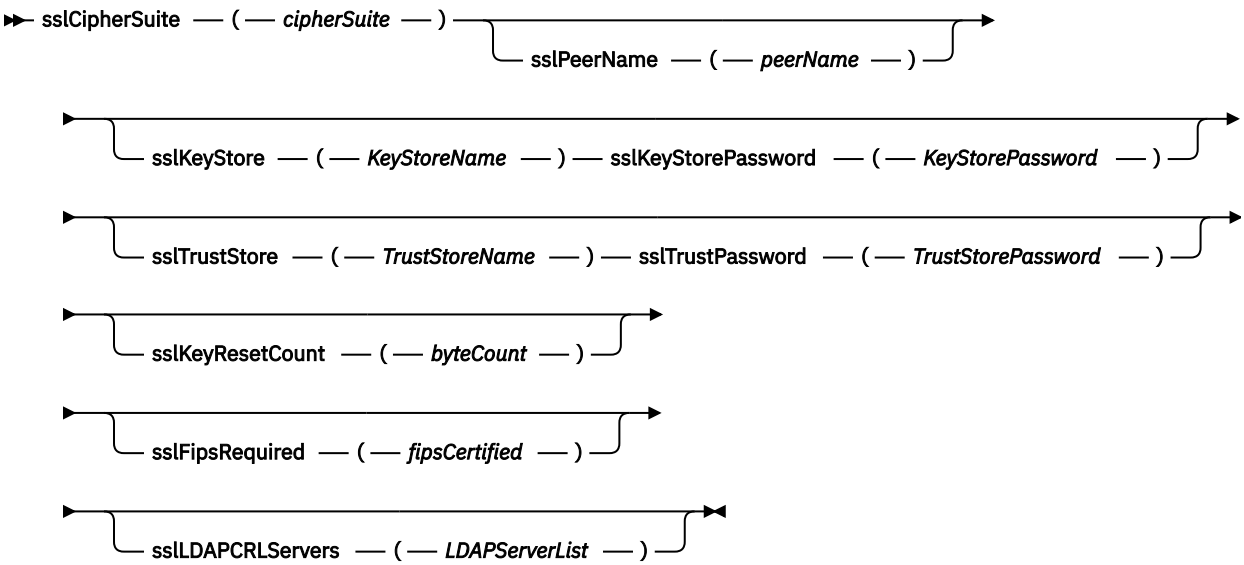
Binding



Client binding property



SSL property



Notas:

[1] **jndiConnectionFactoryName**, **jndiConnectionFactoryName** and **jndiURL** are all required parameters. **jndi-***ContextParameter* is optional.

Parámetros

connectionFactory=*connectionFactoryParameterList*

connectionFactoryParameterList son parámetros que califican cómo se conecta el cliente Axis 2 a un gestor de colas cuando la variante de destino es cola.

connectionFactory no se debe especificar con la variante de destino jndi .

Los parámetros no se pasan al servidor en el URI de solicitud.

Si se omite connectionFactory , la cola debe pertenecer a un gestor de colas predeterminado que se ejecute en el mismo servidor que el cliente Axis 2.

connectionFactoryParameterList:

binding(*bindingType*)

bindingType especifica cómo se conecta el cliente a *qMgrNombre*. El valor predeterminado es auto. *bindingType* toma los valores siguientes:

auto

El remitente intenta los siguientes tipos de conexión, en orden:

1. Si se especifican otras opciones adecuadas para una conexión de cliente, el remitente utiliza un enlace de cliente. Las otras opciones son clientConnection o clientChannel.
2. Utilice una conexión de servidor.
3. Utilice una conexión de cliente.

Utilice `binding(auto)` en el *URI* si no hay ningún gestor de colas local en el cliente SOAP. Se crea una conexión de cliente para el cliente SOAP.

client

Utilice `binding(cliente)` en el *URI* para crear una configuración de cliente para el remitente SOAP.

server

Utilice `binding(servidor)` en el *URI* para crear una configuración de servidor para el remitente SOAP. Si la conexión tiene parámetros de tipo de cliente, la conexión falla y el remitente SOAP de IBM WebSphere MQ muestra un error. Los parámetros de tipo de cliente son clientConnection, clientChannelo parámetros SSL.

xaclient

xaclient sólo es aplicable en .NET y no para clientes Java. Utilice una conexión de cliente XA.

clientChannel(*canal*)

El cliente SOAP utiliza *channel* para realizar una conexión de cliente IBM WebSphere MQ . *channel* debe coincidir con el nombre de un canal de conexión de servidor, a menos que la definición automática de canal esté habilitada en el servidor. clientChannel es un parámetro necesario, a menos que haya proporcionado una tabla de definición de conexión de cliente (CCDT).

Proporcione una CCDT en Java estableciendo `com.ibm.mq.soap.transport.jms.mqchlurl`. En .NET, establezca las variables de entorno `MQCHLLIB` y `MQCHLTAB` ; consulte “Utilice una tabla de definición de canal con el transporte SOAP de WebSphere MQ para el remitente SOAP” en la página 992.

clientConnection(*conexión*)

El cliente SOAP utiliza *conexión* para realizar una conexión de cliente IBM WebSphere MQ . El nombre de host predeterminado es localhosty el puerto predeterminado es 1414. Si la *conexión* es una dirección TCP/IP, toma uno de los tres formatos y puede tener como sufijo un número de puerto.

Los clientes JMS pueden utilizar el formato: `hostname:port` o 'escapar' los corchetes utilizando el formato %X donde X es el valor hexadecimal que representa el carácter de corchete en la página de códigos del URI. Por ejemplo, en ASCII, %28 y %29 para (y) respectivamente.

Los clientes .Net pueden utilizar los corchetes explícitamente: `hostname(port)` o utilizar el formato 'escape'.

Dirección IPv4
Por ejemplo, `192.0.2.0`.

Dirección IPv6
Por ejemplo, `2001:DB8:0:0:0:0:0:0`.

Nombre de host
Por ejemplo, `www.example.com%281687%29`, `www.example.com:1687` o `www.example.com(1687)`.

sslCipherSuite(*CipherSuite*)
CipherSuite especifica la sslCipherSuite utilizada en el canal. La CipherSuite especificada por el cliente debe coincidir con la CipherSuite especificada en el canal de conexión del servidor.

sslFipsRequired(*fipsCertified*)
fipsCertified especifica si *CipherSpec* o *CipherSuite* debe utilizar la criptografía certificada por FIPS en IBM WebSphere MQ en el canal. El efecto de establecer *fipsCertified* es el mismo que establecer el campo `FipsRequired` de la estructura **MQSCO** en una llamada `MQCONNX` .

sslKeyStore(*KeyStoreNombre*)
KeyStoreName especifica la sslKeyStoreName utilizada en el canal. El almacén de claves contiene la clave privada del cliente utilizada para autenticar el cliente en el servidor. El almacén de claves es opcional si la conexión SSL está configurada para aceptar conexiones de cliente anónimas.

sslKeyResetCount(*bytecount*)
bytecount especifica el número de bytes pasados a través de un canal SSL antes de que se deba renegociar la clave secreta SSL. Para inhabilitar la renegociación de claves SSL omita el campo o establézcalo en cero. Cero es el único valor soportado en algunos entornos, consulte Renegociación de la clave secreta en WebSphere MQ classes for Java. El efecto de establecer sslKeyResetCount es el mismo que establecer el campo `KeyResetCount` en la estructura **MQSCO** en una llamada `MQCONNX` .

sslKeyStorePassword(*KeyStoreContraseña*)
KeyStoreContraseña especifica la sslKeyStorePassword utilizada en el canal.

sslLDAPCRLServers(*LDAPServerList*)

LDAPServerList especifica una lista de servidores LDAP que se utilizarán para la comprobación de la lista de revocación de certificados.

Para las conexiones de cliente habilitadas para SSL, *LDAPServerList* es una lista de servidores LDAP que se van a utilizar para la comprobación de la lista de revocación de certificados (CRL). El certificado proporcionado por el gestor de colas se compara con uno de los servidores CRL LDAP listados; si se encuentra, la conexión falla. Cada servidor LDAP se intenta a su vez hasta que se establece la conectividad con uno de ellos. Si es imposible conectarse a alguno de los servidores, el certificado se rechaza. Una vez que se ha establecido correctamente una conexión con uno de ellos, el certificado se acepta o se rechaza en función de las CRL presentes en ese servidor LDAP.

Si *LDAPServerList* está en blanco, el certificado que pertenece al gestor de colas no se compara con una lista de revocación de certificados. Se visualiza un mensaje de error si la lista proporcionada de URI de LDAP no es válida. El efecto de establecer este campo es el mismo que el de incluir registros `MQAIR` y acceder a ellos desde una estructura **MQSCO** en un `MQCONNX`.

sslPeerName(*peerName*)
peerName especifica el sslPeerNombre utilizado en el canal.

sslTrustStore(*TrustStoreNombre*)
TrustStoreNombre especifica el sslTrustStoreName utilizado en el canal. El almacén de confianza contiene el certificado público del servidor, o su cadena de claves, para autenticar el servidor en el cliente. El almacén de confianza es opcional si se utiliza el certificado raíz de una entidad emisora de certificados para autenticar el servidor. En Java, los certificados raíz se conservan en el almacén de certificados JRE, `cacerts`.

sslTrustStorePassword(*TrustStore*)
TrustStoreContraseña especifica el sslTrustStorePassword utilizado en el canal.

CustomParameter=customParameterValue

CustomParameter es el nombre definido por el usuario de un parámetro personalizado y *customParameterValue* es el valor del parámetro.

Los parámetros personalizados que no utiliza el cliente Axis 2 los envía el cliente Axis 2 al servidor SOAP. Consulte la documentación del servidor. connectionFactory es un parámetro personalizado que utiliza el cliente Axis 2 y que no se pasa al servidor.

CustomParameter no debe coincidir con el nombre de un parámetro existente.

Si *CustomParameter* empieza por la serie `jndi-` , se utiliza en la búsqueda de un destino JNDI; consulte jndi-.

deliveryMode=*deliveryMode*
deliveryMode establece la persistencia de mensajes. El valor predeterminado es PERSISTENT.

jndi:*destinationName*
destinationName es un nombre de destino JNDI que se correlaciona con una cola JMS. Si se especifica la variante de destino jndi , debe proporcionar un *destinationName*.

jndiConnectionFactoryName=*connectionFactoryNombre*
connectionFactoryNombre establece el nombre JNDI de la fábrica de conexiones. Si la variante de destino es jndi, se debe proporcionar *connectionFactoryName* .

jndiInitialContextFactory=*contextFactoryNombre*
contextFactoryNombre establece el nombre JNDI de la fábrica de contexto inicial. Si la variante de destino es jndi, se debe proporcionar *contextFactoryNombre* . Consulte Utilización de JNDI para recuperar objetos administrados en una aplicación JMS.

jndiURL=*providerURL*
jndiURL establece el nombre de URL del proveedor JNDI. Si la variante de destino es jndi, se debe especificar *jndiURL* .

jndi-*ContextParameter=contextParameterValue*
jndi-*ContextParameter* es el nombre definido por el usuario de un parámetro personalizado que se utiliza para pasar información al proveedor JNDI. *contextParameterValue* es la información que se pasa.

priority=*priorityValue*
priorityValue establece la prioridad del mensaje JMS. `0` es bajo, 9 es alto. El valor predeterminado es 4.

queue:*queueName*
queueName es el nombre de una cola JMS en la que se coloca la solicitud SOAP. Si se especifica la variante de cola, se debe proporcionar un nombre de cola. Si la cola no pertenece a un gestor de colas predeterminado en el mismo servidor que el cliente, establezca el parámetro connectionFactory .

replyToName=*replyToDestino*
replyToDestino establece el nombre de cola de destino. Si la variante de destino es jndi, el nombre es un nombre JNDI que debe correlacionarse con una cola. Si la variante es cola , el nombre es una cola JMS. El valor predeterminado es `SYSTEM.SOAP.RESPONSE.QUEUE`.

targetService=*serviceName*
El nombre utilizado por el servidor SOAP para iniciar el servicio web de destino.

En Axis, *serviceName* es el nombre completo de un servicio Java, por ejemplo: `targetService=www.example.org.StockQuote`. Si no se especifica targetService , se carga un servicio utilizando el mecanismo Axis predeterminado.

timeToLive=*milisegundos*
Establezca *milisegundos* en el tiempo antes de que caduque el mensaje. El valor predeterminado, `0`, es el mensaje que nunca caduca.

Ejemplos

```
jms:jndi:REQUESTQ
    ?jndiURL=file:/C:/JMSAdmin
    &jndiInitialContextFactory=com.sun.jndi.fscontext.RefFSContextFactory
    &jndiConnectionFactoryName=ConnectionFactory
    &replyToName=RESPONSEQ
    &deliveryMode(NON_PERSISTENT)
```

Figura 28. Utilice jms: jndi para enviar una solicitud SOAP/JMS

```
jms:queue:SOAPJ.demos
    ?connectionFactory=connectQueueManager(QM1)
     Bind(Client)
     ClientChannel(SOAPClient)
     ClientConnection(www.example.org(1418))
    &deliveryMode(NON_PERSISTENT)
```

Figura 29. Utilice jms: queue para enviar una solicitud SOAP/JMS

Servicios web soportados

No es necesario modificar el código que se ha escrito para que se ejecute como un servicio web para utilizar el transporte IBM WebSphere MQ para SOAP. Es necesario desplegar servicios de forma diferente para ejecutarse con el transporte IBM WebSphere MQ para SOAP en lugar de utilizar HTTP.

Descripción

El transporte WebSphere MQ para SOAP proporciona un escucha SOAP para ejecutar servicios para .NET Framework 1 y .NET 2, y para Axis 1.4. El canal personalizado de WebSphere MQ para Microsoft Windows Communication Foundation ejecuta servicios para .NET Framework 3. WebSphere Application Server y CICS proporcionan soporte para ejecutar servicios a través del transporte WebSphere MQ para SOAP. Cree una exportación personalizada para utilizar WebSphere Enterprise Service Bus o WebSphere Process Server.

El escucha SOAP de WebSphere MQ puede procesar solicitudes SOAP de forma transaccional. Ejecute **amqwdeployWMQService** utilizando la opción -x . La opción de dos fases sólo está soportada para escuchas que utilizan enlaces de servidor. Otros entornos pueden proporcionar soporte transaccional para el transporte WebSphere MQ para SOAP. Consulte su documentación.

WebSphere MQ Transport para SOAP actualmente no da soporte al protocolo SOAP sobre JMS estándar del sector emergente que se ha enviado a W3C. Puede distinguir un mensaje SOAP/JMS escrito en el nuevo estándar buscando la propiedad JMS BindingVersion . El transporte WebSphere MQ para SOAP no establece la propiedad BindingVersion .

Eje 1.4

Normalmente, una clase Java se puede utilizar sin modificación. Los tipos de argumentos para los métodos del servicio web deben estar soportados por el motor Axis. Consulte la documentación de Axis para obtener más detalles. Si el servicio utiliza un objeto complejo como argumento, o devuelve uno, dicho objeto debe cumplir con la especificación del Java™ . Consulte los ejemplos de Figura 32 en la página 1008, Figura 33 en la página 1008y Figura 34 en la página 1009:

1. Tener un constructor público sin parámetros.
2. Los tipos complejos del bean deben tener métodos getter y setters públicos con el formato:

Prepare el servicio para el despliegue utilizando el programa de utilidad **amqwdeployWMQService** . El servicio lo invoca el escucha SOAP WebSphere MQ que utiliza `axis.jar` para ejecutar el servicio.

El único gestor de transacciones de dos fases soportado para Axis 1.4 es WebSphere MQ.

El programa de utilidad de despliegue proporcionado no da soporte al caso en el que un servicio devuelve un objeto de un paquete diferente al propio servicio. Para utilizar un objeto devuelto en un paquete diferente, escriba su propio programa de utilidad de despliegue. Puede basar el programa de utilidad de despliegue en el ejemplo proporcionado, o capturar los mandatos que genera, utilizando la opción -v . Modifique los mandatos para producir un script a medida.

Si el servicio utiliza clases que son externas a la infraestructura Axis y al entorno de ejecución SOAP de WebSphere MQ , debe establecer el CLASSPATHcorrecto. Para cambiar CLASSPATH, corrija el script generado que inicia o define los escuchas para incluir los servicios necesarios, de una de las maneras siguientes:

- Corrija el CLASSPATH directamente en el script después de la llamada a **amqwsetcp**.
- Cree un script específico de servicio para personalizar el CLASSPATH e invoque este script en el script generado después de la llamada a **amqwsetcp**.
- Cree un proceso de despliegue personalizado para personalizar el CLASSPATH en el script generado automáticamente.

.NET Framework 1 y .NET Framework 2

No es necesario modificar un servicio que ya se ha preparado como un servicio web HTTP para utilizarlo como un servicio web de WebSphere MQ . Debe desplegarse utilizando el programa de utilidad **amqwdeployWMQService** .

El único gestor de transacciones de dos fases soportado para .NET Framework 1 y .NET 2 es Microsoft Transaction Server (MTS).

Si el código de servicio no se ha preparado como un servicio web HTTP, debe convertirlo en un servicio web. Declare la clase como un servicio web e identifique cómo se formatean los parámetros de cada método. Debe comprobar que los argumentos para los métodos del servicio son compatibles con el entorno. Figura 30 en la página 1008 y Figura 31 en la página 1008 muestran una clase .NET que se ha preparado como un servicio web. Las adiciones realizadas se muestran en negrita.

Figura 30 en la página 1008 utiliza el modelo de programación por detrás de código para un servicio web .NET. En el modelo de detrás de código, el origen del servicio se separa del archivo `.asmx` . El archivo `.asmx` declara el nombre del archivo de origen asociado con la palabra clave Codebehind . WebSphere MQ tiene ejemplos de servicios web .NET en línea y detrás de código.

El origen de servicios web .NET debe compilarse antes del despliegue mediante el programa de utilidad de despliegue **amqwdeployWMQService** . El servicio se compila en una biblioteca (`.dll`). La biblioteca debe colocarse en el subdirectorio `./bin` del directorio de despliegue.

.NET Framework 3

Cree un canal personalizado de WebSphere MQ para Microsoft Windows Communication Foundation (WCF) para invocar servicios desplegados en .NET Framework 3. Consulte Canal personalizado deIBM WebSphere MQ para Microsoft Windows Communication Foundation (WCF) para obtener una descripción de cómo configurar WCF para utilizar el transporte WebSphere MQ para SOAP.

WebSphere Application Server

Puede invocar servicios web alojados en WebSphere Application Server utilizando WebSphere MQ Transport for SOAP; consulte Utilización de SOAP sobre JMS para transportar servicios web (en desuso).

Debe modificar el WSDL generado por el despliegue de un servicio JMS en WebSphere Application Server para generar un cliente de servicios web. El WSDL creado por el despliegue en WebSphere Application Server incluye un URI con una referencia JNDI a `InitialContextFactory`de JMS. Debe modificar la referencia JNDI a Nojndi y proporcionar atributos de conexión tal como se describe en “Sintaxis de URI y parámetros para el despliegue de servicios web” en la página 993.

CICS

Puede invocar aplicaciones CICS utilizando WebSphere MQ Transport for SOAP; consulte Configuración del sistema CICS para servicios web.

WebSphere Enterprise Service Bus y WebSphere Process Server for Multiplatforms

WebSphere ESB y WebSphere Process Server for Multiplatforms dan soporte a SOAP sobre JMS, con un enlace creado listo, sólo cuando se utiliza el proveedor de mensajería predeterminado de WebSphere Application Server. Cree un enlace personalizado para JMS para dar soporte al transporte WebSphere MQ para SOAP. Consulte Enlaces de datos JMS.

Ejemplo

```
<%@ WebService Language="C#" CodeBehind="Quote.asmx.cs" Class="Quote.QuoteDotNet" %>
```

Figura 30. Definición de servicio para .NET Framework 2: `Quote.asmx`

```
<%@ WebService Language="C#" CodeBehind="Quote.asmx.cs" Class="Quote.QuoteDotNet" %>
using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;

namespace Quote {
    [WebService(Namespace = "http://www.example.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    public class QuoteDotNet : System.Web.Services.WebService {
        [WebMethod]
        public string getQuote(String symbol){
            return symbol.ToUpper();
        }
    }
}
```

Figura 31. Implementación de servicio para .NET Framework 2: `Quote.asmx.cs`

```
package org.example.www;
public interface CustomerInfoInterface extends java.rmi.Remote {
    public org.example.www.CustomerRecord
           getCustomerName(org.example.www.CustomerRecord request)
           throws java.rmi.RemoteException, org.example.www.GetCustomerName_faultMsg;
}
```

Figura 32. Interfaz de servicio JAX-RPC de Java que utiliza un tipo complejo

```
package org.example.www;
public class CustomerInfoPortImpl implements org.example.www.CustomerInfoInterface{
    public org.example.www.CustomerRecord
           getCustomerName(org.example.www.CustomerRecord request)
           throws java.rmi.RemoteException, org.example.www.GetCustomerName_faultMsg {
        request.setName(request.getID().toString());
        return request;
    }
}
```

Figura 33. Implementación de servicio JAX-RPC de Java utilizando un tipo complejo

```
package org.example.www;
public class CustomerRecord  {
    private java.lang.String name;
    private java.lang.Integer ID;
    public CustomerRecord() {}
    public java.lang.String getName() {
        return name; }
    public void setName(java.lang.String name) {
        this.name = name; }
    public java.lang.Integer getID() {
        return ID; }
    public void setID(java.lang.Integer ID) {
        this.ID = ID; }
}
```

Figura 34. Implementación de bean de servicio JAX-RPC Java de un tipo complejo

Transporte IBM WebSphere MQ para clientes de servicio web SOAP

Puede reutilizar un cliente SOAP sobre HTTP existente con el transporte IBM WebSphere MQ para SOAP. Debe realizar algunas pequeñas modificaciones en el código y el proceso de compilación para convertir el cliente para que funcione con el transporte IBM WebSphere MQ para SOAP.

Codificación

Los clientes JAX-RPC deben estar escritos en Java. Los clientes de .NET Framework 1 y 2 se pueden escribir en cualquier lenguaje que utilice Common Language Runtime. Los ejemplos de código se proporcionan en C# y Visual Basic.

El nivel de soporte transaccional depende del entorno de cliente y del patrón de la interacción SOAP. La solicitud SOAP y la respuesta SOAP no pueden formar parte de la misma transacción atómica.

Debe llamar a `IBM.WMQSOAP.Register.Extension()` en un cliente .NET Framework 1, .NET Framework 2. En un cliente de servicio web Java JAX-RPC, llame a `com.ibm.mq.soap.Register.extension` para registrar el emisor SOAP de WebSphere MQ . El método registra el transporte de WebSphere MQ para el remitente SOAP como manejador para mensajes SOAP utilizando el protocolo `jms:` .

Para crear un cliente .NET Framework 3, genere un proxy de cliente de Windows Communication Foundation utilizando la herramienta **svcutil** ; consulte Generación de un proxy de cliente WCF y archivos de configuración de aplicación utilizando la herramienta svcutil con metadatos de un servicio en ejecución.

Bibliotecas necesarias para crear y ejecutar clientes .NET Framework 1 y 2

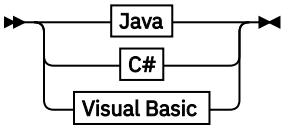
- amqsoap
- Sistema
- System.Web.Services
- System.Xml

Bibliotecas necesarias para crear y ejecutar clientes Axis 1.4

- *MQ_Install*\java\lib\com.ibm.mq.soap.jar;
- *MQ_Install*\java\lib\com.ibm.mq.commonservices.jar;
- *MQ_Install*\java\lib\soap\axis.jar;
- *MQ_Install*\java\lib\soap\jaxrpc.jar
- *MQ_Install*\java\lib\soap\saaj.jar;
- *MQ_Install*\java\lib\soap\commons-logging-1.0.4.jar;
- *MQ_Install*\java\lib\soap\commons-discovery-0.2.jar;
- *MQ_Install*\java\lib\soap\wsdl4j-1.5.1.jar;

- *MQ_Install*\java\jre\lib\xml.jar;
- *MQ_Install*\java\lib\soap\servlet.jar;
- *MQ_Install*\java\lib\com.ibm.mq.jar;
- *MQ_Install*\java\lib\com.ibm.mq.headers.jar;
- *MQ_Install*\java\lib\com.ibm.mq.pcf.jar;
- *MQ_Install*\java\lib\com.ibm.mq.jmqi.jar;
- *MQ_Install*\java\lib\com.ibm.mq.jmqi.remote.jar;
- *MQ_Install*\java\lib\com.ibm.mq.jmqi.local.jar;
- *MQ_Install*\java\lib\connector.jar;
- *MQ_Install*\java\lib\jta.jar;
- *MQ_Install*\java\lib\jndi.jar;
- *MQ_Install*\java\lib\ldap.jar

Register SOAP extension



Java

C#

Visual Basic

Ejemplos de cliente

Figura 35 en la página 1010 es un ejemplo de un cliente .NET Framework 1 o .NET Framework 2 C# que utiliza el modelo de programación en línea. El método **IBM.WMQSOAP.Register.Extension()** registra el emisor SOAP de WebSphere MQ con .NET como manejador de protocolo `jms:` .

```
using System;
namespace QuoteClientProgram {
    class QuoteMain {
        static void Main(string[] args) {
            try {
                IBM.WMQSOAP.Register.Extension();
                Quote q = new Quote();
                Console.WriteLine("Response is: " + q.getQuote("ibm"));
             }  catch (Exception e) {
                Console.WriteLine("Exception is: " + e);
             }
        }
    }
}
```

Figura 35. Ejemplo de cliente de servicio web C#

Figura 36 en la página 1011 es un ejemplo de un cliente Java que utiliza la interfaz de cliente proxy estático JAX-RPC. El método **com.ibm.mq.soap.Register.extension();** registra el remitente SOAP de WebSphere MQ con el proxy de servicio para manejar el protocolo `jms:` .

```
package org.example.www;
import com.ibm.mq.soap.Register;
public class QuoteClient {
   public static void main(String[] args) {
      try {
         Register.extension();
         QuoteSOAPImplServiceLocator locator = new QuoteSOAPImplServiceLocator();
         System.out.println("Response = "
                  + locator.getOrgExampleWwwQuoteSOAPImpl_Wmq().getQuote("IBM"));
      } catch (Exception e) {
         System.out.println("Exception = " + e.getMessage());
      }
   }
}
```

Figura 36. Ejemplo de cliente de servicio web Java

Referencia de salidas de usuarios, salidas de API y servicios instalables

Utilice los enlaces que se proporcionan en esta sección para ayudarle a desarrollar las salidas de usuario, las salidas de API y las aplicaciones de servicios instalables:

- “Estructura MQIEP” en la página 1012
- “Referencia de salida de conversión de datos” en la página 1015
- “MQ_PUBLISH_EXIT - Salida de publicación” en la página 1019
- “Llamadas de salida de canal y estructuras de datos” en la página 1027
- “Referencia a la salida de la API” en la página 1091
- “Información de consulta sobre la interfaz de servicios instalables” en la página 1152

Conceptos relacionados

“Referencia de aplicaciones MQI” en la página 7
Utilice los enlaces que se proporcionan en esta sección para ayudarle a desarrollar las aplicaciones MQI:

“Las clases IBM WebSphere MQ para bibliotecas Java” en la página 1427
La ubicación de las clases IBM WebSphere MQ para bibliotecas Java varía según la plataforma. Especifique esta ubicación cuando inicie una aplicación.

Tareas relacionadas

Desarrollo de aplicaciones

Referencia relacionada

“Referencia SOAP” en la página 954
Transporte de WebSphere MQ para información de referencia SOAP ordenada alfabéticamente.

“Material de referencia para el puente IBM WebSphere MQ para HTTP” en la página 1216
Temas de referencia para el puente IBM WebSphere MQ para HTTP, ordenados alfabéticamente

“Las clases e interfaces .NET deIBM WebSphere MQ” en la página 1252
Las clases e interfaces .NET de IBM WebSphere MQ se listan alfabéticamente. Se describen las propiedades, métodos y constructores.

“IBM WebSphere MQ clases C++” en la página 1315
Las clases C++ de IBM WebSphere MQ encapsulan la interfaz de cola de mensajes (MQI) de IBM WebSphere MQ . Hay un único archivo de cabecera C++, **imqi.hpp**, que cubre todas estas clases.

clases de WebSphere MQ para JMS

Estructura MQIEP

La estructura MQIEP contiene un punto de entrada para cada llamada de función que se permite realizar a las salidas.

Campos

StrucId
Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQIEP_STRUC_ID

Versión
Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQIEP_VERSION_1
Número de versión de estructura de la versión 1.

MQIEP_CURRENT_VERSION
Versión actual de la estructura.

StrucLength
Tipo: MQLONG

Tamaño de la estructura MQIEP en bytes. El valor es el siguiente:

MQIEP_LENGTH_1

Distintivos
Tipo: MQLONG

Proporciona información sobre las direcciones de función. Un distintivo para indicar si la biblioteca tiene hebras puede utilizarse con un distintivo para indicar si la biblioteca es una biblioteca de cliente o de servidor.

El valor siguiente se utiliza para no especificar información de biblioteca:

MQIEPF_NONE

Se utiliza uno de los valores siguientes para especificar si la biblioteca compartida es con hebras o no con hebras:

MQIEPF_NON_THREADED_LIBRARY
Una biblioteca compartida sin hebras

MQIEPF_THREADED_LIBRARY
Una biblioteca compartida con hebras

Se utiliza uno de los valores siguientes para especificar si la biblioteca compartida es un cliente o una biblioteca compartida de servidor:

MQIEPF_BIBLIOTECA_CLIENTE
Una biblioteca compartida de cliente

MQIEPF_BIBLIOTECA_LOCAL
Una biblioteca compartida de servidor

Reserved
Tipo: MQPTR

Llamada MQBACK_Call
Tipo: PMQ_BACK_CALL

Dirección de la llamada MQBACK.

MQBEGIN_Llamada
Tipo: PMQ_BEGIN_CALL

Dirección de la llamada MQBEGIN.

MQBUFMH_Llamada
Tipo: PMQ_BUFMH_CALL

Dirección de la llamada MQBUFMH.

MQCB_Llamada
Tipo: PMQ_CB_CALL

Dirección de la llamada MQCB.

Llamada MQCLOSE_Call
Tipo: PMQ_CLOSE_CALL

Dirección de la llamada MQCLOSE.

MQCMIT_Llamada
Tipo: PMQ_CMIT_CALL

Dirección de la llamada MQCMIT.

MQCONN_Llamada
Tipo: PMQ_CONN_CALL

Dirección de la llamada MQCONN.

MQCONNX_Llamada
Tipo: PMQ_CONNX_CALL

Dirección de la llamada MQCONNX.

MQCRTMH_Llamada
Tipo: PMQ_CRTMH_CALL

Dirección de la llamada MQCRTMH.

MQCTL_Llamada
Tipo: PMQ_CTL_CALL

Dirección de la llamada MQCTL.

MQDISC_Llamada
Tipo: PMQ_DISC_CALL

Dirección de la llamada MQDISC.

Llamada MQDLTMH_Call
Tipo: PMQ_DLTMH_CALL

Dirección de la llamada MQDLTMH.

Llamada MQDLTMP_Call
Tipo: PMQ_DLTMP_CALL

Dirección de la llamada MQDLTMP.

Llamada MQGET_Call
Tipo: PMQ_GET_CALL

Dirección de la llamada MQGET.

MQINQ_Llamada
Tipo: PMQ_INQ_CALL

Dirección de la llamada MQINQ.

MQINQMP_Llamada
Tipo: PMQ_INQMP_CALL

Dirección de la llamada MQINQMP.

Llamada MQMHBUF_Call
Tipo: PMQ_MHBUF_CALL

Dirección de la llamada MQMHBUF.

MQOPEN_Llamada
Tipo: PMQ_OPEN_CALL

Dirección de la llamada MQOPEN.

Llamada MQPUT_Call
Tipo: PMQ_PUT_CALL

Dirección de la llamada MQPUT.

MQPUT1_Call
Tipo: PMQ_PUT1_CALL

Dirección de la llamada MQPUT1 .

MQSET_Call
Tipo: PMQ_SET_CALL

Dirección de la llamada MQSET.

Llamada MQSETMP_Call
Tipo: PMQ_SETMP_CALL

Dirección de la llamada MQSETMP.

MQSTAT_Llamada
Tipo: PMQ_STAT_CALL

Dirección de la llamada MQSTAT.

MQSUB_Llamada
Tipo: PMQ_SUB_CALL

Dirección de la llamada MQSUB.

MQSUBRQ_Llamada
Tipo: PMQ_SUBRQ_CALL

Dirección de la llamada MQSUBRQ.

MQXCNVC_Llamada
Tipo: PMQ_XCNVC_CALL

Dirección de la llamada MQXCNVC.

MQXCLWLN_Llamada
Tipo: PMQ_XCLWLN_CALL

Dirección de la llamada MQXCLWLN.

MQXDX_Llamada
Tipo: PMQ_XDX_CALL

Dirección de la llamada MQXDX.

MQXEP_Llamada
Tipo: PMQ_XEP_CALL

Dirección de la llamada MQXEP.

MQZEP_Llamada
Tipo: PMQ_ZEP_CALL

Dirección de la llamada MQZEP.

Declaración 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 */
 };
```

Referencia de salida de conversión de datos

Para z/OS, debe escribir salidas de conversión de datos en lenguaje ensamblador. Para otras plataformas, se recomienda utilizar el lenguaje de programación C.

Para ayudarle a crear un programa de salida de conversión de datos, se proporciona lo siguiente:

- Un archivo de origen de esqueleto
- Una llamada de conversión de caracteres
- Un programa de utilidad que crea un fragmento de código que realiza la conversión de datos en estructuras de tipo de datos. Este programa de utilidad sólo toma la entrada C. En z/OS, genera código de ensamblador.

Para el procedimiento para escribir los programas, consulte:

- Escritura de una salida de conversión de datos para WebSphere MQ en sistemas UNIX and Linux
- Escritura de una salida de conversión de datos para WebSphere MQ para Windows

Archivo de origen de esqueleto

Estos se pueden utilizar como punto de partida al escribir un programa de salida de conversión de datos.

Los archivos suministrados se listan en Tabla 588 en la página 1016.

Tabla 588. Archivos de origen de esqueleto	
Plataforma	**Archivo**
AIX	amqsvfc0.c
IBM i	QMQMSAMP/QCSRC (AMQSVFC4)
HP-UX	amqsvfc0.c
Linux	amqsvfc0.c
z/OS	CSQ4BAX8 (“1” en la página 1016) CSQ4BAX9 (“2” en la página 1016) CSQ4CAX9 (“3” en la página 1016)
Solaris	amqsvfc0.c
sistemas Windows	amqsvfc0.c
Notas: 1. Ilustra la llamada MQXCVNC. 2. Un derivador para los fragmentos de código generados por el programa de utilidad para su uso en todos los entornos excepto CICS. 3. Un derivador para los fragmentos de código generados por el programa de utilidad para su uso en el entorno CICS .	

Llamada de conversión de caracteres

Utilice la llamada MQXCNVC (convertir caracteres) desde dentro de un programa de salida de conversión de datos para convertir datos de mensajes de caracteres de un juego de caracteres a otro. Para determinados juegos de caracteres multibyte (por ejemplo, juegos de caracteres UCS2), se deben utilizar las opciones adecuadas.

No se pueden realizar otras llamadas MQI desde dentro de la salida; un intento de realizar una llamada de este tipo falla con el código de razón MQRC_CALL_IN_PROGRESS.

Consulte “MQXCNVC-Convertir caracteres” en la página 901 para obtener más información sobre la llamada MQXCNVC y las opciones adecuadas.

Utilidad para crear código de salida de conversión

Utilice esta información para obtener más información sobre cómo crear código de salida de conversión.

Los mandatos para crear el código de salida de conversión son:

IBM i
CVTMQMDTA (Convertir tipo de datos WebSphere MQ)

Windows, sistemas UNIX and Linux
`crtmqcvx` (Crear WebSphere MQ conversion-exit)

El mandato para la plataforma produce un fragmento de código que realiza la conversión de datos en estructuras de tipo de datos, para su uso en el programa de salida de conversión de datos. El mandato toma un archivo que contiene una o más definiciones de estructura de lenguaje C. .

Mensajes de error en sistemas Windows, UNIX and Linux

El mandato `crtmqcvx` devuelve mensajes en el rango de AMQ7953 a AMQ7970.

Estos mensajes se listan en Códigos de razón *WebSphere MQ Mensajes*.

Hay dos tipos principales de error:

- Errores graves, como errores de sintaxis, cuando el proceso no puede continuar.

  Se visualiza un mensaje en la pantalla que indica el número de línea del error en el archivo de entrada. Es posible que el archivo de salida se haya creado parcialmente.
- Otros errores cuando se visualiza un mensaje que indica que se ha encontrado un problema pero que el análisis de la estructura puede continuar.

  El archivo de salida se ha creado y contiene información de error sobre los problemas que se han producido. Esta información de error tiene el prefijo `#error` para que el código generado no sea aceptado por ningún compilador sin intervención para rectificar los problemas.

Sintaxis válida

El archivo de entrada para el programa de utilidad debe ajustarse a la sintaxis del lenguaje C.

Si no está familiarizado con C, consulte el ejemplo C de este tema.

Además, tenga en cuenta las reglas siguientes:

- typedef sólo se reconoce antes de la palabra clave struct.
- Se necesita un código de estructura en las declaraciones de estructura.
- Puede utilizar corchetes vacíos [] para indicar una serie o matriz de longitud variable al final de un mensaje.
- Las matrices multidimensionales y las matrices de series no están soportadas.
- Se reconocen los siguientes tipos de datos adicionales:
  - MQBOOL
  - MQBYTE
  - MQCHAR
  - MQFLOAT32
  - MQFLOAT64
  - MQSHORT
  - MQLONG
  - MQINT8
  - MQUINT8
  - MQINT16
  - MQUINT16
  - MQINT32
  - MQUINT32
  - MQINT64
  - MQUINT64

  Los campos MQCHAR se convierten en página de códigos, pero MQBYTE, MQINT8 y MQUINT8 se dejan intactos. Si la codificación es diferente, MQSHORT, MQLONG, MQINT16, MQUINT16, MQINT32, MQUINT32, MQINT64, MQUINT64, MQFLOAT32, MQFLOAT64 y MQBOOL se convierten en consecuencia.
- *No* utilice los siguientes tipos de datos:
  - doble

- Punteros
- campos de bits

  Esto se debe a que el programa de utilidad para crear el código de salida de conversión no proporciona el recurso para convertir estos tipos de datos. Para superar esto, puedes escribir tus propias rutinas y llamarlas desde la salida.

Otros puntos a tener en cuenta:

- No utilice números de secuencia en el conjunto de datos de entrada.
- Si hay campos para los que desea proporcionar sus propias rutinas de conversión, declárelos como MQBYTE y, a continuación, sustituya las macros CMQXCFBA generadas por su propio código de conversión.

Ejemplo de C

```
  struct TEST { MQLONG    SERIAL_NUMBER;
                MQCHAR    ID[5];
                MQINT16   VERSION;
                MQBYTE    CODE[4];
                MQLONG    DIMENSIONS[3];
                MQCHAR    NAME[24];
              } ;
```

Esto corresponde a las siguientes declaraciones en otros lenguajes de programación:

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

Soportado solo en 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 - Salida de publicación

La llamada MQ_PUBLISH_EXIT puede inspeccionar y modificar los mensajes entregados a los suscriptores.

Finalidad

Utilice la salida de publicación para inspeccionar y modificar los mensajes entregados a los suscriptores:

- Examinar el contenido de un mensaje publicado en cada suscriptor.
- Modificar el contenido de un mensaje publicado en cada suscriptor.
- Modificar la cola en la que se coloca el mensaje.
- Detener la entrega de un mensaje a un suscriptor.

Sintaxis

MQ_PUBLISH_EXIT(*ExitParms, PubContext, SubContext*)

Parámetros

***ExitParms* (MQPSXP)- Input/Output**
ExitParms contiene información sobre la invocación de la salida.

***PubContext* (MQPBC) - Input**
PubContext contiene información contextual sobre el editor de la publicación.

***SubContext* (MQSBC) - Input/Output**
SubContext contiene información contextual sobre el suscriptor que recibe la publicación.

MQPSXP-Estructura de datos de salida de publicación

La estructura MQPSXP describe la información que se pasa a y se devuelve de la salida de publicación.

Tabla 589 en la página 1019 resume los campos de la estructura:

Tabla 589. Campos en MQPSXP

Campo	Descripción
StrucID	Identificador de la estructura
Version	Número de versión de la estructura
ExitId	Tipo de salida que se está llamando
ExitReason	Razón para llamar a la salida
ExitResponse	Respuesta de la salida
ExitResponse2	Respuesta secundaria de salida
Feedback	Código de retroalimentación
ExitUserArea	Salir del área de usuario
ExitData	Datos de salida
QMgrName	Nombre del gestor de colas local
Hconn	Descriptor de contexto de conexión
MsgDescPtr	Dirección del descriptor de mensaje (MQMD)
MsgHandle	Descriptor de contexto para propiedades de mensaje (MQHMSG)
MsgInPtr	Dirección del mensaje de entrada

Tabla 589. Campos en MQPSXP (continuación)	
Campo	**Descripción**
MsgInLength	Longitud del mensaje de entrada
MsgOutPtr	Dirección del mensaje de salida
MsgOutLength	Longitud del mensaje de salida
pEntryPoints	Dirección de la estructura MQIEP

Campos

***StrucID* (MQCHAR4)**
StrucID es el identificador de estructura. El valor es el siguiente:

MQPSXP_STRUCID
MQPSXP_STRUCID es el identificador de la estructura de parámetros de salida de publicación. Para el lenguaje de programación C, la constante MQPSXP_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQPSXP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

StrucID es un campo de entrada para la salida.

***Version* (MQLONG)**
Version es el número de versión de la estructura. El valor es el siguiente:

MQPSXP_VERSION_1
MQPSXP_VERSION_1 es la estructura del parámetro de salida de publicación de la versión 1. La constante MQPSXP_CURRENT_VERSION también se define con el mismo valor.

Version es un campo de entrada para la salida.

***ExitId* (MQLONG)**
ExitId es el tipo de salida que se está llamando. El valor es el siguiente:

MQXT_PUBLISH_EXIT
Salida de publicación.

ExitId es un campo de entrada para la salida.

***ExitReason* (MQLONG)**
ExitReason es la razón por la que se llama a la salida. Los valores posibles son:

MQXR_INIT
Se llama a la salida para esta conexión para la inicialización. La salida puede adquirir e inicializar los recursos que necesita; por ejemplo, el almacenamiento principal.

MQXR_TERM
Se llama a la salida para esta conexión porque la salida está a punto de detenerse. La salida debe liberar los recursos que haya adquirido desde que se inicializó; por ejemplo, el almacenamiento principal.

MQXR_PUBLICATION
El gestor de colas llama a la salida antes de colocar una publicación en una cola de mensajes de un suscriptor. La salida puede cambiar el mensaje, no colocar el mensaje en la cola o detener la publicación.

ExitReason es un campo de entrada para la salida.

***ExitResponse* (MQLONG)**
Establezca *ExitResponse* en la salida para especificar cómo debe continuar el proceso. *ExitResponse* es uno de los valores siguientes:

MQXCC_OK
Establezca MQXCC_OK para continuar procesando normalmente. Establezca MQXCC_OK en respuesta a cualquier valor de ExitReason.

Si ExitReason tiene el valor MQXR_PUBLICATION , los campos *DestinationQName* y *DestinationQMgrName* de la estructura MQSBC identifican el destino al que se envía el mensaje.

MQXCC_FAILED
Establezca MQXCC_FAILED para detener la operación de publicación. El código de terminación MQCC_FAILED y el código de razón 2557 (09FD) (RC2557): MQRC_PUBLISH_EXIT_ERROR se establece en el retorno de la salida.

MQXCC_SUPPRESS_FUNCTION
Establezca MQXCC_SUPPRESS_FUNCTION para detener el proceso normal del mensaje. Solo establezca MQXCC_SUPPRESS_FUNCTION si *ExitReason* tiene el valor MQXR_PUBLICATION .

El gestor de colas sigue procesando el mensaje de acuerdo con la opción MQRO_DISCARD_MSG del campo *Report* del descriptor de mensaje del mensaje.

- Si se especifica la opción MQRO_DISCARD_MSG , el mensaje no se entrega al suscriptor.
- Si no se especifica la opción MQRO_DISCARD_MSG , el mensaje se coloca en la cola de mensajes no entregados. Si no hay ninguna cola de mensajes no entregados o el mensaje no se puede colocar correctamente en la cola de mensajes no entregados, la publicación no se entrega al suscriptor. La entrega de la publicación a otros suscriptores depende de los valores de los atributos de objeto de tema PMSGDLV y NPMSGDLV . Para obtener una explicación de estos atributos, consulte las descripciones de los parámetros para el mandato DEFINE TOPIC .

ExitResponse es un campo de salida de la salida.

***ExitResponse2* (MQLONG)**
ExitResponse2 está reservado para uso futuro.

***Feedback* (MQLONG)**
Feedback es el código de comentarios que se debe utilizar si la salida devuelve MQXCC_SUPPRESS_FUNCTION en *ExitResponse*.

En la entrada a la salida, *Feedback* siempre tiene el valor MQFB_NONE. Si la salida devuelve MQXCC_SUPPRESS_FUNCTION, establezca *Feedback* en el valor que se utilizará para el mensaje cuando el gestor de colas lo coloque en la cola de mensajes no entregados. Al volver de la salida, si *Feedback* tiene el valor original MQFB_NONE, el gestor de colas establece *Feedback* en MQFB_STOPPED_BY_PUBSUB_EXIT.

Feedback es un campo de entrada/salida para la salida.

***ExitUserArea* (MQBYTE16)**
ExitUserArea es un campo que está disponible para que lo utilice la salida. Cada conexión tiene un *ExitUserArea*independiente. La longitud de *ExitUserArea* la proporciona MQ_EXIT_USER_AREA_LENGTH .

El campo *ExitReason* tiene el valor MQXR_INIT en la primera invocación de la salida. *ExitUserArea* se inicializa en MQXUA_NONE en la primera invocación de la salida para una conexión. Los cambios posteriores en *ExitUserArea* se conservan entre invocaciones de la salida.

ExitUserArea es un campo de entrada/salida para la salida.

***ExitData* (MQCHAR32)**
ExitData son datos de salida fijos definidos por el parámetro *PublishExitData* de la stanza en el archivo de inicialización del gestor de colas. Los datos se rellenan con espacios en blanco hasta la longitud completa del campo. Si no hay datos de salida fijos definidos en el archivo de inicialización, *ExitData* está en blanco. La longitud de *ExitData* la proporciona MQ_EXIT_DATA_LENGTH.

ExitData es un campo de entrada para la salida.

***QMgrName* (MQCHAR48)**
QMgrName es el nombre del gestor de colas local. El nombre se rellena con espacios en blanco hasta la longitud completa del campo. La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH .

QMgrName es un campo de entrada para la salida.

***Hconn* (MQHCONN)**
Hconn es el descriptor de contexto que representa una conexión con el gestor de colas. Utilice sólo *Hconn* como parámetro para las llamadas a la función de propiedad de mensaje MQSETMP, MQINQMMPo MQDLTMP para trabajar con propiedades de mensaje.

Hconn es un campo de entrada para la salida.

***MsgDescPtr* (PMQMD)**
MsgDescPtr es la dirección del descriptor de mensaje (MQMD) del mensaje que se está procesando y es una copia del MQMD devuelto por la llamada MQPUT. La salida puede cambiar el contenido del descriptor de mensaje. Cualquier cambio en el contenido del descriptor de mensaje debe realizarse con cuidado. En concreto, en el caso en el que el campo *SubType* de la estructura MQSBC es de valor MQSUBTYPE_PROXY, el campo *CorrelId* del descriptor de mensaje no debe cambiarse.

No se pasa ningún descriptor de mensaje a la salida si *ExitReason* es MQXR_INIT o MQXR_TERM ; en estos casos, *MsgDescPtr* es el puntero nulo.

MsgDescPtr es un campo de entrada para la salida.

***MsgHandle* (MQHMSG)**
MsgHandle es el descriptor de contexto de las propiedades de mensaje. Utilice sólo *MsgHandle* con las llamadas de función de propiedad de mensaje MQSETMP, MQINQMMP o MQDLTMP para trabajar con propiedades de mensaje.

MsgHandle es un campo de entrada para la salida.

***MsgInPtr* (PMQVOID)**
MsgInPtr es la dirección de los datos del mensaje de entrada. El contenido del almacenamiento intermedio direccionado por *MsgInPtr* puede ser modificado por la salida; consulte *MsgOutPtr* .

MsgInPtr es un campo de entrada para la salida.

***MsgInLength* (MQLONG)**
MsgInLength es la longitud en bytes de los datos de mensaje pasados a la salida. La dirección de los datos la proporciona *MsgInPtr*.

MsgInLength es un campo de entrada para la salida.

***MsgOutPtr* (PMQVOID)**
MsgOutPtr es la dirección de un almacenamiento intermedio que contiene datos de mensaje que se devuelven de la salida. En la entrada a la salida, *MsgOutPtr* es nulo. Al volver de la salida, si el valor sigue siendo nulo, el gestor de colas envía el mensaje especificado por *MsgInPtr* , con la longitud proporcionada por *MsgInLength* .

Si la salida modifica los datos del mensaje, utilice uno de los procedimientos siguientes:

- Si la longitud de los datos no cambia, los datos se pueden modificar en el almacenamiento intermedio direccionado por *MsgInPtr* . En este caso, no cambie *MsgOutPtr* y *MsgOutLength*.
- Si los datos modificados son más cortos que los datos originales, los datos se pueden modificar en el almacenamiento intermedio direccionado por *MsgInPtr* . En este caso, *MsgOutPtr* debe establecerse en la dirección del almacenamiento intermedio de mensajes de entrada y *MsgOutLength* debe establecerse en la nueva longitud de los datos del mensaje.
- Si los datos modificados son, o pueden ser, más largos que los datos originales, la salida debe obtener un nuevo almacenamiento intermedio de mensajes. Copie los datos modificados en él. Establezca *MsgOutPtr* en la dirección del nuevo almacenamiento intermedio y establezca *MsgOutLength* en la longitud de los nuevos datos de mensaje. La salida es responsable de

liberar el almacenamiento intermedio direccionado por *MsgOutPtr* cuando se llama a la salida a continuación.

Nota: *MsgOutPtr* es siempre el puntero nulo en la entrada a la salida, y no la dirección de un almacenamiento intermedio de mensajes obtenido anteriormente. Para liberar el almacenamiento intermedio obtenido anteriormente, la salida debe guardar su dirección y longitud. Guarde la información en *ExitUserArea*o en un bloque de control que tenga su dirección guardada en *ExitUserArea* .

MsgOutPtr es un campo de entrada/salida para la salida.

***MsgOutLength* (MQLONG)**
MsgOutLength es la longitud en bytes de los datos de mensaje devueltos por la salida. En la entrada a la salida, este campo es siempre cero. Al volver de la salida, este campo se ignora si *MsgOutPtr* es nulo. Consulte *MsgOutPtr* para obtener información sobre cómo modificar los datos del mensaje.

MsgOutLength es un campo de entrada/salida para la salida.

***pEntryPoints* (PMQIEP)**
pEntryPoints es la dirección de una estructura MQIEP a través de la cual se pueden realizar llamadas MQI y DCI.

Declaración de lenguaje 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-Estructura de datos de contexto de publicación

La estructura MQPBC contiene la información contextual, relacionada con el publicador de la publicación, que se pasa a la salida de publicación.

Tabla 590 en la página 1023 resume los campos de la estructura:

Tabla 590. Campos en MQPBC

Campo	Descripción
StrucID	Identificador de la estructura
Version	Número de versión de la estructura
PubTopicString	Serie de tema de publicación
MsgDescPtr	Dirección del descriptor de mensaje (MQMD)

Campos

***StrucID* (MQCHAR4)**
StrucID es el identificador de estructura. El valor es el siguiente:

MQPBC_STRUCID
MQPBC_STRUCID es el identificador de la estructura de contexto de publicación. Para el lenguaje de programación C, la constante MQPBC_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQPBC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

StrucID es un campo de entrada para la salida.

***Version* (MQLONG)**
Version es el número de versión de la estructura. El valor es el siguiente:

MQPBC_VERSION_1
MQPBC_VERSION_1 es la estructura del parámetro de salida de publicación de la versión 1.

MQPBC_VERSION_2
MQPBC_VERSION_2 es la estructura de parámetros de salida de publicación de la versión 2. La constante MQPBC_CURRENT_VERSION también se define con el mismo valor.

Version es un campo de entrada para la salida.

***PubTopicString* (MQCHARV)**
PubTopicString es la serie de tema en la que se publica.

PubTopicString es un campo de entrada para la salida.

***MsgDescPtr* (PMQMD)**
MsgDescPtr es la dirección de una copia del descriptor de mensaje (MQMD) para el mensaje que se está procesando.

MsgDescPtr es un campo de entrada para la salida.

Declaración de lenguaje 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-Estructura de datos de contexto de suscripción

La estructura MQSBC contiene la información contextual, relacionada con el suscriptor que recibe la publicación, que se pasa a la salida de publicación.

Tabla 591 en la página 1024 resume los campos de la estructura:

Tabla 591. Entrada de campos MQSBC

Campo	Descripción
StrucID	Identificador de la estructura
Version	Número de versión de la estructura
DestinationQMgrName	Nombre del gestor de colas de destino
DestinationQName	Nombre de la cola de destino
SubType	Tipo de suscripción
SubOptions	Opciones de suscripción
ObjectName	Nombre de objeto

Tabla 591. Entrada de campos MQSBC (continuación)	
Campo	**Descripción**
ObjectString	Serie de objeto
SubTopicString	Serie de tema de suscripción
SubName	Nombre de suscripción
SubId	Identificador de suscripción
SelectionString	Dirección de la serie de selección
SubLevel	Nivel de suscripción
PSProperties	Propiedades de publicación/suscripción

Campos

***StrucID* (MQCHAR4)**
Identificador de estructura. El valor es el siguiente:

MQSBC_STRUCID
MQSBC_STRUCID es el identificador de la estructura de parámetros de salida de publicación. Para el lenguaje de programación C, la constante MQSBC_STRUC_ID_ARRAY también está definida; MQSBC_STRUC_ID_ARRAY tiene el mismo valor que MQSBC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

StrucID es un campo de entrada para la salida.

***Version* (MQLONG)**
Número de versión de la estructura. El valor es el siguiente:

MQSBC_VERSION_1
Estructura de parámetros de salida de publicación de la versión 1. La constante MQSBC_CURRENT_VERSION también se define con el mismo valor.

Version es un campo de entrada para la salida.

***DestinationQMgrName* (MQCHAR48)**
DestinationQMgrName es el nombre del gestor de colas al que se envía el mensaje. El nombre se rellena con espacios en blanco hasta la longitud completa del campo. La salida puede modificar el nombre. La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH.

DestinationQMgrName es un campo de entrada/salida para la salida; consulte la nota.

***DestinationQName* (MQCHAR48)**
DestinationQName es el nombre de la cola a la que se envía el mensaje. El nombre se rellena con espacios en blanco hasta la longitud completa del campo. La salida puede modificar el nombre. La longitud de este campo la proporciona MQ_Q_NAME_LENGTH.

DestinationQName es un campo de entrada/salida para la salida; consulte la nota.

***SubType* (MQLONG)**
SubType indica cómo se ha creado la suscripción. Los valores válidos son MQSUBTYPE_API, MQSUBTYPE_ADMIN y MQSUBTYPE_PROXY; consulte Consultar estado de suscripción (respuesta).

SubType es un campo de entrada para la salida.

***SubOptions* (MQLONG)**
SubOptions son las opciones de suscripción; consulte “Opciones (MQLONG)” en la página 545 para obtener una descripción de los valores que puede tomar este campo.

SubOptions es un campo de entrada para la salida.

***ObjectName* (MQCHAR48)**
ObjectName es el nombre del objeto de tema tal como se define en el gestor de colas local. La longitud de este campo la proporciona MQ_TOPIC_NAME_LENGTH. El nombre de objeto es el nombre del objeto de tema administrativo que el gestor de colas ha asociado con la serie de tema. Incluso si el suscriptor ha proporcionado un objeto de tema como parte de la suscripción, el *ObjectName* puede ser un objeto de tema diferente. La asociación de un objeto de tema con una suscripción depende de la resolución completa de *SubTopicString*.

ObjectName es un campo de entrada para la salida.

***ObjectString* (MQCHARV)**
ObjectString es la serie de tema completa de la publicación a la que se ha suscrito. Los comodines de la serie de suscripción original se resuelven. Es diferente al campo MQSD subscription *ObjectString* descrito en “ObjectString (MQCHARV)” en la página 545, que puede contener comodines, y es exclusivo de cualquier nombre de objeto proporcionado por el suscriptor.

ObjectString es un campo de entrada para la salida.

***SubTopicString* (MQCHARV)**
SubTopicString es la serie de tema completa tal como la proporciona el suscriptor. *SubTopicString* es la combinación de la serie de tema definida en un objeto de tema y una serie de tema. Un suscriptor debe proporcionar un objeto de tema, una serie de tema o ambos. Si el suscriptor proporciona una serie de tema, puede contener comodines.

SubTopicString es un campo de entrada para la salida.

***SubName* (MQCHARV)**
SubName es el nombre de suscripción proporcionado por el suscriptor o es un nombre generado.

SubName es un campo de entrada para la salida.

***SubId* (MQBYTE 24)**
SubId es el identificador de suscripción interno exclusivo.

SubId es un campo de entrada para la salida.

***SelectionString* (MQCHARV)**
SelectionString es el criterio de selección utilizado al suscribirse a mensajes de un tema; consulteSelectores .

SelectionString es un campo de entrada para la salida.

***SubLevel* (MQLONG)**
SubLevel es el nivel de intercepción asociado a la suscripción; consulte “SubLevel (MQLONG)” en la página 556 para obtener más detalles.

SubLevel es un campo de entrada para la salida.

***PSProperties* (MQLONG)**
PSProperties son las propiedades de publicación/suscripción. Especifican cómo se añaden las propiedades de mensaje relacionadas con la publicación/suscripción a los mensajes enviados a esta suscripción. Los valores posibles son MQPSPROP_NONE, MQPSPROP_COMPAT, MQPSPROP_RFH2, MQPSPROP_MSGPROP. Consulte Parámetros opcionales (Cambiar, Copiar y Crear suscripción) para obtener una descripción de estos valores.

PSProperties es un campo de entrada para la salida.

Nota: Las comprobaciones de autorización sólo se realizan en los valores originales de *DestinationQMgrName* y *DestinationQName* antes de que se pasen a la salida de publicación. No se realizan nuevas comprobaciones de autorización cuando la salida cambia la cola de destino, ya sea cambiando *DestinationQMgrName* o *DestinationQName*.

Declaración de lenguaje 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;
```

Llamadas de salida de canal y estructuras de datos

Esta colección de temas proporciona información de referencia sobre las llamadas y estructuras de datos especiales de WebSphere MQ que puede utilizar al escribir programas de salida de canal.

Esta información es información de interfaz de programación sensible al producto. Puede escribir salidas de usuario de WebSphere MQ en los siguientes lenguajes de programación:

Plataforma	Lenguajes de programación
WebSphere MQ para z/OS	Assembler y C (que deben ajustarse al entorno de programación del sistema C para salidas del sistema, descrito en la publicación *z/OS C/C++ Programming Guide*.)
WebSphere MQ para IBM i	ILE C, ILE COBOL e ILE RPG
Todas las demás plataformas WebSphere MQ	C

También puede escribir salidas de usuario en Java para utilizarlas únicamente con aplicaciones Java y JMS. Para obtener más información sobre cómo crear y utilizar salidas de canal con las clases WebSphere MQ para Java, consulte Utilización de salidas de canal en clases WebSphere MQ para Java y para clases WebSphere MQ para JMS, consulte Utilización de salidas de canal con clases WebSphere MQ para JMS.

No puede escribir salidas de usuario de WebSphere MQ en TAL o Visual Basic. Sin embargo, se proporciona una declaración para la estructura MQCD en Visual Basic para utilizarla en la llamada MQCONNX desde un programa cliente MQI de WebSphere MQ .

En una serie de casos en las descripciones siguientes, los parámetros son matrices o series de caracteres con un tamaño que no es fijo. Para estos parámetros, se utiliza una "n" en minúsculas para representar una constante numérica. Cuando la declaración para ese parámetro está codificada, "n" debe sustituirse por el valor numérico necesario. Para obtener más información sobre los convenios utilizados en estas descripciones, consulte la publicación “Tipos de datos elementales” en la página 218.

archivos de definición de datos

Los archivos de definición de datos se proporcionan con WebSphere MQ para cada uno de los lenguajes de programación soportados. Para obtener detalles de estos archivos, consulte Copiar, cabecera, incluir y archivos de módulo.

MQ_CHANNEL_EXIT-Salida de canal

La llamada MQ_CHANNEL_EXIT describe los parámetros que se pasan a cada una de las salidas de canal llamadas por el agente de canal de mensajes.

El gestor de colas no proporciona ningún punto de entrada denominado MQ_CHANNEL_EXIT; el nombre MQ_CHANNEL_EXIT no tiene ninguna significación especial puesto que los nombres de las salidas de canal se proporcionan en la definición de canal MQCD.

Hay cinco tipos de salida de canal:

- Salida de seguridad de canal
- Salida de mensajes de canal
- Salida de envío de canal
- Salida de recepción de canal
- Salida de reintento de mensajes de canal

Los parámetros son similares para cada tipo de salida, y la descripción que se proporciona aquí se aplica a todos ellos, excepto cuando se indique específicamente.

Sintaxis

MQ_CHANNEL_EXIT *(ChannelExitParms, ChannelDefinition, DataLength, AgentBufferLength, AgentBuffer, ExitBufferLength, ExitBufferAddr)*

Parámetros

La llamada MQ_CHANNEL_EXIT tiene los parámetros siguientes.

Parámetros de ChannelExit(MQCXP)-entrada/salida

Bloque de parámetros de salida de canal.

Esta estructura contiene información adicional relacionada con la invocación de la salida.La salida establece información en esta estructura para indicar cómo procede el MCA.

ChannelDefinition (MQCD)-entrada/salida

Definición de canal.

Esta estructura contiene parámetros establecidos por el administrador para controlar el comportamiento del canal.

DataLength (MQLONG)-entrada/salida

Longitud de los datos.

Los datos dependen del tipo de salida:

- Para una salida de seguridad de canal, cuando se invoca la salida, este parámetro contiene la longitud de cualquier mensaje de seguridad en el campo *AgentBuffer* , si *ExitReason* es MQXR_SEC_MSG. Es cero si no hay ningún mensaje. La salida debe establecer este campo en la longitud de cualquier mensaje de seguridad que se vaya a enviar a su asociado si establece *ExitResponse* en MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG. Los datos del mensaje se encuentran en *AgentBuffer* o *ExitBufferAddr*.

  El contenido de los mensajes de seguridad es responsabilidad exclusiva de las salidas de seguridad.
- Para una salida de mensajes de canal, cuando se invoca la salida, este parámetro contiene la longitud del mensaje (incluida la cabecera de cola de transmisión). La salida debe establecer este campo en la longitud del mensaje en *AgentBuffer* o *ExitBufferAddr* que va a continuar. Debe ser mayor o igual que la longitud de la cabecera de cola de transmisión (MQXQH).
- Para una salida de emisión o recepción de canal, cuando se invoca la salida, este parámetro contiene la longitud de la transmisión. La salida debe establecer este campo en la longitud de la transmisión en *AgentBuffer* o *ExitBufferAddr* que va a continuar.

Si una salida de seguridad envía un mensaje, y no hay ninguna salida de seguridad en el otro extremo del canal, o el otro extremo establece un *ExitResponse* de MQXCC_OK, la salida de inicio se vuelve a invocar con MQXR_SEC_MSG y una respuesta nula (*DataLength*= 0).

AgentBufferLongitud (MQLONG)-entrada

Longitud del almacenamiento intermedio del agente.

Este parámetro puede ser mayor que *DataLength* en la invocación.

Para las salidas de envío y recepción de mensajes de canal, la salida puede utilizar cualquier espacio no utilizado en la invocación para expandir los datos en su lugar. Si se hace esto, la salida debe establecer correctamente el parámetro *DataLength* .

En el lenguaje de programación C, este parámetro se pasa por dirección.

AgentBuffer (MQBYTE ×AgentBufferLongitud)-entrada/salida

Almacenamiento intermedio de agente.

El contenido de este parámetro depende del tipo de salida:

- Para una salida de seguridad de canal, al invocar la salida contiene un mensaje de seguridad si *ExitReason* es MQXR_SEC_MSG. Para devolver un mensaje de seguridad, la salida puede utilizar este almacenamiento intermedio o su propio almacenamiento intermedio (*ExitBufferAddr*).
- Para una salida de mensajes de canal, al invocar la salida, este parámetro contiene:
  - La cabecera de cola de transmisión (MQXQH), que incluye el descriptor de mensaje (que a su vez contiene la información de contexto del mensaje), seguida inmediatamente de
  - Los datos del mensaje

  Si el mensaje va a continuar, la salida puede realizar una de las acciones siguientes:

  - Deje el contenido del almacenamiento intermedio intacto
  - Modificar el contenido en su lugar (devolviendo la nueva longitud de los datos en *DataLength*; no debe ser mayor que *AgentBufferLength*)
  - Copie el contenido en *ExitBufferAddr*, realizando los cambios necesarios

  Los cambios que realiza la salida en la cabecera de cola de transmisión no se comprueban; sin embargo, las modificaciones erróneas pueden significar que el mensaje no se puede colocar en el destino.
- Para una salida de emisión o recepción de canal, al invocar la salida contiene los datos de transmisión. La salida puede realizar una de las acciones siguientes:
  - Deje el contenido del almacenamiento intermedio intacto
  - Modificar el contenido en su lugar (devolviendo la nueva longitud de los datos en *DataLength*; no debe ser mayor que *AgentBufferLength*)
  - Copie el contenido en *ExitBufferAddr*, realizando los cambios necesarios

  La salida no debe cambiar los primeros 8 bytes de los datos.

ExitBufferLongitud (MQLONG)-entrada/salida

Longitud del almacenamiento intermedio de salida.

En la primera invocación de la salida, este parámetro se establece en cero. A partir de entonces, cualquier valor que la salida devuelva, en cada invocación, se presentará a la salida la próxima vez que se invoque. El MCA no utiliza el valor.

Nota: Este parámetro no lo deben utilizar las salidas escritas en lenguajes de programación que no soportan el tipo de datos de puntero.

ExitBufferAddr (MQPTR)-entrada/salida

Dirección del almacenamiento intermedio de salida.

Este parámetro es un puntero a la dirección de un almacenamiento intermedio de almacenamiento gestionado por la salida, donde puede elegir devolver datos de mensaje o transmisión (en función del tipo de salida) al agente si el almacenamiento intermedio del agente es o no es lo suficientemente grande, o si es más conveniente que la salida lo haga.

En la primera invocación de la salida, la dirección pasada a la salida es nula. A partir de entonces, cualquier dirección que la salida devuelva, en cada invocación, se presentará a la salida la próxima vez que se invoque.

Nota: Este parámetro no lo deben utilizar las salidas escritas en lenguajes de programación que no dan soporte al tipo de datos de puntero.

Invocación en C

```
exitname (&ChannelExitParms, &ChannelDefinition,
         &DataLength, &AgentBufferLength, AgentBuffer,
         &ExitBufferLength, &ExitBufferAddr);
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
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 */
```

Invocación en COBOL

```
     CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION,
                           DATALENGTH, AGENTBUFFERLENGTH, AGENTBUFFER,
                           EXITBUFFERLENGTH, EXITBUFFERADDR.
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
**   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.
```

Invocación de RPG (ILE)

```
     C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     C                    CALLP     exitname(MQCXP : MQCD : DATLEN :
     C                                       ABUFL : ABUF : EBUFL :
     C                                       EBUF)
```

La definición de prototipo para la llamada es:

```
     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                            *
```

Invocación de ensamblador de System/390

```
         CALL EXITNAME,(CHANNELEXITPARMS,CHANNELDEFINITION,DATALENGTH, X
               AGENTBUFFERLENGTH,AGENTBUFFER,EXITBUFFERLENGTH,         X
               EXITBUFFERADDR)
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
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
```

Notas de uso

1. La función realizada por la salida de canal la define el proveedor de la salida. Sin embargo, la salida debe ajustarse a las reglas definidas aquí y en el bloque de control asociado, MQCXP.
2. El parámetro *ChannelDefinition* pasado a la salida de canal puede ser una de varias versiones. Consulte el campo *Version* en la estructura MQCD para obtener más información.
3. Si la salida de canal recibe una estructura MQCD con el campo *Version* establecido en un valor mayor que MQCD_VERSION_1, la salida debe utilizar el campo *ConnectionName* en MQCD, en preferencia al campo *ShortConnectionName* .
4. En general, las salidas de canal pueden cambiar la longitud de los datos de mensaje. Esto puede surgir como resultado de que la salida añada datos al mensaje, o elimine datos del mensaje, o comprima o cifre el mensaje. Sin embargo, se aplican restricciones especiales si el mensaje es un segmento que sólo contiene parte de un mensaje lógico. En particular, no debe haber ningún cambio neto en la longitud del mensaje como resultado de las acciones de salidas complementarias de envío y recepción.

   Por ejemplo, se permite que una salida de envío acorte el mensaje comprimiéndolo, pero la salida de recepción complementaria debe restaurar la longitud original del mensaje descomprimiéndolo, de modo que no haya ningún cambio neto en la longitud del mensaje.

   Esta restricción surge porque cambiar la longitud de un segmento haría que los desplazamientos de segmentos posteriores del mensaje fueran incorrectos, y esto inhibiría la capacidad del gestor de colas de reconocer que los segmentos formaban un mensaje lógico completo.

MQ_CHANNEL_AUTO_DEF_EXIT-Salida de definición automática de canal

La llamada MQ_CHANNEL_AUTO_DEF_EXIT describe los parámetros que se pasan a la salida de definición automática de canal llamada por el agente de canal de mensajes.

El gestor de colas no proporciona ningún punto de entrada denominado MQ_CHANNEL_AUTO_DEF_EXIT; el nombre MQ_CHANNEL_AUTO_DEF_EXIT no tiene ninguna importancia especial porque los nombres de las salidas de definición automática se proporcionan en el gestor de colas.

Sintaxis

MQ_CHANNEL_AUTO_DEF_EXIT *(ChannelExitParms, ChannelDefinition)*

Parámetros

La llamada MQ_CHANNEL_AUTO_DEF_EXIT tiene los parámetros siguientes.

Parámetros de ChannelExit(MQCXP)-entrada/salida

Bloque de parámetros de salida de canal.

Esta estructura contiene información adicional relacionada con la invocación de la salida.La salida establece información en esta estructura para indicar cómo procede el MCA.

ChannelDefinition (MQCD)-entrada/salida

Definición de canal.

Esta estructura contiene parámetros establecidos por el administrador para controlar el comportamiento de los canales que se crean automáticamente.La salida establece información en esta estructura para modificar el comportamiento predeterminado establecido por el administrador.

La salida no debe modificar los campos MQCD listados:

- *ChannelName*
- *ChannelType*
- *StrucLength*
- *Version*

Si se cambian otros campos, el valor establecido por la salida debe ser válido. Si el valor no es válido, se graba un mensaje de error en el archivo de registro de errores o se visualiza en la consola (según corresponda al entorno).

Invocación en C

```
exitname (&ChannelExitParms, &ChannelDefinition);
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
MQCXP  ChannelExitParms;   /* Channel exit parameter block */
MQCD   ChannelDefinition;  /* Channel definition */
```

Invocación en COBOL

```
     CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION.
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
**   Channel exit parameter block
 01  CHANNELEXITPARMS.
     COPY CMQCXPV.
**   Channel definition
 01  CHANNELDEFINITION.
     COPY CMQCDV.
```

Invocación de RPG (ILE)

```
     C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     C                   CALLP     exitname(MQCXP : MQCD)
```

La definición de prototipo para la llamada es:

```
     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
```

Invocación de ensamblador de System/390

```
         CALL EXITNAME,(CHANNELEXITPARMS,CHANNELDEFINITION)
```

Los parámetros pasados a la salida se declaran de la siguiente manera:

```
CHANNELEXITPARMS   CMQCXPA  ,  Channel exit parameter block
CHANNELDEFINITION  CMQCDA   ,  Channel definition
```

Notas de uso

1. La función realizada por la salida de canal la define el proveedor de la salida. Sin embargo, la salida debe ajustarse a las reglas definidas aquí y en el bloque de control asociado, MQCXP.
2. El parámetro *ChannelExitParms* pasado a la salida de definición automática de canal es una estructura MQCXP. La versión de MQCXP pasada depende del entorno en el que se ejecuta la salida; consulte la descripción del campo *Version* en "MQCXP-Parámetro de salida de canal" en la página 1075 para obtener más detalles.
3. El parámetro *ChannelDefinition* pasado a la salida de definición automática de canal es una estructura MQCD. La versión de MQCD pasada depende del entorno en el que se ejecuta la salida; consulte la descripción del campo *Version* en "MQCD-Definición de canal" en la página 1035 para obtener más detalles.

MQXWAIT-Esperar en salida

La llamada MQXWAIT espera a que se produzca un suceso. Solo se puede utilizar desde una salida de canal en z/OS.

El uso de MQXWAIT ayuda a evitar problemas de rendimiento que de otro modo podrían producirse si una salida de canal hace algo que provoca una espera. Un MVS ECB (bloque de control de sucesos) indica el suceso en espera de MQXWAIT. El ECB se describe en la descripción del bloque de control MQXWD.

Sintaxis

MQXWAIT *(Hconn, WaitDesc, CompCode, Reason)*

Parámetros

La llamada MQXWAIT tiene los parámetros siguientes.

Hconn (MQHCONN)-entrada

Descriptor de conexión.

Este manejador representa la conexión con el gestor de colas.El valor de *Hconn* ha sido devuelto por una llamada MQCONN anterior emitida en la misma invocación o anterior de la salida.

WaitDesc (MQXWD)-entrada/salida

Descriptor de espera.

Este parámetro describe el suceso a esperar.Consulte "MQXWD-Descriptor de espera de salida" en la página 1089 para obtener detalles de los campos de esta estructura.

CompCode (MQLONG)-salida

Código de terminación.

Es uno de los códigos siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida

Código de razón que califica *CompCode*.

Si el valor de *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') El adaptador no está disponible.

MQRC_OPTIONS_ERROR
(2046, X'7FE') Las opciones no son válidas o no son coherentes.

MQRC_XWAIT_CANCELADO
(2107, X'83B') Se ha cancelado la llamada MQXWAIT.

MQRC_XWAIT_ERROR
(2108, X'83C') Invocación de llamada MQXWAIT no válida.

Invocación en C

```
MQXWAIT (Hconn, &WaitDesc, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONN  Hconn;     /* Connection handle */
MQXWD    WaitDesc;  /* Wait descriptor */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Invocación de ensamblador de System/390

```
         CALL MQXWAIT,(HCONN,WAITDESC,COMPCODE,REASON)
```

Declare los parámetros como se indica a continuación:

```
HCONN     DS       F  Connection handle
WAITDESC  CMQXWDA  ,  Wait descriptor
COMPCODE  DS       F  Completion code
REASON    DS       F  Reason code qualifying COMPCODE
```

MQCD-Definición de canal

La estructura MQCD contiene los parámetros que controlan la ejecución de un canal. Se pasa a cada salida de canal que se llama desde un agente de canal de mensajes (MCA).

Si desea más información sobre las salidas de canal, consulte "MQ_CHANNEL_EXIT-Salida de canal" en la página 1027. La descripción de este tema está relacionada con los canales de mensajes y con los canales MQI.

Campos de nombre de salida

Cuando se llama a una salida, el campo relevante de *SecurityExit*, *MsgExit*, *SendExit*, *ReceiveExit*y *MsgRetryExit* contiene el nombre de la salida que se está invocando actualmente. El significado del nombre en estos campos depende del entorno en el que se ejecuta el MCA. Excepto donde se indique, el nombre se alinea a la izquierda dentro del campo, sin blancos intercalados; el nombre se rellena con blancos hasta la longitud del campo. En las descripciones siguientes, los corchetes ([]) indican información opcional:

Sistemas UNIX
El nombre de salida es el nombre de un módulo o biblioteca cargable dinámicamente, con el sufijo del nombre de una función que reside en dicha biblioteca. El nombre de función debe estar entre paréntesis. Opcionalmente, el nombre de biblioteca puede tener como prefijo una vía de acceso de directorio:

```
[path]library(function)
```

El nombre está limitado a un máximo de 128 caracteres.

z/OS
El nombre de salida es el nombre de un módulo de carga que es válido para la especificación en el parámetro EP de la macro LINK o LOAD. El nombre está limitado a un máximo de ocho caracteres.

Windows
El nombre de salida es el nombre de una biblioteca de enlace dinámico, con el sufijo del nombre de una función que reside en dicha biblioteca. El nombre de función debe estar entre paréntesis. Opcionalmente, el nombre de biblioteca puede tener como prefijo una vía de acceso de directorio y una unidad:

```
[d:][path]library(function)
```

El nombre está limitado a un máximo de 128 caracteres.

IBM i
El nombre de salida es un nombre de programa de 10 bytes seguido de un nombre de biblioteca de 10 bytes. Si los nombres tienen menos de 10 bytes de longitud, cada nombre se rellena con espacios en blanco para que tenga 10 bytes. El nombre de biblioteca puede ser `*LIBL` excepto cuando se llama a una salida de definición automática de canal, en cuyo caso se necesita un nombre completo.

Cambio de campos MQCD en una salida de canal

Una salida de canal pueden cambiar los campos del MQCD. El valor cambiado permanece en el MQCD y se pasa a las salidas restantes de una cadena de salida y a cualquier conversación que comparta la instancia de canal. El MQCD modificado también lo utiliza el MCA para su proceso normal durante el tiempo de vida continuo del canal.

La salida no debe modificar los siguientes campos MQCD:

- ChannelName
- ChannelType
- StrucLength

- Versión

Referencia relacionada

“Campos” en la página 1036
Este tema lista todos los campos de la estructura MQCD y describe cada campo.

“Declaración C” en la página 1062
Esta declaración es la declaración C para la estructura MQCD.

“declaración COBOL” en la página 1064
Esta declaración es la declaración COBOL para la estructura MQCD.

“Declaración RPG (ILE)” en la página 1066
Esta declaración es la declaración RPG para la estructura MQCD.

“ System/390” en la página 1069
Esta declaración es la declaración de ensamblador System/390 para la estructura MQCD.

“Declaración de Visual Basic” en la página 1071
Esta declaración es la declaración de Visual Basic de la estructura MQCD.

“Cambio de campos MQCD en una salida de canal” en la página 1072
Una salida de canal pueden cambiar los campos del MQCD. Sin embargo, estos cambios normalmente no se realizan, excepto en las circunstancias listadas.

Campos

Este tema lista todos los campos de la estructura MQCD y describe cada campo.

BatchHeartbeat (MQLONG)
Este campo especifica el intervalo de tiempo que se utiliza para desencadenar una pulsación por lotes para el canal.

La pulsación por lotes permite a los canales emisores determinar si la instancia de canal remoto sigue activa antes de pasar a ser dudosa. Se produce una pulsación por lotes si un canal emisor no se ha comunicado con la instancia de canal remoto dentro del intervalo de tiempo especificado.

El valor está en el rango de 0 a 999 999; las unidades son milisegundos. Un valor de cero indica que la pulsación por lotes no está habilitada.

Este campo sólo es relevante para los canales que tienen un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

BatchInterval (MQLONG)
Este campo especifica el tiempo aproximado en milisegundos que un canal mantiene abierto un lote, si se han transmitido menos de *BatchSize* mensajes en el lote actual.

Si *BatchInterval* es mayor que cero, el lote termina por cualquiera de los sucesos siguientes que se produzcan primero:

- Se han enviado *BatchSize* mensajes, o
- Han transcurrido *BatchInterval* milisegundos desde el inicio del lote.

Si *BatchInterval* es cero, el lote termina por cualquiera de los sucesos siguientes que se produzcan primero:

- Se han enviado *BatchSize* mensajes, o
- la cola de transmisión pasa a estar vacía.

BatchInterval debe estar en el rango de cero a 999 999 999 999.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente cuando *Version* es menor que MQCD_VERSION_4.

BatchSize (MQLONG)
Este campo especifica el número máximo de mensajes que se pueden enviar a través de un canal antes de sincronizar el canal.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_SVRCONN o MQCHT_CLNTCONN.

ChannelMonitoring (MQLONG)
Este campo especifica el nivel actual de recopilación de datos de supervisión para el canal.

Este campo no es relevante para los canales con un ChannelType de MQCHT_CLNTCONN.

Es uno de los valores siguientes:

- MQMON_OFF
- MQMON_LOW
- MQMON_MEDIO
- MQMON_HIGH

Es un campo de entrada para la salida. No está presente si *Version* es menor que MQCD_VERSION_8.

ChannelName (MQCHAR20)
Este campo especifica el nombre de definición de canal.

Debe haber una definición de canal del mismo nombre en la máquina remota para poder comunicarse.

El nombre debe utilizar sólo los caracteres:

- Mayúsculas A-Z
- Minúsculas a-z
- Números 0-9
- Punto (.)
- Barra inclinada (/)
- Subrayado (_)
- Signo de porcentaje (%)

y debe rellenarse a la derecha con espacios en blanco. No se permiten los espacios en blanco iniciales o intercalados.

La longitud de este campo la proporciona MQ_CHANNEL_NAME_LENGTH.

ChannelStatistics (MQLONG)
Este campo especifica el nivel actual de recopilación de datos de estadísticas para el canal.

Este campo no es relevante para los canales con un ChannelType de MQCHT_CLNTCONN.

Es uno de los valores siguientes:

- MQMON_OFF
- MQMON_LOW
- MQMON_MEDIO
- MQMON_HIGH

Es un campo de entrada para la salida. No está presente si *Version* es menor que MQCD_VERSION_8.

ChannelType (MQLONG)
Este campo especifica el tipo de canal.

Es uno de los valores siguientes:

MQCHT_SENDER
Remitente.

MQCHT_SERVER
Servidor.

MQCHT_RECEIVER
Receptor.

MQCHT_REQUESTER
Solicitante.

MQCHT_CLNTCONN
Conexión de cliente.

MQCHT_SVRCONN
Conexión de servidor (para que lo utilicen los clientes).

MQCHT_CLUSSDR
Remitente de clúster.

MQCHT_CLUSRCVR
Receptor de clúster.

Peso de ClientChannel(MQLONG)
Este campo especifica una ponderación para influir en qué definición de canal de conexión de cliente se utiliza.

El atributo de peso ClientChannelse utiliza para que las definiciones de canal de cliente se puedan seleccionar de forma aleatoria en función de su ponderación cuando haya más de una definición adecuada disponible. Cuando un cliente emite una MQCONN solicitando conexión con un grupo de gestores de colas, especificando un nombre de gestor de colas que empieza con un asterisco, y hay más de una definición de canal adecuada disponible en la tabla de definiciones de canal de cliente (CCDT), la definición que se debe utilizar se selecciona aleatoriamente basándose en la ponderación, con cualquier definición de ClientChannelWeight (0) aplicable seleccionada primero en orden alfabético.

Especifique un valor entre 0 y 99. El valor predeterminado es 0.

El valor 0 indica que no se realiza ningún equilibrio de carga y que las definiciones aplicables se seleccionan en orden alfabético. Para habilitar el equilibrio de carga, elija un valor entre 1 y 99, donde 1 es el peso más bajo y 99 el más alto. La distribución de mensajes entre dos o más canales con ponderaciones distintas de cero es proporcional a la proporción de esas ponderaciones. Por ejemplo, tres canales con valores de peso de ClientChannelde 2, 4 y 14 se seleccionan aproximadamente 10%, 20% y 70% del tiempo. Esta distribución no está garantizada.

Este atributo sólo es válido para el tipo de canal de conexión de cliente.

Es un campo de entrada para la salida. El campo no está presente si *Versión* es menor que MQCD_VERSION_9.

ClusterPtr (MQPTR)
Este campo especifica la dirección de una lista de nombres de clúster.

Si *ClustersDefined* es mayor que cero, esta dirección es la dirección de una lista de nombres de clúster.El canal pertenece a cada clúster listado.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_5.

ClustersDefined (MQLONG)
Este campo especifica el número de clústeres a los que pertenece el canal.

Este campo es el número de nombres de clúster a los que apunta *ClusterPtr*.Es cero o mayor.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_5.

CLWLChannelPriority (MQLONG)
Este campo especifica la prioridad del canal de carga de trabajo del clúster.

El algoritmo de selección del gestor de carga de trabajo selecciona un destino con la prioridad más alta del conjunto de destinos seleccionados en función del rango. Si hay dos gestores de colas de destino posibles, este atributo se puede utilizar para realizar una migración tras error de un gestor de colas en el otro gestor de colas. Todos los mensajes van al gestor de colas con la prioridad más alta hasta que finaliza, a continuación, los mensajes van al gestor de colas con la siguiente prioridad más alta.

El valor está en el rango de 0 a 9. El valor predeterminado es 0.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

Para obtener más información, consulte Configuración de un clúster de gestores de colas.

CLWLChannelRank (MQLONG)
Este campo especifica el rango de canal de carga de trabajo de clúster.

El algoritmo de selección del gestor de carga de trabajo selecciona un destino con el rango más alto. Cuando el destino final es un gestor de colas en un clúster diferente, puede establecer el rango de gestores de colas de pasarela intermedios (en la intersección de clústeres vecinos) para que el algoritmo de selección elija correctamente un gestor de colas de destino más próximo al destino final.

El valor está en el rango de 0 a 9. El valor predeterminado es 0.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

Para obtener más información, consulte Configuración de un clúster de gestores de colas .

CLWLChannelWeight (MQLONG)
Este campo especifica el peso del canal de carga de trabajo del clúster.

Peso de canal de carga de trabajo de clúster.

El algoritmo de selección del gestor de carga de trabajo utiliza el atributo "weight" del canal al desvío de la opción de destino para que se puedan enviar más mensajes a una máquina determinada. Por ejemplo, puede dar a un canal en un servidor UNIX grande un "peso" mayor que otro canal en un PC de escritorio pequeño, y el algoritmo de selección elige el servidor UNIX con más frecuencia que el PC.

El valor está en el rango de 1 a 99. El valor predeterminado es 50.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

Para obtener más información, consulte Configuración de un clúster de gestores de colas.

ConnectionAffinity (MQLONG)
Este campo especifica si las aplicaciones cliente que se conectan varias veces utilizando el mismo nombre de gestor de colas, utilizan el mismo canal de cliente.

Utilice este atributo cuando hay disponibles varias definiciones de canal aplicables.

El valor puede ser uno de los siguientes:

MQCAFTY_PREFERIDO
La primera conexión en un proceso que lee una tabla de definiciones de canal de cliente (CCDT) crea una lista de definiciones aplicables basadas en la ponderación con cualquier definición de CLNTWGHT (0) aplicable primero y en orden alfabético. Cada conexión del proceso intenta conectar utilizando la primera definición de la lista. Si una conexión no es satisfactoria, se utiliza la siguiente definición. Las

definiciones no satisfactorias con valores CLNTWGHT distintos de 0 se mueven al final de la lista. Las definiciones CLNTWGHT(0) permanecen en el principio de la lista y se seleccionan en primer lugar para cada conexión.

Cada proceso de cliente con el mismo nombre de host siempre crea la misma lista.

Para las aplicaciones cliente escritas en C, C++ o la infraestructura de programación .NET (incluido .NET totalmente gestionado), la lista se actualiza si la CCDT se ha modificado desde que se creó la lista.

Este es el valor predeterminado.

MQCAFTY_NONE
La primera conexión de un proceso que lee una CCDT crea una lista de definiciones aplicables. Todas las conexiones en un proceso seleccionan una definición aplicable según el peso con cualquier definición CLNTWGHT(0) aplicable seleccionada primero en orden alfabético.

Para las aplicaciones cliente escritas en C, C++ o la infraestructura de programación .NET (incluido .NET totalmente gestionado), la lista se actualiza si la CCDT se ha modificado desde que se creó la lista.

Este atributo sólo es válido para el tipo de canal de conexión de cliente.

Es un campo de entrada para la salida. El campo no está presente si *Versión* es menor que MQCD_VERSION_9.

ConnectionName (MQCHAR264)
Este campo especifica el nombre de conexión para el canal.

Para los canales de clúster receptor (cuando se especifica), CONNAME está relacionado con el gestor de colas local y para otros canales está relacionado con el gestor de colas de destino. El valor que especifique depende del protocolo de transmisión (*TransportType*) que se va a utilizar:

- Para MQXPT_LU62, es el nombre completo de la unidad lógica asociada.
- Para MQXPT_NETBIOS, es el nombre NetBIOS definido en la máquina remota.
- Para MQXPT_TCP, es el nombre de host, la dirección de red de la máquina remota especificada en formato decimal con puntos IPv4 o formato hexadecimal IPv6 , o la máquina local para canales de clúster receptor.
- Para MQXPT_SPX, es una dirección de estilo SPX que consta de una dirección de red de 4 bytes, una dirección de nodo de 6 bytes y un número de socket de 2 bytes.

Al definir un canal, este campo no es relevante para los canales con un *ChannelType* de MQCHT_SVRCONN o MQCHT_RECEIVER. Sin embargo, cuando la definición de canal se pasa a una salida, este campo contiene la dirección del socio, sea cual sea el tipo de canal.

La longitud de este campo la proporciona MQ_CONN_NAME_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_2.

DataConversion (MQLONG)
Este campo especifica si el agente de canal de mensajes emisor intenta la conversión de los datos de mensaje de aplicación si el agente de canal de mensajes receptor no puede realizar esta conversión.

Este campo sólo se aplica a los mensajes que no son segmentos de mensajes lógicos; el MCA nunca intenta convertir mensajes que son segmentos.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR. Es uno de los siguientes:

MQCDC_SENDER_CONVERSION
Conversión por remitente.

MQCDC_NO_SENDER_CONVERSION
Sin conversión por remitente.

DefReconnect (MQLONG)
El atributo de canal `DefReconnect` establece el valor de atributo de reconexión predeterminado para un canal de conexión de cliente.

La opción de reconexión de cliente automática predeterminada. Puede configurar un IBM WebSphere MQ MQI client para que reconecte automáticamente una aplicación cliente. El IBM WebSphere MQ MQI client intenta reconectarse a un gestor de colas después de la anomalía de una anomalía de la conexión. Intenta reconectarse sin que el cliente de la aplicación emita una llamada `MQCONN` o `MQCONNX` MQI.

La reconexión es una opción de `MQCONNX` . Utilizando el atributo de canal `DefReconnect` puede añadir un comportamiento de reconexión a las aplicaciones existentes que utilizan `MQCONN`. También puede cambiar el comportamiento de reconexión de las aplicaciones que utilizan `MQCONNX`.

También puede establecer el valor `DefRecon` desde el archivo `mqclient.ini` para establecer o modificar el comportamiento de reconexión. El valor `DefRecon` del archivo `mqclient.ini` tiene prioridad sobre el atributo de canal `DefReconnect` .

Syntax
`DefReconnect( MQRCN_NO|MQRCN_YES|MQRCN_Q_MGR|`
`MQRCN_DISABLED)`

Parámetros
MQRCN_NO
`MQRCN_NO` es el valor predeterminado.

A menos que `MQCONNX`lo altere temporalmente, el cliente no se vuelve a conectar automáticamente.

MQRCN_YES
A menos que `MQCONNX`lo altere temporalmente, el cliente se vuelve a conectar automáticamente.

MQRCN_Q_MGR
A menos que lo altere temporalmente `MQCONNX`, el cliente se vuelve a conectar automáticamente, pero sólo al mismo gestor de colas. La opción `QMGR` tiene el mismo efecto que `MQCNO_RECONNECT_Q_MGR`.

MQRCN_DISABLED
La reconexión está inhabilitada, aunque lo solicite el programa cliente utilizando la llamada MQI de `MQCONNX` .

La reconexión automática de cliente no está soportada por las clases IBM WebSphere MQ para Java.

Tabla 592. La reconexión automática depende de los valores establecidos en la aplicación y en la definición de canal

DefReconnect	**Opciones de reconexión establecidas en la aplicación**			
	`MQCNO_RECONNECT`	`MQCNO_RECONNECT_Q_MGR`	`MQCNO_RECONNECT_AS_DEF`	`MQCNO_RECONNECT_DISABLED`
`MQRCN_NO`	SÍ	QMGR	No	No
`MQRCN_YES`	SÍ	QMGR	SÍ	No
`MQRCN_Q_MGR`	SÍ	QMGR	QMGR	No
`MQRCN_DISABLED`	No	No	No	No

Conceptos relacionados
Reconexión de cliente automática
Reconexión de canal y cliente
Stanza CHANNELS del archivo de configuración de cliente

Referencia relacionada

Opciones de conexión
Opciones que controlan la acción de MQCONNX.

Descripción (MQCHAR64)
Este campo se puede utilizar para comentarios descriptivos.

El contenido del campo no es significativo para los agentes de canal de mensajes. Sin embargo, sólo debe contener caracteres que se puedan visualizar. No puede contener ningún carácter nulo; si es necesario, se rellena a la derecha con espacios en blanco. En una instalación DBCS, el campo puede contener caracteres DBCS (sujetos a una longitud máxima de campo de 64 bytes).

Nota: Si este campo contiene caracteres que no están en el juego de caracteres del gestor de colas (tal como se define en el atributo de gestor de colas *CodedCharSetId*), es posible que estos caracteres se conviertan incorrectamente si este campo se envía a otro gestor de colas.

La longitud de este campo la proporciona MQ_CHANNEL_DESC_LENGTH.

DiscInterval (MQLONG)
Este campo especifica el tiempo máximo en segundos durante el cual el canal espera a que llegue un mensaje a la cola de transmisión, antes de terminar el canal.

En otras palabras, especifica el intervalo de desconexión.

El valor A de cero hace que el MCA espere indefinidamente.

Para los canales de conexión de servidor que utilizan el protocolo TCP, el intervalo representa el valor de desconexión de inactividad del cliente, especificado en segundos. Si una conexión de servidor no ha recibido ninguna comunicación de su cliente asociado durante este tiempo, termina la conexión. El intervalo de inactividad de conexión de servidor sólo se aplica entre las llamadas de API de WebSphere MQ de un cliente, por lo que ningún cliente se desconecta durante una MQGET de larga ejecución con llamada de espera.

Este atributo no es aplicable para los canales de conexión de servidor que utilizan protocolos distintos de TCP.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, MQCHT_CLUSRCVR o MQCHT_SVRCONN.

ExitDataLongitud (MQLONG)
Este campo especifica la longitud en bytes de cada uno de los elementos de datos de usuario en las listas de elementos de datos de usuario de salida a los que se dirigen los campos *MsgUserDataPtr*, *SendUserDataPtr*y *ReceiveUserDataPtr* .

Esta longitud no es necesariamente la misma que MQ_EXIT_DATA_LENGTH.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

ExitNameLongitud (MQLONG)
Este campo especifica la longitud en bytes de cada uno de los nombres de las listas de nombres de salida a los que se dirigen los campos *MsgExitPtr*, *SendExitPtr*y *ReceiveExitPtr* .

Esta longitud no es necesariamente la misma que MQ_EXIT_NAME_LENGTH.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

Lista de HdrComp[2] (MQLONG)
Este campo especifica la lista de técnicas de compresión de datos de cabecera soportadas por el canal.

La lista contiene uno o varios de los valores siguientes:

MQCOMPRESS_NONE
No se lleva a cabo ninguna compresión de datos de cabecera.

MQCOMPRESS_SISTEMA
Se lleva a cabo la compresión de datos de cabecera.

Los valores no utilizados en la matriz se establecen en MQCOMPRESS_NOT_AVAILABLE.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

HeartbeatInterval (MQLONG)
Este campo especifica el tiempo en segundos entre los flujos de pulsaciones.

La interpretación de este campo depende del tipo de canal, tal como se indica a continuación:

- Para un tipo de canal de MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER MQCHT_REQUESTER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR, este campo es el tiempo en segundos entre los flujos de pulsaciones pasados desde el MCA emisor cuando no hay mensajes en la cola de transmisión. Esto da al MCA receptor la oportunidad de desactivar temporalmente el canal. Para ser útil, *HeartbeatInterval* debe ser menor que *DiscInterval*.
- Para un tipo de canal de MQCHT_CLNTCONN o MQCHT_SVRCONN con el campo Conversaciones de compartición de MQCD establecido en cero, este campo es el tiempo en segundos entre los flujos de latido pasados desde el MCA del servidor cuando dicho MCA ha emitido una llamada MQGET con la opción MQGMO_WAIT en nombre de una aplicación cliente. Esto permite al MCA del servidor manejar situaciones en las que la conexión de cliente falla durante un MQGET con MQGMO_WAIT.
- Para un tipo de canal de MQCHT_CLNTCONN o MQCHT_SVRCONN con el campo Conversaciones de compartición de MQCD establecido en un valor distinto de cero, este campo es el tiempo en segundos entre el flujo de pulsaciones cuando no hay flujos de datos enviados o recibidos. Esto permite que el canal se desactive temporalmente de forma eficiente.

El valor está en el rango de 0 a 999 999. El valor que se utiliza es el mayor de los valores especificados en el lado de envío y el lado de recepción a menos que se especifique un valor de 0 en cada lado, en cuyo caso no se produce ningún intercambio de pulsaciones.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

Intervalo de KeepAlive(MQLONG)
Este campo especifica el valor pasado a la pila de comunicaciones para la temporización de estado activo para el canal.

El valor es aplicable para los protocolos de comunicaciones TCP/IP y SPX, aunque no todas las implementaciones soportan este parámetro.

El valor está en el rango de 0 a 99 999; las unidades son segundos. Un valor de cero indica que el estado activo del canal no está habilitado, aunque el estado activo puede seguir produciéndose si el estado activo TCP/IP (en lugar del estado activo del canal) está habilitado. También es válido el siguiente valor especial:

MQKAI_AUTO
Automático.

Este valor indica que el intervalo de estado activo se calcula a partir del intervalo de latido negociado, como se indica a continuación:

- Si el intervalo de pulsaciones negociado es mayor que cero, el intervalo de estado activo que se utiliza es el intervalo de pulsaciones más 60 segundos.
- Si el intervalo de pulsaciones negociado es cero, el intervalo de estado activo que se utiliza es cero.

- En z/OS, el estado activo de TCP/IP se produce cuando se especifica TCPKEEP (YES) en el objeto del gestor de colas.
- En otros entornos, el estado activo de TCP/IP se produce cuando se especifica el parámetro KEEPALIVE=YES en la stanza TCP del archivo de configuración de gestión de colas distribuidas.

Este campo sólo es relevante para los canales que tienen un *TransportType* de MQXPT_TCP o MQXPT_SPX.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

LocalAddress (MQCHAR48)
Este campo especifica la dirección TCP/IP local definida para el canal para las comunicaciones de salida.

Este campo está en blanco si no se ha definido ninguna dirección específica para las comunicaciones de salida.La dirección puede incluir opcionalmente un número de puerto o un rango de números de puerto. El formato de esta dirección es:

```
[ip-addr][(low-port[,high-port])]
```

donde los corchetes ([]) indican información opcional, `ip-addr` se especifica en IPv4 decimal con puntos, IPv6 hexadecimal o formato alfanumérico, y `low-port` y `high-port` son números de puerto entre paréntesis. Todos son opcionales.

Una dirección IP, puerto o rango de puertos específicos para las comunicaciones de salida es útil en escenarios de recuperación en los que un canal se reinicia en una pila TCP/IP diferente.

LocalAddress es similar en forma a *ConnectionName*, pero no debe confundirse con él. *LocalAddress* especifica las características de las comunicaciones locales, mientras que *ConnectionName* especifica cómo llegar a un gestor de colas remoto.

Este campo sólo es relevante para canales con un *TransportType* de MQXPT_TCP y un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLNTCONN, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

La longitud de este campo la proporciona MQ_LOCAL_ADDRESS_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_7.

LongMCAUserIdLength (MQLONG)
Este campo especifica la longitud en bytes del identificador de usuario MCA completo al que apunta *LongMCAUserIdPtr*.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_CLNTCONN.

Es un campo de entrada/salida para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_6.

LongMCAUserIdPtr (MQPTR)
Este campo especifica la dirección del identificador de usuario de MCA largo.

Si *LongMCAUserIdLength* es mayor que cero, este campo es la dirección del identificador de usuario MCA completo.La longitud del identificador completo la proporciona *LongMCAUserIdLength*. Los primeros 12 bytes del identificador de usuario de MCA también están contenidos en el campo *MCAUserIdentifier*.

Consulte la descripción del campo *MCAUserIdentifier* para obtener detalles del identificador de usuario de MCA.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_SDR, MQCHT_SVR, MQCHT_CLNTCONN o MQCHT_CLUSSDR.

Es un campo de entrada/salida para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_6.

LongRemoteUserIdLongitud (MQLONG)
Este campo especifica la longitud en bytes del identificador de usuario remoto completo al que apunta *LongRemoteUserIdPtr*.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLNTCONN o MQCHT_SVRCONN.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_6.

LongRemoteUserIdPtr (MQPTR)
Este campo especifica la dirección del identificador de usuario remoto largo.

Si *LongRemoteUserIdLength* es mayor que cero, este distintivo es la dirección del identificador de usuario remoto completo.La longitud del identificador completo la proporciona *LongRemoteUserIdLength*. Los primeros 12 bytes del identificador de usuario remoto también están contenidos en el campo *RemoteUserIdentifier*.

Consulte la descripción del campo *RemoteUserIdentifier* para obtener detalles del identificador de usuario remoto.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLNTCONN o MQCHT_SVRCONN.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_6.

Recuento de LongRetry(MQLONG)
Este campo especifica el recuento utilizado después de que se haya agotado el recuento especificado por *ShortRetryCount* .

Especifica el número máximo de intentos adicionales que se realizan para conectarse a la máquina remota, a intervalos especificados por *LongRetryInterval*, antes de registrar un error en el operador.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Intervalo LongRetry(MQLONG)
Este campo especifica el número máximo de segundos que se debe esperar antes de volver a intentar la conexión con la máquina remota.

El intervalo entre reintentos se puede ampliar si el canal tiene que esperar a estar activo.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

MaxInstances (MQLONG)
Este campo especifica el número máximo de instancias simultáneas de un canal de conexión de servidor individual que se pueden iniciar.

Este campo sólo se utiliza en canales de conexión de servidor.

El campo puede tener un valor en el rango de 0 a 999.999.999. El valor cero impide el acceso de los clientes.

El valor por omisión de este campo es 999 999 999.

Si el valor de este campo se reduce a un número menor que el número de instancias del canal de conexión con el servidor que se están ejecutando actualmente, estas instancias en ejecución no se verán afectadas. Sin embargo, las nuevas instancias no se pueden iniciar hasta que hayan dejado de ejecutarse suficientes instancias existentes, de modo que el número de instancias actualmente en ejecución sea menor que el valor del campo.

MaxInstancesPerClient (MQLONG)
Este campo especifica el número máximo de instancias simultáneas de un canal de conexión de servidor individual que se pueden iniciar desde un único cliente.

En este contexto, las conexiones que se originan desde la misma dirección de red remota se consideran procedentes del mismo cliente.

Este campo sólo se utiliza en canales de conexión de servidor.

El campo puede tener un valor en el rango de 0 a 999.999.999. El valor cero impide el acceso de los clientes.

El valor por omisión de este campo es 999 999 999.

Si el valor de este campo se reduce a un número menor que el número de instancias del canal de conexión con el servidor que se están ejecutando actualmente desde clientes individuales, las instancias en ejecución no se verán afectadas. Sin embargo, las nuevas instancias de cualquiera de estos clientes no se pueden iniciar hasta que hayan dejado de ejecutarse suficientes instancias existentes de modo que el número de instancias actualmente en ejecución, que se originan en el cliente que intenta iniciar una nueva, sea menor que el valor del campo.

MaxMsgLongitud (MQLONG)
Este campo especifica la longitud máxima de mensaje que puede transmitirse en el canal.

Este valor se compara con el del canal remoto y el máximo real es el menor de los dos valores.

MCAName (MQCHAR20)
Este campo es un campo reservado.

El valor de este campo está en blanco.

La longitud de este campo la proporciona MQ_MCA_NAME_LENGTH.

MCASecurityId (MQBYTE40)
Este campo especifica el identificador de seguridad para el MCA.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_CLNTCONN.

El siguiente valor especial indica que no hay ningún identificador de seguridad:

MQSID_NONE
No se ha especificado ningún identificador de seguridad.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQSID_NONE_ARRAY; esta constante tiene el mismo valor que MQSID_NONE, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada/salida para la salida. La longitud de este campo la proporciona MQ_SECURITY_ID_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_6.

MCAType (MQLONG)
Este campo especifica el tipo de programa de agente de canal de mensajes.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

El valor puede ser uno de los siguientes:

MQMCAT_PROCESO
proceso.

El agente de canal de mensajes se ejecuta como un proceso independiente.

HEBRA MQMCAT_THREAD
Hebra (IBM i, UNIXy Windows).

El agente de canal de mensajes se ejecuta como una hebra separada.

Este campo no está presente cuando *Versión* es menor que MQCD_VERSION_2.

MCAUserIdentifier (MQCHAR12)
Este campo especifica el identificador de usuario para el agente de canal de mensajes (MCA).

Este campo utiliza los primeros 12 bytes del identificador de usuario de MCA y puede ser establecido por un agente de seguridad.

Hay dos campos que contienen el identificador de usuario de MCA:

- *MCAUserIdentifier* contiene los primeros 12 bytes del identificador de usuario de MCA y se rellena con espacios en blanco si el identificador es inferior a 12 bytes. *MCAUserIdentifier* puede estar en blanco.
- *LongMCAUserIdPtr* apunta al identificador de usuario de MCA completo, que puede tener más de 12 bytes. Su longitud la proporciona *LongMCAUserIdLength*. El identificador completo no contiene blancos de cola y no termina en nulo. Si el identificador está en blanco, *LongMCAUserIdLength* es cero y el valor de *LongMCAUserIdPtr* no está definido.

  Nota: *LongMCAUserIdPtr* no está presente si *Version* es menor que MQCD_VERSION_6.

Si el identificador de usuario de MCA no está en blanco, especifica el identificador de usuario que utilizará el agente de canal de mensajes para la autorización para acceder a los recursos de WebSphere MQ . Para los tipos de canal MQCHT_REQUESTER, MQCHT_RECEIVER y MQCHT_CLUSRCVR, si PutAuthority es MQPA_DEFAULT, este es el identificador de usuario utilizado para las comprobaciones de autorización para la operación de colocación en colas de destino.

Si el identificador de usuario de MCA está en blanco, el agente de canal de mensajes utiliza su identificador de usuario predeterminado.

El identificador de usuario de MCA se puede establecer mediante una salida de seguridad para indicar el identificador de usuario que debe utilizar el agente de canal de mensajes. La salida puede cambiar *MCAUserIdentifier*o la serie a la que apunta *LongMCAUserIdPtr*. Si ambos se cambian pero difieren entre sí, el MCA utiliza *LongMCAUserIdPtr* en preferencia a *MCAUserIdentifier*. Si la salida cambia la longitud de la serie a la que se refiere *LongMCAUserIdPtr*, *LongMCAUserIdLength* debe establecerse de forma correspondiente. Si la salida aumenta la longitud del identificador, la salida debe asignar el almacenamiento de la longitud necesaria, establecer dicho almacenamiento en el identificador necesario y colocar la dirección de dicho almacenamiento en *LongMCAUserIdPtr*. La salida es responsable de liberar ese almacenamiento cuando la salida se invoca más tarde con la razón MQXR_TERM.

Para los canales con un *ChannelType* de MQCHT_SVRCONN, si *MCAUserIdentifier* en la definición de canal está en blanco, cualquier identificador de usuario transferido desde el cliente se copia en él. Este identificador de usuario (después de cualquier modificación por parte de la salida de seguridad en el servidor) es el que se supone que ejecuta la aplicación cliente.

El identificador de usuario de MCA no es relevante para los canales con un *ChannelType* de MQCHT_SDR, MQCHT_SVR, MQCHT_CLNTCONN, MQCHT_CLUSSDR.

Es un campo de entrada/salida para la salida. La longitud de este campo la proporciona MQ_USER_ID_LENGTH. Este campo no está presente cuando *Version* es menor que MQCD_VERSION_2.

ModeName (MQCHAR8)
Este campo especifica el nombre de modalidad de LU 6.2 .

Este campo sólo es relevante si el protocolo de transmisión (*TransportType*) es MQXPT_LU62y *ChannelType* no es MQCHT_SVRCONN ni MQCHT_RECEIVER.

Este campo siempre está en blanco. En su lugar, la información está contenida en el objeto lateral de comunicaciones.

La longitud de este campo la proporciona MQ_MODE_NAME_LENGTH.

MsgCompLista [16] (MQLONG)
Este campo especifica la lista de técnicas de compresión de datos de mensaje soportadas por el canal.

La lista contiene uno o varios de los valores siguientes:

MQCOMPRESS_NONE
No se lleva a cabo ninguna compresión de datos de mensaje.

MQCOMPRESS_RLE
Se lleva a cabo la compresión de datos de mensaje utilizando la codificación de longitud de ejecución.

MQCOMPRESS_ZLIBFAST
Se lleva a cabo la compresión de datos de mensaje utilizando el método de compresión zlib. Se prefiere un tiempo de compresión rápido.

MQCOMPRESS_ZLIBHIGH
Se lleva a cabo la compresión de datos de mensaje utilizando el método de compresión zlib. Se prefiere un nivel elevado de compresión.

Los valores no utilizados en la matriz se establecen en MQCOMPRESS_NOT_AVAILABLE.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_8.

MsgExit (MQCHARn)
Este campo especifica el nombre de salida de mensaje de canal.

Si este nombre no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente después de que se haya recuperado un mensaje de la cola de transmisión (emisor o servidor), o inmediatamente antes de que se coloque un mensaje en una cola de destino (receptor o solicitante).

  A la salida se le proporciona toda la cabecera de cola de mensajes y transmisión de la aplicación para su modificación.
- Durante la inicialización y terminación del canal.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_SVRCONN o MQCHT_CLNTCONN; nunca se invoca una salida de mensaje para dichos canales.

Consulte “MQCD-Definición de canal” en la página 1035 para obtener una descripción del contenido de este campo en varios entornos.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

MsgExitPtr (MQPTR)
Este campo especifica la dirección del primer campo *MsgExit* .

Si *MsgExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de nombres de cada salida de mensaje de canal de la cadena.

Cada nombre está en un campo de longitud *ExitNameLength*, rellenado a la derecha con espacios en blanco. Hay *MsgExitsDefined* campos que se unen entre sí-uno para cada salida.

Los cambios realizados en estos nombres por una salida se conservan, aunque la salida de canal de mensajes no realiza ninguna acción explícita-no cambia qué salidas se invocan.

Si *MsgExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

MsgExitsdefinido (MQLONG)
Este campo especifica el número de salidas de mensajes de canal definidas en la cadena.

Es mayor o igual que cero.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

Recuento de MsgRetry(MQLONG)
Este campo especifica el número de veces que el MCA intenta colocar el mensaje, después de que el primer intento haya fallado.

Este campo indica el número de veces que el MCA intenta la operación de apertura o colocación, si el primer MQOPEN o MQPUT falla con el código de terminación MQCC_FAILED.El efecto de este atributo depende de si *MsgRetryExit* está en blanco o no:

- Si *MsgRetryExit* está en blanco, el atributo *MsgRetryCount* controla si el MCA intenta reintentos. Si el valor del atributo es cero, no se intenta ningún reintento. Si el valor del atributo es mayor que cero, los reintentos se intentan a intervalos proporcionados por el atributo *MsgRetryInterval* .

  Los reintentos sólo se intentan para los siguientes códigos de razón:

  – MQRC_PAGESET_FULL
  – MQRC_PUT_XX_ENCODE_CASE_CAPS_LOCK_ON inhibida
  – MQRC_Q_FULL

  Para otros códigos de razón, el MCA continúa inmediatamente con su proceso de anomalía normal, sin reintentar el mensaje anómalo.
- Si *MsgRetryExit* no está en blanco, el atributo *MsgRetryCount* no afecta al MCA; en su lugar, es la salida de reintento de mensaje la que determina cuántas veces se intenta el reintento y a qué intervalos; la salida se invoca aunque el atributo *MsgRetryCount* sea cero.

  El atributo *MsgRetryCount* está disponible para la salida en la estructura MQCD, pero la salida que no es necesaria para respetarlo-los reintentos continúan indefinidamente hasta que la salida devuelve MQXCC_SUPPRESS_FUNCTION en el campo *ExitResponse* de MQCXP.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

Este campo no está presente cuando *Version* es menor que MQCD_VERSION_3.

Salida MsgRetry(MQCHARn)
Este campo especifica el nombre de salida de reintento de mensaje de canal.

La salida de reintento de mensaje es una salida que invoca el MCA cuando el MCA recibe un código de terminación de MQCC_FAILED de una llamada MQOPEN o MQPUT.La finalidad de la salida es especificar un intervalo de tiempo durante el cual el MCA espera antes de volver a intentar la operación MQOPEN o MQPUT. De forma alternativa, la salida se puede establecer para no volver a intentar la operación.

La salida se invoca para todos los códigos de razón que tienen un código de terminación de MQCC_FAILED-los valores de la salida determinan qué códigos de razón desea que el MCA vuelva a intentarlo, para cuántos intentos y en qué intervalos de tiempo.

Cuando la operación ya no se intenta, el MCA realiza su proceso de anomalía normal; este proceso incluye la generación de un mensaje de informe de excepción (si lo especifica el remitente) y la colocación del mensaje original en la cola de mensajes no entregados o el descarte del mensaje (según si el remitente ha especificado MQRO_DEAD_LETTER_Q o MQRO_DISCARD_MSG). Las anomalías relacionadas con la cola de mensajes no entregados (por ejemplo, la cola de mensajes no entregados llena) no hacen que se invoque la salida de reintento de mensaje.

Si el nombre de salida no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente antes de realizar la espera antes de volver a intentar entregar un mensaje
- Durante la inicialización y terminación del canal

Consulte “MQCD-Definición de canal” en la página 1035 para obtener una descripción del contenido de este campo en varios entornos.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

Este campo no está presente cuando *Version* es menor que MQCD_VERSION_3.

Intervalo MsgRetry(MQLONG)
Este campo especifica el intervalo mínimo en milisegundos después del cual se reintenta la operación de apertura o colocación.

El efecto de este atributo depende de si *MsgRetryExit* está en blanco o no:

- Si *MsgRetryExit* está en blanco, el atributo *MsgRetryInterval* especifica el periodo mínimo que el MCA espera antes de reintentar un mensaje, si el primer MQOPEN o MQPUT falla con el código de terminación MQCC_FAILED. Un valor de cero significa que el reintento se realizará lo antes posible después del intento anterior. Los reintentos sólo se realizan si *MsgRetryCount* es mayor que cero.

  Este atributo también se utiliza como tiempo de espera si la salida de reintento de mensaje devuelve un valor no válido en el campo *MsgRetryInterval* de MQCXP.
- Si *MsgRetryExit* no está en blanco, el atributo *MsgRetryInterval* no afecta al MCA; en su lugar, es la salida de reintento de mensaje la que determina cuánto tiempo espera el MCA. El atributo *MsgRetryInterval* se pone a disposición de la salida en la estructura MQCD, pero la salida no es necesaria para respetarlo.

El valor está en el rango de 0 a 999.999.999.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

Este campo no está presente cuando *Version* es menor que MQCD_VERSION_3.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_4.

MsgRetryUserData (MQCHAR32)
Este campo especifica los datos de usuario de salida de reintento de mensaje de canal.

Estos datos se pasan a la salida de reintento de mensaje de canal en el campo *ExitData* del parámetro *ChannelExitParms* (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Estos cambios no afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR.

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH. Este campo no está presente cuando *Version* es menor que MQCD_VERSION_3.

Este campo no es relevante en WebSphere MQ para IBM i.

Datos de MsgUser(MQCHAR32)
Este campo especifica los datos de usuario de salida de mensajes de canal.

Estos datos se pasan a la salida de mensajes de canal en el campo *ExitData* del parámetro *ChannelExitParms* (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el

contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Estos cambios no afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Este campo no es relevante en WebSphere MQ para IBM i.

MsgUserDataPtr (MQPTR)
Este campo especifica la dirección del primer campo *MsgUserData* .

Si *MsgExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de elementos de datos de usuario para cada salida de mensaje de canal de la cadena.

Cada elemento de datos de usuario está en un campo de longitud *ExitDataLength*, rellenado a la derecha con espacios en blanco. Hay *MsgExitsDefined* campos que se unen entre sí-uno para cada salida. Si el número de elementos de datos de usuario definidos es menor que el número de nombres de salida, los elementos de datos de usuario no definidos se establecen en blancos. Por el contrario, si el número de elementos de datos de usuario definidos es mayor que el número de nombres de salida, los elementos de datos de usuario en exceso se ignoran y no se presentan a la salida.

Los cambios realizados en estos valores por una salida se conservan. Esto permite que una salida pase información a otra salida. No se realiza ninguna validación en ningún cambio, por lo que, por ejemplo, los datos binarios se pueden escribir en estos campos si es necesario.

Si *MsgExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

NetworkPriority (MQLONG)
Este campo especifica la prioridad de la conexión de red para el canal.

Cuando hay disponibles varias vías de acceso a un destino determinado, se elige la vía de acceso con la prioridad más alta. El valor está en el rango de 0 a 9; 0 es la prioridad más baja.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_5.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_6.

NonPersistentMsgSpeed (MQLONG)
Este campo especifica la velocidad a la que viajan los mensajes no persistentes a través del canal.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER, MQCHT_REQUESTER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

El valor puede ser uno de los siguientes:

MQNPMS_NORMAL
Velocidad normal.

Si un canal está definido como MQNPMS_NORMAL, los mensajes no persistentes viajan a través del canal a una velocidad normal. Esto tiene la ventaja de que estos mensajes no se pierden si hay una anomalía de canal. Además, los mensajes persistentes y no persistentes en la misma cola de transmisión mantienen su orden relativo entre sí.

MQNPMS_FAST
Velocidad rápida.

Si un canal está definido como MQNPMS_FAST, los mensajes no persistentes viajan a través del canal a una velocidad rápida. Esto mejora el rendimiento del canal, pero significa que los mensajes no persistentes se pierden si se produce una anomalía de canal. Además, es posible que los mensajes no persistentes salten por delante de los mensajes persistentes que esperan en la misma cola de transmisión, es decir, el orden de los mensajes no persistentes no se mantiene en relación con los mensajes persistentes. Sin embargo, se mantiene el orden de los mensajes no persistentes relativos entre sí. De forma similar, se mantiene el orden de los mensajes persistentes relativos entre sí.

Contraseña (MQCHAR12)
Este campo especifica la contraseña utilizada por el agente de canal de mensajes al intentar iniciar una sesión SNA segura con un agente de canal de mensajes remoto.

Este campo no puede estar en blanco sólo en sistemas UNIX y Windows , y sólo es relevante para los canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER o MQCHT_CLNTCONN. En z/OS, este campo no es relevante.

La longitud de este campo la proporciona MQ_PASSWORD_LENGTH. Sin embargo, sólo se utilizan los primeros 10 caracteres.

Este campo no está presente si *Version* es menor que MQCD_VERSION_2.

PropertyControl (MQLONG)
Este campo especifica qué sucede con las propiedades de los mensajes cuando el mensaje está a punto de enviarse a un gestor de colas V6 o anterior (un gestor de colas que no entiende el concepto de un descriptor de propiedad).

El valor puede ser:

COMPATIBILIDAD de MQPROP_COMPATIBILITY
Si el mensaje contiene una propiedad con el prefijo **mcd.**, **jms.**, **usr.** o **mqext.**, todas las propiedades del mensaje se entregan a la aplicación en una cabecera MQRFH2 . De lo contrario, todas las propiedades del mensaje, excepto las propiedades contenidas en el descriptor de mensaje (o extensión), se descartan y ya no son accesibles para la aplicación.

Este valor es el valor predeterminado; permite a las aplicaciones, que esperan que las propiedades relacionadas con JMS estén en una cabecera MQRFH2 en los datos del mensaje, continuar trabajando sin modificar.

MQPROP_NONE
Todas las propiedades del mensaje, excepto las propiedades del descriptor de mensaje (o extensión), se eliminan del mensaje antes de que el mensaje se envíe al gestor de colas remoto.

MQPROP_ALL
Todas las propiedades del mensaje se incluyen con el mensaje cuando se envía al gestor de colas remoto. Las propiedades, excepto aquellas que se encuentran en el descriptor de mensaje (o extensión), se colocan en una o más cabeceras MQRFH2 en los datos del mensaje.

Este atributo es aplicable a los canales Remitente, Servidor, Remitente de clúster y Receptor de clúster.

“MQIA_ * (Selectores de atributos enteros)” en la página 115
“MQPROP_ * (Valores de control de propiedades de cola y canal y longitud máxima de propiedades)” en la página 152

PutAuthority (MQLONG)
Este campo especifica si el identificador de usuario en la información de contexto asociada con un mensaje se utiliza para establecer la autorización para transferir el mensaje a la cola de destino.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_REQUESTER, MQCHT_RECEIVER o MQCHT_CLUSRCVR. Es uno de los siguientes:

MQPA_PREDETERMINADO
Se utiliza el identificador de usuario predeterminado.

CONTEXTO_MQPA
Se utiliza el identificador de usuario de contexto.

MQPA_ALTERNATE_OR_MCA
Se utiliza el ID de usuario del campo UserIdentifier del descriptor de mensaje. No se utiliza ningún ID de usuario recibido de la red. Este valor solo está soportado en z/OS.

MQPA_ONLY_MCA
Se utiliza el ID de usuario predeterminado. No se utiliza ningún ID de usuario recibido de la red. Este valor solo está soportado en z/OS.

QMgrName (MQCHAR48)
Este campo especifica el nombre del gestor de colas al que se puede conectar una salida.

Para canales con un *ChannelType* que no sea MQCHT_CLNTCONN, este campo es el nombre del gestor de colas al que se puede conectar una salida, que en los sistemas UNIX, Linux y Windows , siempre no está en blanco.

La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH.

ReceiveExit (MQCHARn)
Este campo especifica el nombre de salida de recepción de canal.

Si este nombre no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente antes de que se procesen los datos de red recibidos.

  A la salida se le proporciona el almacenamiento intermedio de transmisión completo tal como se ha recibido. El contenido del almacenamiento intermedio se puede modificar según sea necesario.
- Durante la inicialización y terminación del canal.

Consulte "MQCD-Definición de canal" en la página 1035 para obtener una descripción del contenido de este campo en varios entornos.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

ReceiveExitPtr (MQPTR)
Este campo especifica la dirección del primer campo *ReceiveExit* .

Si *ReceiveExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de nombres de cada salida de recepción de canal de la cadena.

Cada nombre está en un campo de longitud *ExitNameLength*, rellenado a la derecha con espacios en blanco. Hay *ReceiveExitsDefined* campos que se unen entre sí-uno para cada salida.

Los cambios realizados en estos nombres por una salida se conservan, aunque la salida de canal de mensajes no realiza ninguna acción explícita-no cambia qué salidas se invocan.

Si *ReceiveExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

ReceiveExitsdefinido (MQLONG)
Este campo especifica el número de salidas de recepción de canal definidas en la cadena.

Es mayor o igual que cero.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

Datos de ReceiveUser(MQCHAR32)
Este canal especifica los datos de usuario de salida de recepción de canal.

Estos datos se pasan a la salida de recepción de canal en el campo *ExitData* del parámetro *ChannelExitParms* (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Esto se aplica a las salidas en diferentes conversaciones. Estos cambios no afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Este campo no es relevante en WebSphere MQ para IBM i.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_2.

ReceiveUserDataPtr (MQPTR)
Este campo especifica la dirección del primer campo *ReceiveUserData* .

Si *ReceiveExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de elementos de datos de usuario para cada salida de recepción de canal de la cadena.

Cada elemento de datos de usuario está en un campo de longitud *ExitDataLength*, rellenado a la derecha con espacios en blanco. Hay *ReceiveExitsDefined* campos que se unen entre sí-uno para cada salida. Si el número de elementos de datos de usuario definidos es menor que el número de nombres de salida, los elementos de datos de usuario no definidos se establecen en blancos. Por el contrario, si el número de elementos de datos de usuario definidos es mayor que el número de nombres de salida, los elementos de datos de usuario en exceso se ignoran y no se presentan a la salida.

Los cambios realizados en estos valores por una salida se conservan. Esto permite que una salida pase información a otra salida. No se realiza ninguna validación en ningún cambio, por lo que, por ejemplo, los datos binarios se pueden escribir en estos campos si es necesario.

Si *ReceiveExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_5.

RemotePassword (MQCHAR12)
Este campo especifica la contraseña de un socio.

Este campo sólo contiene información válida si *ChannelType* es MQCHT_CLNTCONN o MQCHT_SVRCONN.

- Para una salida de seguridad en un canal MQCHT_CLNTCONN, esta contraseña es una contraseña que se ha obtenido del entorno. La salida puede elegir enviarla a la salida de seguridad en el servidor.
- Para una salida de seguridad en un canal MQCHT_SVRCONN, este campo puede contener una contraseña que se ha obtenido del entorno en el cliente, si no hay ninguna salida de seguridad de cliente. La salida puede utilizar esta contraseña para validar el identificador de usuario en *RemoteUserIdentifier*.

  Si hay una salida de seguridad en el cliente, esta información se puede obtener en un flujo de seguridad del cliente.

La longitud de este campo la proporciona MQ_PASSWORD_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_2.

RemoteSecurityId (MQBYTE40)
Este campo especifica el identificador de seguridad para el usuario remoto.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_CLNTCONN o MQCHT_SVRCONN.

El siguiente valor especial indica que no hay ningún identificador de seguridad:

MQSID_NONE
No se ha especificado ningún identificador de seguridad.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQSID_NONE_ARRAY; esta constante tiene el mismo valor que MQSID_NONE, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para la salida. La longitud de este campo la proporciona MQ_SECURITY_ID_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_6.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCD_VERSION_7.

Identificador RemoteUser(MQCHAR12)
Este campo especifica los primeros 12 bytes de un identificador de usuario de un socio.

Hay dos campos que contienen el identificador de usuario remoto:

- *RemoteUserIdentifier* contiene los primeros 12 bytes del identificador de usuario remoto y se rellena con espacios en blanco si el identificador es inferior a 12 bytes. *RemoteUserIdentifier* puede estar en blanco.
- *LongRemoteUserIdPtr* apunta al identificador de usuario remoto completo, que puede tener más de 12 bytes. Su longitud la proporciona *LongRemoteUserIdLength*. El identificador completo no contiene blancos de cola y no termina en nulo. Si el identificador está en blanco, *LongRemoteUserIdLength* es cero y el valor de *LongRemoteUserIdPtr* no está definido.

  LongRemoteUserIdPtr no está presente si *Version* es menor que MQCD_VERSION_6.

El identificador de usuario remoto sólo es relevante para los canales con un *ChannelType* de MQCHT_CLNTCONN o MQCHT_SVRCONN.

- Para una salida de seguridad en un canal MQCHT_CLNTCONN, este valor es un identificador de usuario que se ha obtenido del entorno. La salida puede elegir enviarla a la salida de seguridad en el servidor.
- Para una salida de seguridad en un canal MQCHT_SVRCONN, este campo puede contener un identificador de usuario que se ha obtenido del entorno en el cliente, si no hay ninguna salida de seguridad de cliente. La salida puede validar este ID de usuario (posiblemente con la contraseña en *RemotePassword*) y actualizar el valor en *MCAUserIdentifier*.

  Si hay una salida de seguridad en el cliente, esta información se puede obtener en un flujo de seguridad del cliente.

La longitud de este campo la proporciona MQ_USER_ID_LENGTH. Este campo no está presente si *Version* es menor que MQCD_VERSION_2.

SecurityExit (MQCHARn)
Este campo especifica el nombre de salida de seguridad de canal.

Si este nombre no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente después de establecer un canal.

  Antes de transferir ningún mensaje, se da la oportunidad a la salida de causar flujos de mensajes de seguridad para validar la autorización de conexión.
- Al recibir una respuesta a un flujo de mensajes de seguridad.

Los flujos de mensajes de seguridad recibidos del procesador remoto en la máquina remota se asignan a la salida.

- Durante la inicialización y terminación del canal.

Consulte"MQCD-Definición de canal" en la página 1035 para obtener una descripción del contenido de este campo en varios entornos.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

Datos de SecurityUser(MQCHAR32)
Este canal especifica los datos de usuario de salida de seguridad de canal.

Estos datos se pasan a la salida de seguridad de canal en el campo *ExitData* del parámetro *ChannelExitParms* (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Esto se aplica a las salidas en diferentes conversaciones. Estos cambios no afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Este campo no es relevante en WebSphere MQ para IBM i.

SendExit (MQCHARn)
Este campo especifica el nombre de salida de emisión de canal.

Si este nombre no está en blanco, se llama a la salida en las horas siguientes:

- Inmediatamente antes de que se envíen los datos en la red.

  A la salida se le proporciona el almacenamiento intermedio de transmisión completo antes de que se transmita. El contenido del almacenamiento intermedio se puede modificar según sea necesario.
- Durante la inicialización y terminación del canal.

Consulte "MQCD-Definición de canal" en la página 1035 para obtener una descripción del contenido de este campo en varios entornos.

La longitud de este campo la proporciona MQ_EXIT_NAME_LENGTH.

Nota: El valor de esta constante es específico del entorno.

SendExitPtr (MQPTR)
Este campo especifica la dirección del primer campo *SendExit* .

Si *SendExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de nombres de cada salida de envío de canal de la cadena.

Cada nombre está en un campo de longitud *ExitNameLength*, rellenado a la derecha con espacios en blanco. Hay *SendExitsDefined* campos que se unen entre sí-uno para cada salida.

Los cambios realizados en estos nombres por una salida se conservan, aunque la salida de envío de mensajes no realiza ninguna acción explícita-no cambia qué salidas se invocan.

Si *SendExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

SendExitsdefinidos (MQLONG)
Este campo especifica el número de salidas de envío de canal definidas en la cadena.

Es mayor o igual que cero.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

Datos de SendUser(MQCHAR32)
Este campo especifica los datos de usuario de salida de emisión de canal.

Estos datos se pasan a la salida de emisión de canal en el campo *ExitData* del parámetro *ChannelExitParms* (consulte MQ_CHANNEL_EXIT).

Este campo contiene inicialmente los datos que se han establecido en la definición de canal. Sin embargo, durante el tiempo de vida de esta instancia de MCA, el MCA conserva los cambios realizados en el contenido de este campo por una salida de cualquier tipo, y los hace visibles para las invocaciones subsiguientes de salidas (independientemente del tipo) para esta instancia de MCA. Esto se aplica a las salidas en diferentes conversaciones. Estos cambios no afectan a la definición de canal utilizada por otras instancias de MCA. Se puede utilizar cualquier carácter (incluidos los datos binarios).

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Este campo no es relevante en WebSphere MQ para IBM i.

SendUserDataPtr (MQPTR)
Este campo especifica la dirección del campo *SendUserData* .

Si *SendExitsDefined* es mayor que cero, esta dirección es la dirección de la lista de elementos de datos de usuario para cada salida de mensaje de canal de la cadena.

Cada elemento de datos de usuario está en un campo de longitud *ExitDataLength*, rellenado a la derecha con espacios en blanco. Hay *MsgExitsDefined* campos que se unen entre sí-uno para cada salida. Si el número de elementos de datos de usuario definidos es menor que el número de nombres de salida, los elementos de datos de usuario no definidos se establecen en blancos. Por el contrario, si el número de elementos de datos de usuario definidos es mayor que el número de nombres de salida, los elementos de datos de usuario en exceso se ignoran y no se presentan a la salida.

Los cambios realizados en estos valores por una salida se conservan. Esto permite que una salida pase información a otra salida. No se realiza ninguna validación en ningún cambio, por lo que, por ejemplo, los datos binarios se pueden escribir en estos campos si es necesario.

Si *SendExitsDefined* es cero, este campo es el puntero nulo.

En plataformas en las que el lenguaje de programación no soporta el tipo de datos de puntero, este campo se declara como una serie de bytes de la longitud adecuada.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_4.

SeqNumberEnvolver (MQLONG)
Este campo especifica el número de secuencia de mensaje más alto permitido.

Cuando se alcanza este valor, los números de secuencia se reinician para empezar de nuevo en 1.

Este valor no es negociable y debe coincidir en las definiciones de canal local y remoto.

Este campo no es relevante para los canales con un *ChannelType* de MQCHT_SVRCONN o MQCHT_CLNTCONN.

SharingConversations (MQLONG)
Este campo especifica el número máximo de conversaciones que pueden compartir una instancia de canal asociada con este canal.

Este campo se utiliza en la conexión de cliente y en los canales de conexión de servidor.

Un valor de 0 significa que el canal funciona como lo hacía en versiones anteriores a WebSphere MQ Versión 7.0 con respecto a los atributos siguientes:

- Compartimiento de conversaciones
- Lectura hacia adelante
- STOP CHANNEL(<channelname>) MODE(QUIESCE)
- Pulsaciones
- Consumo asíncrono de cliente

Un valor de 1 es el valor mínimo para el comportamiento de WebSphere MQ V7.0 . Aunque sólo se permite una conversación en la instancia de canal, está disponible la lectura anticipada, el consumo asíncrono y el comportamiento de la versión 7 de la pulsación de CLNTCONN-SVRCONN y la detención del canal en reposo.

Es un campo de entrada para la salida. No está presente si *Version* es menor que MQCD_VERSION_9.

El valor predeterminado de este campo es 10.

Nota: Los límites *MaxInstances* y *MaxInstancesPerClient* aplicados a un canal restringen el número de instancias de canal, no el número de conversaciones que pueden estar compartiendo dichas instancias.

Nombre de ShortConnection(MQCHAR20)
Este campo especifica los primeros 20 bytes de un nombre de conexión.

Si el campo *Version* es MQCD_VERSION_1, *ShortConnectionName* contiene el nombre de conexión completo.

Si el campo *Version* es MQCD_VERSION_2 o superior, *ShortConnectionName* contiene los primeros 20 caracteres del nombre de conexión. El nombre completo de la conexión se proporciona mediante el campo *ConnectionName* ; *ShortConnectionName* y los primeros 20 caracteres de *ConnectionName* son idénticos.

Consulte *ConnectionName* para obtener detalles del contenido de este campo.

Nota: El nombre de este campo se ha cambiado para MQCD_VERSION_2 y las versiones posteriores de MQCD; el campo se denominaba anteriormente *ConnectionName*.

La longitud de este campo la proporciona MQ_SHORT_CONN_NAME_LENGTH.

Recuento de ShortRetry(MQLONG)
Este campo especifica el número máximo de intentos que se realizan para conectarse a una máquina remota.

Este campo es el número máximo de intentos que se realizan para conectarse a la máquina remota, a intervalos especificados por *ShortRetryInterval*, antes de que se utilicen *LongRetryCount* y *LongRetryInterval* (normalmente más largos).

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

ShortRetryIntervalo (MQLONG)
Este campo especifica el número máximo de segundos que se debe esperar antes de volver a intentar la conexión con la máquina remota.

El intervalo entre reintentos se puede ampliar si el canal tiene que esperar a estar activo.

Este campo sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR o MQCHT_CLUSRCVR.

SSLCipherSpec (MQCHAR32)
Este campo especifica la especificación de cifrado que se utiliza cuando se utiliza SSL.

Si SSLCipherSpec está en blanco, el canal no utiliza SSL. Si no está en blanco, este campo contiene una serie que especifica la CipherSpec en uso.

Este parámetro es válido para todos los tipos de canal. Está soportado en AIX, HP-UX, Linux, IBM i, Solaris, Windowsy z/OS. Sólo es válido para tipos de canal de un tipo de transporte (TRPTYPE) de TCP.

Es un campo de entrada para la salida. La longitud de este campo la proporciona MQ_SSL_CIPHER_SPEC_LENGTH. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

SSLClientAuth (MQLONG)
Este campo especifica si es necesaria la autenticación de cliente SSL.

Este campo sólo es relevante para las definiciones de canal SVRCONN.

Es uno de los valores siguientes:

MQSCA_REQUIRED
Se requiere autenticación de cliente.

MQSCA_OPTIONAL
Autenticación de cliente opcional.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

SSLPeerNameLongitud (MQLONG)
Este campo especifica la longitud en bytes del nombre de igual SSL al que apunta *SSLPeerNamePtr*.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

SSLPeerNamePtr (MQPTR)
Este campo especifica la dirección del nombre de igual SSL.

Cuando se recibe un certificado durante un reconocimiento SSL satisfactorio, el nombre distinguido del asunto del certificado se copia en el campo MQCD al que accede SSLPeerNamePtr al final del canal que recibe el certificado. Sobrescribe el valor SSLPeerName para el canal si este valor está presente en la definición de canal del usuario local. Si se especifica una salida de seguridad en este extremo del canal, recibe el nombre distinguido del certificado de igual en el MQCD.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCD_VERSION_7.

Nota: Las aplicaciones de salida de seguridad creadas antes del release de WebSphere MQ v7.1 pueden requerir actualización. Para obtener más información, consulte Programas de salida de seguridad de canal .

StrucLength (MQLONG)
Este campo especifica la longitud en bytes de la estructura MQCD.

La longitud no incluye ninguna de las series a las que se dirigen los campos de puntero contenidos en la estructura. El valor puede ser uno de los siguientes:

MQCD_LENGTH_4
Longitud de la estructura de definición de canal version-4 .

MQCD_LENGTH_5
Longitud de la estructura de definición de canal version-5 .

MQCD_LENGTH_6
Longitud de la estructura de definición de canal version-6 .

MQCD_LENGTH_7
Longitud de la estructura de definición de canal version-7 .

MQCD_LENGTH_8
Longitud de la estructura de definición de canal version-8 .

MQCD_LENGTH_9
Longitud de la estructura de definición de canal version-9 .

La constante siguiente especifica la longitud de la versión actual:

MQCD_LONGITUD_ACTUAL
Longitud de la versión actual de la estructura de definición de canal.

Nota: Estas constantes tienen valores que son específicos del entorno.

El campo no está presente si *Version* es menor que MQCD_VERSION_4.

TpName (MQCHAR64)
Este campo especifica el nombre de programa de transacción de LU 6.2 .

Este campo sólo es relevante si el protocolo de transmisión (*TransportType*) es MQXPT_LU62y *ChannelType* no es MQCHT_SVRCONN ni MQCHT_RECEIVER.

Este campo siempre está en blanco en las plataformas en las que la información está contenida en el objeto lateral de comunicaciones.

La longitud de este campo la proporciona MQ_TP_NAME_LENGTH.

TransportType (MQLONG)
Este campo especifica el protocolo de transmisión que se va a utilizar.

El valor no se comprueba si el canal se ha iniciado desde el otro extremo.

Es uno de los valores siguientes:

MQXPT_LU62
Protocolo de transporte LU 6.2 .

MQXPT_TCP
Protocolo de transporte TCP/IP.

MQXPT_NETBIOS
Protocolo de transporte NetBIOS .

Este valor está soportado en los entornos siguientes: Windows.

MQXPT_SPX
Protocolo de transporte SPX.

Este valor está soportado en los entornos siguientes: Windows, además de clientes WebSphere MQ conectados a estos sistemas.

UseDLQ (MQLONG)
Este campo especifica si se utiliza la cola de mensajes no entregados (o la cola de mensajes no entregados) cuando los canales no pueden entregar los mensajes.

Puede contener uno de los valores siguientes:

MQUSEDLQ_NO
Los mensajes que un canal no puede entregar se tratan como un error. El canal descarta el mensaje o el canal finaliza, de acuerdo con el valor NPMSPEED.

MQUSEDLQ_SÍ
Cuando el atributo de gestor de colas DEADQ proporciona el nombre de una cola de mensajes no entregados, se utiliza, de lo contrario, el comportamiento es como para NO. YES es el valor predeterminado.

UserIdentifier (MQCHAR12)
Este campo especifica el identificador de usuario utilizado por el agente de canal de mensajes al intentar iniciar una sesión SNA segura con un agente de canal de mensajes remoto.

Este campo no puede estar en blanco sólo en sistemas UNIX y Windows , y sólo es relevante para canales con un *ChannelType* de MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER o MQCHT_CLNTCONN. En z/OS, este campo no es relevante.

La longitud de este campo la proporciona MQ_USER_ID_LENGTH. Sin embargo, sólo se utilizan los primeros 10 caracteres.

Este campo no está presente cuando *Version* es menor que MQCD_VERSION_2.

Versión (MQLONG)
El campo Version especifica el número de versión más alto que puede establecer para la estructura.

El valor depende del entorno:

MQCD_VERSION_1
Estructura de definición de canal de la versión 1.

MQCD_VERSION_2
Estructura de definición de canal de la versión 2.

La versión 2 no la utiliza ningún producto IBM WebSphere MQ actual.

MQCD_VERSION_3
Estructura de definición de canal de la versión 3.

La versión 3 es la más alta en la que puede establecer el campo en MQSeries Versión 2 en los siguientes entornos: HP Integrity NonStop Servery sistemas UNIX and Linux no listados en otros lugares.

MQCD_VERSION_4
Estructura de definición de canal de la versión 4.

La versión 4 no la utiliza ningún producto IBM WebSphere MQ actual.

MQCD_VERSION_5
Estructura de definición de canal de la versión 5.

La versión 5 es la más alta en la que puede establecer el campo en MQSeries para OS/390 Versión 5 Release 2.

MQCD_VERSION_6
Estructura de definición de canal de la versión 6.

La versión 6 no es la versión actual de la estructura MQCD de ningún producto IBM WebSphere MQ existente. Sin embargo, se puede pasar una estructura MQCD de la versión 6 a MQCONNX utilizando los campos ClientConnOffset o ClientConnPtr de la estructura MQCNO .

En las plataformas distribuidas, la versión 6 es la versión predeterminada en los inicializadores MQCD_DEFAULT y MQCD_CLIENT_CONN_DEFAULT . Si desea hacer referencia a los campos MQCD_VERSION_7, MQCD_VERSION_8o MQCD_VERSION_9 del MQCD, inicialice explícitamente el campo MQCD **Version** en MQCD_VERSION_7, MQCD_VERSION_8o MQCD_VERSION_9 , según corresponda.

En z/OS, MQCD_VERSION_7 es el valor predeterminado.

MQCD_VERSION_7
Estructura de definición de canal de la versión 7.

La versión 7 es la más alta en la que puede establecer el campo en IBM WebSphere MQ Version 5.3 en los entornos siguientes: AIX, HP-UX, Solaris, Windows, y en IBM WebSphere MQ for z/OS Version 5.3 y Version 5.3.1. MQCD_VERSION_7 es el valor predeterminado para las versiones de IBM WebSphere MQ for z/OS.

MQCD_VERSION_8
Estructura de definición de canal de la versión 8.

La versión 8 es la más alta en la que puede establecer el campo en IBM WebSphere MQ Version 6.0 en todas las plataformas.

MQCD_VERSION_9
Estructura de definición de canal de la versión 9.

La versión 9 es la más alta en la que puede establecer el campo en IBM WebSphere MQ Version 7.0 y IBM WebSphere MQ Version 7.0.1 en todas las plataformas.

MQCD_VERSION_10
Estructura de definición de canal de la versión 10.

La versión 10 es la más alta en la que puede establecer el campo en IBM WebSphere MQ Version 7.1 y IBM WebSphere MQ Version 7.5 en todas las plataformas.

Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQCD_CURRENT_VERSION
El valor establecido en MQCD_CURRENT_VERSION es la versión actual de la estructura de definición de canal que se está utilizando.

El valor de MQCD_CURRENT_VERSION depende del entorno. Contiene el valor más alto soportado por la plataforma.

MQCD_CURRENT_VERSION no se utiliza para inicializar las estructuras predeterminadas proporcionadas en los archivos de cabecera, copia e inclusión proporcionados para distintos lenguajes de programación. La inicialización predeterminada de Version depende de la plataforma y del release.

Para IBM WebSphere MQ Version 7.0 y versiones posteriores, las declaraciones MQCD en los archivos de cabecera, copia e inclusión se inicializan en MQCD_VERSION_6. Para utilizar campos MQCD adicionales, las aplicaciones deben establecer el número de versión en MQCD_CURRENT_VERSION. Si está escribiendo una aplicación que es portable entre varios entornos, debe elegir una versión que esté soportada en todos los entornos.

Consejo: Cuando se introduce una nueva versión de la estructura MQCD , el diseño de la parte existente no cambia. La salida debe comprobar el número de versión. Debe ser igual o mayor que la versión más baja que contiene los campos que la salida necesita utilizar.

XmitQName (MQCHAR48)
Este campo especifica el nombre de la cola de transmisión de la que se recuperan los mensajes.

Este campo sólo es relevante para los canales con un *ChannelType* de MQCHT_SENDER o MQCHT_SERVER.

La longitud de este campo la proporciona MQ_Q_NAME_LENGTH.

Declaración C
Esta declaración es la declaración C para la estructura 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];         /* SSL CipherSpec */
   MQPTR     SSLPeerNamePtr;            /* Address of SSL peer name */
   MQLONG    SSLPeerNameLength;         /* Length of SSL peer name */
   MQLONG    SSLClientAuth;             /* Whether SSL 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 */
 };
```

declaración COBOL

Esta declaración es la declaración COBOL para la estructura 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 **
      ** SSL CipherSpec
       15  MQCD-SSLCIPHERSPEC PIC X(32).
      ** Address of SSL peer name
       15  MQCD-SSLPEERNAMEPTR POINTER.
      ** Length of SSL peer name
       15  MQCD-SSLPEERNAMELENGTH PIC S9(9) BINARY.
      ** Whether SSL 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 **
```

Declaración RPG (ILE)

Esta declaración es la declaración RPG para la estructura 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* SSL CipherSpec
     D  CDSCS               1217   1248
     D* Address of SSL peer name
     D  CDSPN               1249   1264*
     D* Length of SSL peer name
     D  CDSPL               1265   1268I 0
     D* Whether SSL 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 **
```

System/390

Esta declaración es la declaración de ensamblador System/390 para la estructura 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   SSL CipherSpec
MQCD_SSLPEERNAMEPTR          DS   F      Address of SSL peer name
MQCD_SSLPEERNAMELENGTH       DS   F      Length of SSL peer name
MQCD_SSLCLIENTAUTH           DS   F      Whether SSL 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_LENGTH                  EQU  *-MQCD
                             ORG  MQCD
MQCD_AREA                    DS   CL(MQCD_LENGTH)
```

Declaración de Visual Basic

Esta declaración es la declaración de Visual Basic de la estructura MQCD.

En Visual Basic, la estructura MQCD se puede utilizar con la estructura MQCNO en la llamada 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  'SSL CipherSpec'
  SSLPeerNamePtr         As MQPTR      'Address of SSL peer name'
  SSLPeerNameLength      As Long       'Length of SSL peer name'
  SSLClientAuth          As Long       'Whether SSL 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
```

Cambio de campos MQCD en una salida de canal

Una salida de canal pueden cambiar los campos del MQCD. Sin embargo, estos cambios normalmente no se realizan, excepto en las circunstancias listadas.

Si un programa de salida de canal cambia un campo en la estructura de datos MQCD, el nuevo valor normalmente lo ignora el proceso de canal WebSphere MQ . Sin embargo, el nuevo valor permanece en el MQCD y se le pasa a las salidas restantes en una cadena de salida y a cualquier conversación que comparta la instancia de canal.

Si SharingConversations se establece en FALSE en la estructura MQCXP, se pueden realizar cambios en determinados campos, en función del tipo de programa de salida, el tipo de canal y el código de razón de salida. La tabla siguiente muestra los campos que se pueden cambiar y afectan al comportamiento del canal, y en qué circunstancias. Si un programa de salida cambia uno de estos campos en cualquier otra circunstancia, o cualquier campo no listado, el nuevo valor es ignorado por el proceso de canal. El nuevo valor permanece en el MQCD y se pasa a las salidas restantes de una cadena de salida y a cualquier conversación que comparta la instancia de canal.

Cualquier tipo de programa de salida cuando se llama para la inicialización (MQXR_INIT) puede cambiar el campo ChannelName de cualquier tipo de canal, siempre que MQCXP SharingConverstions esté establecido en FALSE. Sólo una salida de seguridad puede cambiar el campo MCAUserIdentifier , independientemente del valor de MQCXP SharingConverstions.

Campo	**Código de razón de salida**	**Tipo de salida**	**ChannelType**
ChannelName	MQXR_INIT	Todo	Todo
TransportType	MQXR_INIT	Todo	Todo
XmitQName	MQXR_INIT	Todo	SDR, RCVR
ModeName	MQXR_INIT	Todo	Todo
TpName	MQXR_INIT	Todo	Todo

Campo	Código de razón de salida	Tipo de salida	ChannelType
BatchSize	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
DiscInterval	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Recuento de ShortRetry	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
shortRetryInterval	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Recuento de LongRetry	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
longRetryInterval	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
SeqNumberAjustar	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
MaxMsgLength	MQXR_INIT	Todo	Todo
PutAuthority	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
DataConversion	MQXR_INIT	Todo	Todo
MCAUserIdentifier	MQXR_INIT, MQXR_INIT_SEC, MQXR_SEC_MSG, MQXR_SEC_PARMS	Seguridad	RCVR, RQSTR, SVRCONN, CLUSRCVR

Campo	Código de razón de salida	Tipo de salida	ChannelType
ConnectionName	MQXR_INIT	Todo	SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, CLUSRCVR
MsgRetryUserData	MQXR_INIT	Todo	RCVR, RQSTR, CLUSRCVR
Recuento de MsgRetry	MQXR_INIT	Todo	RCVR, RQSTR, CLUSRCVR
Intervalo de MsgRetry	MQXR_INIT	Todo	RCVR, RQSTR, CLUSRCVR
HeartbeatInterval	MQXR_INIT	Todo	Todo
BatchInterval	MQXR_INIT	Todo	SDR, SVR, CLUSSDR, CLUSRCVR
NonPersistentMsgSpeed	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
MCASecurityId	MQXR_INIT, MQXR_INIT_SEC, MQXR_SEC_MSG, MQXR_SEC_PARMS	Seguridad	SDR, SVR, RCVR, RQSTR, SVRCONN, CLUSSDR, CLUSRCVR
SSLCipherSpec	MQXR_INIT	Todo	Todo
SSLPeerNamePtr	MQXR_INIT	Todo	Todo
SSLPeerNameLongitud	MQXR_INIT	Todo	Todo
SSLClientAuth	MQXR_INIT	Todo	SVR, RCVR, RQSTR, SVRCONN, CLUSRCVR
Intervalo de mantenimiento de activación	MQXR_INIT	Todo	Todo
LocalAddress	MQXR_INIT	Todo	SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, CLUSRCVR

Campo	Código de razón de salida	Tipo de salida	ChannelType
BatchHeartbeat	MQXR_INIT	Todo	SDR, SVR, CLUSSDR, CLUSRCVR
Lista de HdrComp	MQXR_INIT	Todo	Todo
Lista de MsgComp	MQXR_INIT	Todo	Todo
ChannelMonitoring	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, SVRCONN, CLUSSDR, CLUSRCVR
ChannelStatistics	MQXR_INIT	Todo	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
SharingConversations	MQXR_INIT	Todo	SVRCONN, CLNTCONN
PropertyControl	MQXR_INIT	Todo	SDR, SVR, CLUSSDR, CLUSRCVR

MQCXP-Parámetro de salida de canal

La estructura MQCXP se pasa a cada tipo de salida llamada por un agente de canal de mensajes (MCA), un canal de conexión de cliente o un canal de conexión de servidor.

Consulte MQ_CHANNEL_EXIT.

Los campos descritos como "entrada a la salida" en las descripciones que siguen son ignorados por el canal cuando la salida devuelve el control al canal. Los campos de entrada que la salida cambie en el bloque de parámetros de salida de canal no se conservarán para su siguiente invocación. Los cambios realizados en los campos de entrada/salida (por ejemplo, el campo *ExitUserArea*) se conservan sólo para las invocaciones de esa instancia de la salida. Estos cambios no se pueden utilizar para pasar datos entre salidas diferentes definidas en el mismo canal, o entre la misma salida definida en canales diferentes.

Referencia relacionada

“Campos” en la página 1075
Este tema lista todos los campos de la estructura MQCXP y describe cada campo.

“Declaración C” en la página 1086
Esta declaración es la declaración C para la estructura MQCXP.

“declaración COBOL” en la página 1087
Esta declaración es la declaración COBOL para la estructura MQCXP.

“Declaración RPG (ILE)” en la página 1088
Esta declaración es la declaración RPG para la estructura MQCXP.

“ System/390” en la página 1089
Esta declaración es la declaración de ensamblador System/390 para la estructura MQCXP.

Campos

Este tema lista todos los campos de la estructura MQCXP y describe cada campo.

StrucId (MQCHAR4)
Este campo especifica el identificador de estructura.

El valor debe ser:

MQCXP_STRUC_ID
Identificador de la estructura de parámetros de salida de canal.

Para el lenguaje de programación C, también se define la constante MQCXP_STRUC_ID_ARRAY; esta constante tiene el mismo valor que MQCXP_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

Es un campo de entrada para la salida.

Versión (MQLONG)
Este campo especifica el número de versión de la estructura.

El valor depende del entorno:

MQCXP_VERSION_1
Estructura del parámetro de salida de canal Version-1 .

MQCXP_VERSION_2
Estructura del parámetro de salida de canal Version-2 .

El campo tiene este valor en los entornos siguientes: HP Integrity NonStop Server.

MQCXP_VERSION_3
Estructura del parámetro de salida de canal Version-3 .

El campo tiene este valor en los entornos siguientes: sistemas UNIX no listados en otros lugares.

MQCXP_VERSION_4
Estructura del parámetro de salida de canal Version-4 .

MQCXP_VERSION_5
Estructura del parámetro de salida de canal Version-5 .

MQCXP_VERSION_6
Estructura del parámetro de salida de canal Version-6 .

MQCXP_VERSION_8
Estructura del parámetro de salida de canal Version-8 .

El campo tiene este valor en los entornos siguientes: z/OS, AIX, HP-UX, Linux, IBM i, Solaris, Windows.

Los campos que existen sólo en las versiones más recientes de la estructura se identifican como tales en las descripciones de los campos. La constante siguiente especifica el número de versión de la versión actual:

MQCXP_CURRENT_VERSION
Versión actual de la estructura de parámetros de salida de canal.

El valor depende del entorno.

Nota: Cuando se introduce una nueva versión de la estructura MQCXP, el diseño de la parte existente no cambia. Por lo tanto, la salida debe comprobar que el número de versión es igual o mayor que la versión más baja que contiene los campos que la salida necesita utilizar.

Es un campo de entrada para la salida.

ExitId (MQLONG)
Este campo especifica el tipo de salida que se llama y se establece en la entrada de la rutina de salida.

Son posibles los siguientes valores:

MQXT_CHANNEL_SEC_EXIT
Salida de seguridad de canal.

MQXT_CHANNEL_MSG_EXIT
Salida de mensajes de canal.

MQXT_CHANNEL_SEND_EXIT
Salida de envío de canal.

MQXT_CHANNEL_RCV_EXIT
Salida de recepción de canal.

MQXT_CHANNEL_MSG_RETRY_EXIT
Salida de reintento de mensaje de canal.

MQXT_CHANNEL_AUTO_DEF_EXIT
Salida de definición automática de canal.

En z/OS, este tipo de salida sólo está soportado para canales de tipo MQCHT_CLUSSDR y MQCHT_CLUSRCVR.

Es un campo de entrada para la salida.

ExitReason (MQLONG)
Este campo especifica la razón por la que se llama a la salida y se establece en la entrada de la rutina de salida.

La salida de definición automática no la utiliza. Son posibles los siguientes valores:

MQXR_INIT
Salga de la inicialización.

Este valor indica que la salida se está invocando por primera vez. Permite que la salida adquiera e inicialice los recursos que necesite (por ejemplo: memoria).

MQXR_TERM
Terminación de salida.

Este valor indica que la salida está a punto de terminar. La salida debe liberar los recursos que haya adquirido desde que se inicializó (por ejemplo: memoria).

MQXR_MSG
Procesar un mensaje.

Este valor indica que se está invocando la salida para procesar un mensaje. Este valor sólo se produce para salidas de mensajes de canal.

MQXR_XMIT
Procesar una transmisión.

Este valor sólo se produce para salidas de envío y recepción de canal.

MQXR_SEC_MSG
Se ha recibido el mensaje de seguridad.

Este valor sólo se produce para salidas de seguridad de canal.

MQXR_INIT_SEC
Inicie el intercambio de seguridad.

Este valor sólo se produce para salidas de seguridad de canal.

La salida de seguridad del receptor siempre se invoca con esta razón inmediatamente después de ser invocada con MQXR_INIT, para darle la oportunidad de iniciar un intercambio de seguridad. Si rechaza la oportunidad (devolviendo MQXCC_OK en lugar de MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG), la salida de seguridad del remitente se invoca con MQXR_INIT_SEC.

Si la salida de seguridad del receptor no inicia un intercambio de seguridad (devolviendo MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG), la salida de seguridad del remitente nunca se invoca con MQXR_INIT_SEC; en su lugar, se invoca con MQXR_SEC_MSG para procesar el mensaje del receptor. (En cualquier caso, se invoca por primera vez con MQXR_INIT.)

A menos que una de las salidas de seguridad solicite la terminación del canal (estableciendo *ExitResponse* en MQXCC_SUPPRESS_FUNCTION o MQXCC_CLOSE_CHANNEL), el intercambio de seguridad debe completarse en el lado que ha iniciado el intercambio. Por lo tanto, si se invoca una salida de seguridad con MQXR_INIT_SEC y inicia un intercambio, la próxima vez que se invoque la salida será con MQXR_SEC_MSG. Esto sucede si hay un mensaje de seguridad para que la salida se procese o no. Hay un mensaje de seguridad si el socio devuelve MQXCC_SEND_SEC_MSG o MQXCC_SEND_AND_REQUEST_SEC_MSG, pero no si el socio devuelve MQXCC_OK o no hay ninguna salida de seguridad en el socio. Si no hay ningún mensaje de seguridad para procesar, la salida de seguridad en el extremo de inicio se vuelve a invocar con un *DataLength* de cero.

MQXR_REINTENTAR
Vuelva a intentar un mensaje.

Este valor sólo se produce para salidas de reintento de mensaje.

MQXR_AUTO_CLUSSDR
Definición automática de un canal de clúster emisor.

Este valor sólo se produce para salidas de definición automática de canal.

MQXR_AUTO_RECEIVER
Definición automática de un canal receptor.

Este valor sólo se produce para salidas de definición automática de canal.

MQXR_AUTO_SVRCONN
Definición automática de un canal de conexión de servidor.

Este valor sólo se produce para salidas de definición automática de canal.

MQXR_AUTO_CLUSRCVR
Definición automática de un canal de clúster receptor.

Este valor sólo se produce para salidas de definición automática de canal.

MQXR_SEC_PARMS
Parámetros de seguridad

Este valor sólo se aplica a las salidas de seguridad e indica que se está pasando una estructura MQCSP a la salida. Para obtener más información, consulte "MQCSP-Parámetros de seguridad" en la página 313.

Nota:

1. Si tiene más de una salida definida para un canal, cada uno de ellos se invoca con MQXR_INIT cuando se inicializa el MCA. Además, cada uno de ellos se invoca con MQXR_TERM cuando termina el MCA.
2. Para la salida de definición automática de canal, *ExitReason* no se establece si *Version* es menor que MQCXP_VERSION_4. El valor MQXR_AUTO_SVRCONN está implícito en este caso.

Es un campo de entrada para la salida.

ExitResponse (MQLONG)
Este campo especifica la respuesta de la salida.

Este campo lo establece la salida para comunicarse con el MCA.Tiene que ser uno de los valores siguientes:

MQXCC_Correcto
La salida se ha completado correctamente.

- Para la salida de seguridad de canal, este valor indica que la transferencia de mensajes ahora puede continuar con normalidad.
- Para la salida de reintento de mensaje de canal, este valor indica que el MCA debe esperar el intervalo de tiempo devuelto por la salida en el campo *MsgRetryInterval* en MQCXP y, a continuación, vuelva a intentar el mensaje.

El campo *ExitResponse2* puede contener información adicional.

MQXCC_SUPPRESS_FUNCTION
Suprimir función.

- Para la salida de seguridad de canal, este valor indica que el canal debe terminar.
- Para la salida de mensajes de canal, este valor indica que el mensaje no debe continuar hacia su destino. En su lugar, el MCA genera un mensaje de informe de excepción (si lo ha solicitado el remitente del mensaje original) y coloca el mensaje contenido en el almacenamiento intermedio original en la cola de mensajes no entregados (si el remitente ha especificado MQRO_DEAD_LETTER_Q), o lo descarta (si el remitente ha especificado MQRO_DISCARD_MSG).

  Para los mensajes persistentes, si el remitente ha especificado MQRO_DEAD_LETTER_Q, pero la colocación en la cola de mensajes no entregados falla, o no hay ninguna cola de mensajes no entregados, el mensaje original se deja en la cola de transmisión y el mensaje de informe no se genera. El mensaje original también se deja en la cola de transmisión si el mensaje de informe no se puede generar correctamente.

  El campo *Feedback* de la estructura MQDLH al principio del mensaje en la cola de mensajes no entregados indica por qué el mensaje se ha colocado en la cola de mensajes no entregados; este código de retorno también se utiliza en el descriptor de mensaje del mensaje de informe de excepción (si el remitente ha solicitado uno).
- Para la salida de reintento de mensaje de canal, este valor indica que el MCA no espera y vuelve a intentar el mensaje; en su lugar, el MCA continúa inmediatamente con su proceso de anomalía normal (el mensaje se coloca en la cola de mensajes no entregados o se descarta, tal como especifica el emisor del mensaje).
- Para la salida de definición automática de canal, debe especificarse MQXCC_OK o MQXCC_SUPPRESS_FUNCTION. Si no se especifica ninguno de estos valores, se presupone MQXCC_SUPPRESS_FUNCTION de forma predeterminada y se abandona la definición automática.

Esta respuesta no está soportada para las salidas de envío y recepción de canal.

MQXCC_SEND_SEC_MSG
Enviar mensaje de seguridad.

Este valor sólo lo puede establecer una salida de seguridad de canal. Indica que la salida ha proporcionado un mensaje de seguridad que debe transmitirse al socio.

MQXCC_SEND_AND_REQUEST_SEC_MSG
Envíe un mensaje de seguridad que requiera una respuesta.

Este valor sólo lo puede establecer una salida de seguridad de canal. Indica

- que la salida ha proporcionado un mensaje de seguridad que puede transmitirse al socio, y
- que la salida requiere una respuesta del socio. Si no se recibe ninguna respuesta, el canal debe terminar, porque la salida todavía no ha decidido si las comunicaciones pueden continuar.

MQXCC_SUPPRESS_EXIT
Suprimir salida.

- Este valor lo pueden establecer todos los tipos de salida de canal que no sean una salida de seguridad o una salida de definición automática. Suprime cualquier invocación adicional de esa salida (como si su nombre hubiera estado en blanco en la definición de canal), hasta la terminación del canal, cuando se vuelve a invocar la salida con un *ExitReason* de MQXR_TERM.
- Si una salida de reintento de mensaje devuelve este valor, los reintentos de mensaje para los mensajes posteriores se controlan mediante los atributos de canal *MsgRetryCount* y *MsgRetryInterval* como normales. Para el mensaje actual, el MCA realiza el número de reintentos pendientes, a intervalos proporcionados por el atributo de canal *MsgRetryInterval* , pero sólo si el código de razón es uno que el MCA reintentaría normalmente (consulte el campo *MsgRetryCount* descrito en "MQCD-Definición de canal" en la página 1035). El número de reintentos pendientes es el valor del atributo *MsgRetryCount* , menos el número de veces que la salida ha devuelto MQXCC_OK para el mensaje actual; si este número es negativo, el MCA no realiza más reintentos para el mensaje actual.

MQXCC_CLOSE_CHANNEL
Cierre el canal.

Este valor lo puede establecer cualquier tipo de salida de canal excepto una salida de definición automática.

Si la compartición de conversaciones no está habilitada, este valor cierra el canal.

Si la compartición de conversaciones está habilitada, este valor finaliza la conversación. Si esta conversación es la única conversación en el canal, el canal también se cierra.

Este campo es un campo de entrada/salida de la salida.

ExitResponse2 (MQLONG)
Este campo especifica la respuesta secundaria de la salida.

Este campo se establece en cero en la entrada a la rutina de salida.Se puede establecer mediante la salida para proporcionar más información a las funciones de canal de WebSphere MQ . La salida de definición automática no la utiliza.

La salida puede establecer uno o varios de los valores siguientes. Si se necesita más de uno, se añaden los valores. Las combinaciones que no son válidas se anotan; se permiten otras combinaciones.

MQXR2_PUT_WITH_DEF_ACTION
Poner con acción predeterminada.

Este valor lo establece la salida de mensajes de canal del receptor. Indica que el mensaje debe colocarse con la acción predeterminada del MCA, que es el ID de usuario predeterminado del MCA, o el contexto *UserIdentifier* en el MQMD (descriptor de mensaje) del mensaje.

El valor es cero, que corresponde al valor inicial establecido cuando se invoca la salida. La constante se proporciona a efectos de documentación.

MQXR2_PUT_WITH_DEF_USERID
Colocar con identificador de usuario predeterminado.

Este valor sólo lo puede establecer la salida de mensajes de canal del receptor. Indica que el mensaje debe colocarse con el identificador de usuario predeterminado del MCA.

MQXR2_PUT_WITH_MSG_USERID
Transferir con el identificador de usuario del mensaje.

Este valor sólo lo puede establecer la salida de mensajes de canal del receptor. Indica que el mensaje debe colocarse con el contexto *UserIdentifier* en el MQMD (descriptor de mensaje) del mensaje (esto puede haber sido modificado por la salida).

Solo debe establecerse uno de MQXR2_PUT_WITH_DEF_ACTION, MQXR2_PUT_WITH_DEF_USERIDy MQXR2_PUT_WITH_MSG_USERID .

MQXR2_USE_AGENT_BUFFER
Utilice el almacenamiento intermedio del agente.

Este valor indica que los datos que deben pasarse están en *AgentBuffer*, no en *ExitBufferAddr*.

El valor es cero, que corresponde al valor inicial establecido cuando se invoca la salida. La constante se proporciona a efectos de documentación.

MQXR2_USE_EXIT_BUFFER
Utilice el almacenamiento intermedio de salida.

Este valor indica que los datos que deben pasarse están en *ExitBufferAddr*, no en *AgentBuffer*.

Solo se debe establecer uno de MQXR2_USE_AGENT_BUFFER y MQXR2_USE_EXIT_BUFFER .

MQXR2_DEFAULT_CONTINUATION
Continuación predeterminada.

La continuación con la siguiente salida de la cadena depende de la respuesta de la última salida invocada:

- Si se devuelve MQXCC_SUPPRESS_FUNCTION o MQXCC_CLOSE_CHANNEL, no se llama a más salidas de la cadena.
- De lo contrario, se invoca la siguiente salida de la cadena.

MQXR2_CONTINUE_CHAIN
Continúe con la siguiente salida.

MQXR2_SUPPRESS_CHAIN
Omita las salidas restantes de la cadena.

Es un campo de entrada/salida para la salida.

Feedback (MQLONG)
Este campo especifica el código de comentarios.

Este campo se establece en MQFB_NONE en la entrada a la rutina de salida.

Si una salida de mensajes de canal establece el campo *ExitResponse* en MQXCC_SUPPRESS_FUNCTION, el campo *Feedback* especifica el código de retorno que identifica por qué el mensaje se ha colocado en la cola de mensajes no entregados (undelivered-message) y también se utiliza para enviar un informe de excepción si se ha solicitado uno. En este caso, si el campo *Feedback* es MQFB_NONE, se utiliza el siguiente código de retorno:

MQFB_STOPPED_BY_MSG_EXIT
Mensaje detenido por salida de mensajes de canal.

El MCA no utiliza el valor devuelto en este campo por las salidas de seguridad de canal, envío, recepción y reintento de mensaje.

El valor devuelto en este campo por las salidas de definición automática no se utiliza si *ExitResponse* es MQXCC_OK, pero de lo contrario se utiliza para el parámetro *AuxErrorDataInt1* en el mensaje de suceso.

Es un campo de entrada/salida de la salida.

MaxSegmentLongitud (MQLONG)
Este campo especifica la longitud máxima en bytes que se puede enviar en una sola transmisión.

La salida de definición automática no la utiliza. Es de interés para una salida de emisión de canal, porque esta salida debe asegurarse de que no aumenta el tamaño de un segmento de transmisión a un valor mayor que *MaxSegmentLength*. La longitud incluye los 8 bytes iniciales que la salida no debe cambiar. El valor se negocia entre las funciones de canal de WebSphere MQ cuando se inicia el canal. Consulte Escritura de programas de salida de canal para obtener más información sobre las longitudes de segmento.

El valor de este campo no es significativo si *ExitReason* es MQXR_INIT.

Es un campo de entrada para la salida.

Área ExitUser(MQBYTE16)
Este campo especifica el área de usuario de salida: un campo disponible para que lo utilice la salida.

Se inicializa a cero binario antes de la primera invocación de la salida (que tiene un *ExitReason* establecido en MQXR_INIT) y, a partir de entonces, los cambios realizados en este campo por la salida se conservan entre las invocaciones de la salida.

Se define el valor siguiente:

MQXUA_NONE
No hay información de usuario.

El valor es cero binario para la longitud del campo.

Para el lenguaje de programación C, también se define la constante MQXUA_NONE_ARRAY; esta constante tiene el mismo valor que MQXUA_NONE, pero es una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_EXIT_USER_AREA_LENGTH. Es un campo de entrada/salida para la salida.

ExitData (MQCHAR32)
Este campo especifica los datos de salida.

Este campo se establece en la entrada de la rutina de salida en la información que las funciones de canal de WebSphere MQ han tomado de la definición de canal.Si no hay dicha información disponible, este campo está en blanco.

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

Es un campo de entrada para la salida.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCXP_VERSION_2.

Recuento de MsgRetry(MQLONG)
Este campo especifica el número de veces que se ha reintentado el mensaje.

La primera vez que se invoca la salida para un mensaje determinado, este campo tiene el valor cero (todavía no se han intentado reintentos).En cada invocación posterior de la salida para ese mensaje, el valor se incrementa en uno por el MCA.

Es un campo de entrada para la salida. El valor de este campo no es significativo si *ExitReason* es MQXR_INIT. El campo no está presente si *Version* es menor que MQCXP_VERSION_2.

Intervalo MsgRetry(MQLONG)
Este campo especifica el intervalo mínimo en milisegundos después del cual se reintenta la operación de transferencia.

La primera vez que se invoca la salida para un mensaje determinado, este campo contiene el valor del atributo de canal *MsgRetryInterval* .La salida puede dejar el valor sin modificar o modificarlo para especificar un intervalo de tiempo diferente en milisegundos. Si la salida devuelve MQXCC_OK en *ExitResponse*, el MCA espera al menos este intervalo de tiempo antes de reintentar la operación MQOPEN o MQPUT. El intervalo de tiempo especificado debe ser cero o mayor.

La segunda y siguientes veces que se invoca la salida para ese mensaje, este campo contiene el valor devuelto por la invocación anterior de la salida.

Si el valor devuelto en el campo *MsgRetryInterval* es menor que cero o mayor que 999 999 999 999, y *ExitResponse* es MQXCC_OK, el MCA ignora el campo *MsgRetryInterval* en MQCXP y espera en su lugar el intervalo especificado por el atributo de canal *MsgRetryInterval* .

Es un campo de entrada/salida para la salida. El valor de este campo no es significativo si *ExitReason* es MQXR_INIT. El campo no está presente si *Version* es menor que MQCXP_VERSION_2.

Razón MsgRetry(MQLONG)
Este campo especifica el código de razón del intento anterior de colocar el mensaje.

Este campo es el código de razón del intento anterior de colocar el mensaje; es uno de los valores MQRC_*.

Es un campo de entrada para la salida. El valor de este campo no es significativo si *ExitReason* es MQXR_INIT. El campo no está presente si *Version* es menor que MQCXP_VERSION_2.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCXP_VERSION_3.

HeaderLength (MQLONG)
Este campo especifica la longitud de la información de cabecera.

Este campo sólo es relevante para una salida de mensajes y una salida de reintento de mensajes.El valor es la longitud de las estructuras de cabecera de direccionamiento al principio de los datos del mensaje; son la estructura MQXQH, la MQMDE (cabecera de extensión de descripción de mensaje) y (para un

mensaje de lista de distribución) la estructura MQDH y las matrices de registros MQOR y MQPMR que siguen a la estructura MQXQH.

La salida de mensajes puede examinar esta información de cabecera y modificarla si es necesario, pero los datos que devuelve la salida deben estar en el formato correcto. La salida no debe, por ejemplo, cifrar o comprimir los datos de cabecera en el extremo emisor, incluso si la salida de mensaje en el extremo receptor realiza cambios compensatorios.

Si la salida de mensajes modifica la información de cabecera de forma que cambie su longitud (por ejemplo, añadiendo otro destino a un mensaje de lista de distribución), debe cambiar el valor de *HeaderLength* correspondiente antes de devolver.

Es un campo de entrada/salida para la salida. El valor de este campo no es significativo si *ExitReason* es MQXR_INIT. El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

PartnerName (MQCHAR48)
Este campo especifica el nombre del socio.

El nombre del socio, como se indica a continuación:

- Para canales SVRCONN, es el ID de usuario conectado en el cliente.
- Para todos los demás tipos de canal, es el nombre del gestor de colas del socio.

Cuando se inicializa la salida, este campo está en blanco porque el gestor de colas no conoce el nombre del socio hasta después de que se haya realizado la negociación inicial.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

FAPLevel (MQLONG)
Nivel de protocolos y formatos negociados.

Es un campo de entrada para la salida. Los cambios en este campo sólo se deben realizar bajo la dirección del servicio IBM .El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

CapabilityFlags (MQLONG)
Este campo especifica los distintivos de capacidad.

Se definen los siguientes elementos:

MQCF_NONE
Sin distintivos.

MQCF_DIST_LISTS
Listas de distribución soportadas.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

ExitNumber (MQLONG)
Este campo especifica el número ordinal de la salida.

El número ordinal de la salida, dentro del tipo definido en *ExitId*.Por ejemplo, si la salida que se invoca es la tercera salida de mensajes definida, este campo contiene el valor 3. Si el tipo de salida es uno para el que no se puede definir una lista de salidas (por ejemplo, una salida de seguridad), este campo tiene el valor 1.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_3.

Los campos siguientes de esta estructura no están presentes si *Version* es menor que MQCXP_VERSION_5.

ExitSpace (MQLONG)
Este campo especifica el número de bytes en el almacenamiento intermedio de transmisión reservado para que lo utilice la salida.

Este campo sólo es relevante para una salida de envío.Especifica la cantidad de espacio en bytes que las funciones de canal de WebSphere MQ reservan en el almacenamiento intermedio de transmisión para que lo utilice la salida. Este campo permite que la salida añada al almacenamiento intermedio de transmisión una pequeña cantidad de datos (normalmente no más de unos pocos cientos de bytes) para que los utilice una salida de recepción complementaria en el otro extremo. La salida de recepción debe eliminar los datos añadidos por la salida de envío.

El valor siempre es cero en z/OS.

Nota: Este recurso no se debe utilizar para enviar grandes cantidades de datos, ya que puede degradar el rendimiento o incluso inhibir el funcionamiento del canal.

Al establecer *ExitSpace* la salida se garantiza que siempre hay al menos ese número de bytes disponibles en el almacenamiento intermedio de transmisión para que los utilice la salida. Sin embargo, la salida puede utilizar menos de la cantidad reservada, o más de la cantidad reservada si hay espacio disponible en el almacenamiento intermedio de transmisión. El espacio de salida en el almacenamiento intermedio se proporciona siguiendo los datos existentes.

La salida sólo puede establecer *ExitSpace* cuando *ExitReason* tiene el valor MQXR_INIT; en todos los demás casos, el valor devuelto por la salida se ignora. En la entrada a la salida, *ExitSpace* es cero para la llamada MQXR_INIT y es el valor devuelto por la llamada MQXR_INIT en otros casos.

Si el valor devuelto por la llamada MQXR_INIT es negativo, o si hay menos de 1024 bytes disponibles en el almacenamiento intermedio de transmisión para los datos de mensaje después de reservar el espacio de salida solicitado para todas las salidas de envío de la cadena, el MCA genera un mensaje de error y cierra el canal. De forma similar, si durante la transferencia de datos las salidas de la cadena de salida de envío asignan más espacio de usuario del que reservaron, de manera que quedan menos de 1024 bytes en el almacenamiento intermedio de transmisión para los datos de mensaje, el MCA genera un mensaje de error y cierra el canal. El límite de 1024 permite que los flujos de control y administrativos del canal sean procesados por la cadena de salidas de envío, sin necesidad de que los flujos sean segmentados.

Es un campo de entrada/salida para la salida si *ExitReason* es MQXR_INIT, y un campo de entrada en todos los demás casos. El campo no está presente si *Version* es menor que MQCXP_VERSION_5.

SSLCertUserId (MQCHAR12)
Este campo especifica el UserId asociado con el certificado remoto.

Está en blanco en todas las plataformas excepto z/OS

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

SSLRemCertIssNameLongitud (MQLONG)
Este campo especifica la longitud en bytes del nombre distinguido completo del emisor del certificado remoto al que apunta SSLCertRemoteIssuerNamePtr.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6. El valor es cero si no es un canal SSL.

SSLRemCertIssNamePtr (PMQVOID)
Este campo especifica la dirección del nombre distinguido completo del emisor del certificado remoto.

Su valor es el puntero nulo si no es un canal SSL.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

Nota: El comportamiento de las salidas de seguridad de canal al determinar el nombre distinguido de asunto y el nombre distinguido de emisor se ha modificado en el release de WebSphere MQ v7.1. Para obtener más información, consulte Programas de salida de seguridad de canal.

SecurityParms (PMQCSP)
Este campo especifica la dirección de la estructura MQSCP utilizada para especificar un ID de usuario y una contraseña.

El valor inicial de este campo es el puntero nulo.

Es un campo de entrada/salida para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

CurHdrCompresión (MQLONG)
Este campo especifica qué técnica se está utilizando actualmente para comprimir los datos de cabecera.

Se establece en uno de los valores siguientes:

MQCOMPRESS_NONE
No se lleva a cabo ninguna compresión de datos de cabecera.

MQCOMPRESS_SISTEMA
Se lleva a cabo la compresión de datos de cabecera.

El valor se puede modificar mediante la salida de mensajes de un canal emisor a uno de los valores soportados negociados a los que se accede desde el campo de lista HdrCompde la MQCD. Esto permite que la técnica utilizada para comprimir los datos de cabecera se elija para cada mensaje basándose en el contenido del mensaje. El valor alterado sólo se utiliza para el mensaje actual. El canal finaliza si el atributo se modifica a un valor no soportado. El valor se ignora si se altera fuera de la salida de mensajes de un canal emisor.

Es un campo de entrada/salida para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

Compresión CurMsg(MQLONG)
Este campo especifica qué técnica se está utilizando actualmente para comprimir los datos del mensaje.

Se establece en uno de los valores siguientes:

MQCOMPRESS_NONE
No se lleva a cabo ninguna compresión de datos de cabecera.

MQCOMPRESS_RLE
Se lleva a cabo la compresión de datos de mensaje utilizando la codificación de longitud de ejecución.

MQCOMPRESS_ZLIBFAST
Se lleva a cabo la compresión de datos de mensaje utilizando el método de compresión zlib. Se prefiere un tiempo de compresión rápido.

MQCOMPRESS_ZLIBHIGH
Se lleva a cabo la compresión de datos de mensaje utilizando el método de compresión zlib. Se prefiere un nivel elevado de compresión.

El valor se puede modificar mediante la salida de mensajes de un canal emisor a uno de los valores soportados negociados a los que se accede desde el campo de lista MsgCompde la MQCD. Esto permite que la técnica utilizada para comprimir los datos del mensaje se decida para cada mensaje basándose en el contenido del mensaje. El valor alterado sólo se utiliza para el mensaje actual. El canal finaliza si el atributo se modifica a un valor no soportado. El valor se ignora si se altera fuera de la salida de mensajes de un canal emisor.

Es un campo de entrada/salida para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_6.

Hconn (MQHCONN)
Este campo especifica el descriptor de conexión que utiliza la salida si necesita realizar alguna llamada MQI dentro de la salida.

Este campo no es relevante para las salidas que se ejecutan en canales de conexión de cliente, donde contiene el valor MQHC_UNUSABLE_HCONN (-1).

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_7.

SharingConversations (MQBOOL)
Este campo especifica si la conversación es la única que se puede ejecutar actualmente en esta instancia de canal, o si actualmente se puede ejecutar más de una conversación en esta instancia de canal.

También indica si el programa de salida está sujeto al riesgo de que otro programa de salida que se ejecuta al mismo tiempo altere el MQCD.

Este campo sólo es relevante para los programas de salida que se ejecutan en canales de conexión de cliente o de conexión de servidor.

Se establece en uno de los valores siguientes:

FALSE
La instancia de salida es la única instancia de salida que se puede ejecutar actualmente en esta instancia de canal. Esto permite a la salida actualizar de forma segura los campos MQCD sin contención de otras salidas que se ejecutan en otras instancias de canal. Si el canal actúa sobre los cambios en los campos MQCD se define mediante la tabla de campos MQCD en "Cambio de campos MQCD en una salida de canal" en la página 1072.

TRUE
La instancia de salida no es la única instancia de salida que se puede ejecutar actualmente en esta instancia de canal. El canal no actúa sobre los cambios realizados en MQCD, excepto los cambios listados en la tabla de campos MQCD en "Cambio de campos MQCD en una salida de canal" en la página 1072 por razones de salida que no sean MQXR_INIT. Si esta salida actualiza los campos MQCD, asegúrese de que no haya ninguna contención de otras salidas en ejecución en otras conversaciones al mismo tiempo proporcionando serialización entre las salidas que se ejecutan en esta instancia de canal.

Es un campo de entrada para la salida. El campo no está presente si *Version* es menor que MQCXP_VERSION_7.

MCAUserSource (MQLONG)
Este campo especifica el origen del ID de usuario de MCA proporcionado.

Puede contener uno de los valores siguientes:

MQUSRC_MAP
El ID de usuario se especifica en el atributo MCAUSER.

MQUSRC_CHANNEL
El ID de usuario fluye desde el socio de entrada o se especifica en el campo MCAUSER definido en el objeto de canal.

Es un campo de entrada para la salida. El campo no está presente si la versión es menor que MQCXP_VERSION_8.

Puntos pEntry(PMQIEP)
Este campo especifica la dirección del punto de entrada de interfaz para la llamada MQI o DCI.

El campo no está presente si *Versión* es menor que MQCXP_VERSION_8.

Declaración C
Esta declaración es la declaración C para la estructura 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 SSL certificate */
  MQLONG    SSLRemCertIssNameLength;   /* Length of
                                   distinguished name of issuer
                                   of remote SSL certificate */
  MQPTR     SSLRemCertIssNamePtr;      /* Address of
                                   distinguished name of issuer
                                   of remote SSL 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 */
;
```

declaración COBOL

Esta declaración es la declaración COBOL para la estructura 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 SSL
** certificate
   15  MQCXP-SSLREMCERTISSNAMELENGTH PIC S9(9) BINARY.
** Address of distinguished name of issuer of remote SSL
** 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.
```

Declaración RPG (ILE)

Esta declaración es la declaración RPG para la estructura 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 SSL certificate
     D  CXSSLCU              161    172
     D* Length of distinguished name of issuer of remote SSL certificate
     D  CXSRCINL             173    176I 0
```

```
     D* Address of distinguished name of issuer of remote SSL 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
```

System/390

Esta declaración es la declaración de ensamblador System/390 para la estructura 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 SSL certificate
MQCXP_SSLREMCERTISSNAMELENGTH   DS   F     Length of distinguished name
*                                  of issuer of remote SSL certificate
MQCXP_SSLREMCERTISSNAMEPTR      DS   F     Address of distinguished name
*                                  of issuer of remote SSL 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_LENGTH            EQU  *-MQCXP
                        ORG  MQCXP
MQCXP_AREA              DS   CL(MQCXP_LENGTH)
```

MQXWD-Descriptor de espera de salida

La estructura MQXWD es un parámetro de entrada/salida en la llamada MQXWAIT.

Esta estructura solo está soportada en z/OS.

Referencia relacionada

“Campos” en la página 1090
Este tema lista todos los campos de la estructura MQXWD y describe cada campo.

“Declaración C” en la página 1090

Esta declaración es la declaración C para la estructura MQXWD.

" System/390" en la página 1091
Esta declaración es la declaración de ensamblador System/390 para la estructura MQXWD.

Campos

Este tema lista todos los campos de la estructura MQXWD y describe cada campo.

StrucId (MQCHAR4)
Este campo especifica el identificador de estructura.

El valor debe ser:

MQXWD_ID_STRUCD
Identificador de la estructura del descriptor de espera de salida.

Para el lenguaje de programación C, también se define la constante MQXWD_STRUC_ID_ARRAY; esta constante tiene el mismo valor que MQXWD_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

El valor inicial de este campo es MQXWD_STRUC_ID.

Versión (MQLONG)
Este campo especifica el número de versión de la estructura.

El valor debe ser:

MQXWD_VERSION_1
Número de versión para la estructura del descriptor de espera de salida.

El valor inicial de este campo es MQXWD_VERSION_1.

Reserved1 (MQLONG)
Este campo está reservado. Su valor debe ser cero.

Este es un campo de entrada.

Reserved2 (MQLONG)
Este campo está reservado. Su valor debe ser cero.

Este es un campo de entrada.

Reserved3 (MQLONG)
Este campo está reservado. Su valor debe ser cero.

Este es un campo de entrada.

BCE (MQLONG)
Este campo especifica el bloque de control de sucesos en el que se debe esperar.

Este campo es el bloque de control de sucesos (ECB) en el que se debe esperar.Debe establecerse en cero antes de que se emita la llamada MQXWAIT; cuando se complete correctamente, contendrá el código de publicación.

Este campo es un campo de entrada/salida.

Declaración C

Esta declaración es la declaración C para la estructura 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 */
};
```

System/390

Esta declaración es la declaración de ensamblador System/390 para la estructura 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)
```

Referencia a la salida de la API

Esta sección proporciona información de referencia principalmente de interés para un programador que escribe salidas de API.

Notas de uso general

notas:

1. Todas las funciones de salida pueden emitir la llamada MQXEP; esta llamada se ha diseñado específicamente para su uso desde las funciones de salida de API.
2. La función MQ_INIT_EXIT no puede emitir ninguna llamada MQ que no sea MQXEP.
3. No puede emitir la llamada MQDISC para la conexión actual.
4. Si una función de salida emite la llamada MQCONN, o la llamada MQCONNX con la opción MQCNO_HANDLE_SHARE_NONE, la llamada se completa con el código de razón MQRC_ALREADY_CONNECTED, y el descriptor de contexto devuelto es el mismo que el que se ha pasado a la salida como parámetro.
5. En general, cuando una función de salida de API emite una llamada MQI, no se llama a las salidas de API de forma recursiva. Sin embargo, si una función de salida emite la llamada MQCONNX con las opciones MQCNO_HANDLE_SHARE_BLOCK o MQCNO_HANDLE_SHARE_NO_BLOCK, la llamada devuelve un nuevo descriptor de contexto compartido. Esto proporciona a la suite de salida un descriptor de conexión propio y, por lo tanto, una unidad de trabajo independiente de la unidad de trabajo de la aplicación. La suite de salida puede utilizar este descriptor de contexto para colocar y obtener mensajes dentro de su propia unidad de trabajo, y confirmar o restituir dicha unidad de trabajo; todo esto se puede realizar sin afectar a la unidad de trabajo de la aplicación de ninguna manera.

   Debido a que la función de salida está utilizando un descriptor de conexión que es diferente del descriptor utilizado por la aplicación, las llamadas de MQ emitidas por la función de salida dan como resultado que se invoquen las funciones de salida de API relevantes. Por lo tanto, las funciones de salida se pueden invocar de forma recursiva. Tenga en cuenta que tanto el campo *ExitUserArea* en MQAXP como el área de cadena de salida tienen ámbito de descriptor de conexión. En consecuencia, una función de salida no puede utilizar esas áreas para indicar a otra instancia de sí misma invocada recursivamente que ya está activa.
6. Las funciones de salida también pueden colocar y obtener mensajes dentro de la unidad de trabajo de la aplicación. Cuando la aplicación confirma o restituye la unidad de trabajo, todos los mensajes de la unidad de trabajo se confirman o se restituyen juntos, independientemente de quién los haya colocado en la unidad de trabajo (función de aplicación o salida). Sin embargo, la salida puede hacer que la aplicación supere los límites del sistema antes de lo que sería el caso (por ejemplo, superando el número máximo de mensajes no confirmados en una unidad de trabajo).

Cuando una función de salida utiliza la unidad de trabajo de la aplicación de esta forma, la función de salida normalmente debe evitar emitir la llamada MQCMIT, ya que esto confirma la unidad de trabajo de la aplicación y puede afectar al funcionamiento correcto de la aplicación. Sin embargo, es posible que a veces la función de salida tenga que emitir la llamada MQBACK, si la función de salida encuentra un error grave que impide que se confirme la unidad de trabajo (por ejemplo, un error al colocar un mensaje como parte de la unidad de trabajo de la aplicación). Cuando se llame a MQBACK, asegúrese de que no se cambien los límites de la unidad de trabajo de la aplicación. En esta situación, la función de salida debe establecer los valores adecuados para asegurarse de que el código de terminación MQCC_WARNING y el código de razón MQRC_BACKED_OUT se devuelven a la aplicación, para que la aplicación pueda detectar el hecho de que la unidad de trabajo se ha restituido.

Si una función de salida utiliza el descriptor de conexión de la aplicación para emitir llamadas MQ , estas llamadas no dan como resultado más invocaciones de las funciones de salida de API.

7. Si una función de salida MQXR_BEFORE termina de forma anómala, es posible que el gestor de colas pueda recuperarse de la anomalía. Si es posible, el gestor de colas continúa el proceso como si la función de salida hubiera devuelto MQXCC_FAILED. Si el gestor de colas no puede recuperarse, la aplicación finaliza.
8. Si una función de salida MQXR_AFTER termina de forma anómala, es posible que el gestor de colas pueda recuperarse de la anomalía. Si es posible, el gestor de colas continúa el proceso como si la función de salida hubiera devuelto MQXCC_FAILED. Si el gestor de colas no puede recuperarse, la aplicación finaliza. Tenga en cuenta que en el último caso, los mensajes recuperados fuera de una unidad de trabajo se pierden (esta es la misma situación que la aplicación que falla inmediatamente después de eliminar un mensaje de la cola).
9. El proceso MCA realiza una confirmación de dos fases.

   Si una salida de API intercepta un MQCMIT de un proceso MCA preparado e intenta realizar una acción dentro de la unidad de trabajo, la acción fallará con el código de razón MQRC_UOW_NOT_AVAILABLE.
10. Para un entorno de varias instalaciones, la única forma de tener una salida que funcione tanto con Websphere MQ versión 7.0 como con la versión 7.1 es escribir la salida de una forma que enlace la versión 7.0 con mqm.Lib y, para salidas no primarias o reubicadas, para asegurarse de que la aplicación encuentra la mqm.Lib correcta para la instalación con la que está asociado actualmente el gestor de colas, antes del inicio de la aplicación. (Por ejemplo, ejecute el mandato **setmqenv -m QM** antes de iniciar la aplicación, incluso si el gestor de colas es propiedad de una instalación de la versión 7.0 .)
11. Cuando haya varias instalaciones de IBM WebSphere MQ disponibles, utilice las salidas escritas para una versión anterior de IBM WebSphere MQ, ya que es posible que la nueva funcionalidad añadida en la versión posterior no funcione con las versiones anteriores. Para obtener más información sobre los cambios entre releases, consulte Cambios en WebSphere MQ 7.5.

Estructura de parámetros de salida de API de IBM WebSphere MQ (MQAXP)

La estructura MQAXP, un bloque de control externo, se utiliza como parámetro de entrada o salida para la salida de API. Este tema también proporciona información sobre cómo los gestores de colas procesan las funciones de salida.

MQAXP tiene la siguiente declaración C:

```
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 */
};
```

La siguiente lista de parámetros se pasa cuando se invocan las funciones de una salida de API:

StrucId (MQCHAR4)-entrada
El identificador de estructura de parámetro de salida, con un valor de:

```
MQAXP_STRUC_ID.
```

El manejador de salida establece este campo en la entrada para cada función de salida.

Versión (MQLONG)-entrada
El número de versión de la estructura, con un valor de:

MQAXP_VERSION_1
Estructura de parámetros de salida de API de la versión 1.

MQAXP_VERSION_2
Estructura de parámetros de salida de API de la versión 2.

VERSIÓN_ACTUAL_MQAXP
Número de versión actual para la estructura de parámetros de salida de API.

El manejador de salida establece este campo en la entrada para cada función de salida.

ExitId (MQLONG)-entrada
El identificador de salida, establecido en la entrada de la rutina de salida, que indica el tipo de salida:

MQXT_API_EXIT
Salida de API.

ExitReason (MQLONG)-entrada
La razón por la que se invoca la salida, establecida en la entrada de cada función de salida:

MQXR_CONNECTION
La salida se invoca para inicializarse a sí misma antes de una llamada MQCONN o MQCONNX, o para finalizar a sí misma después de una llamada MQDISC.

MQXR_ANTES
La salida se invoca antes de ejecutar una llamada de API, o antes de convertir datos en un MQGET.

MQXR_AFTER
La salida se está invocando después de ejecutar una llamada de API.

ExitResponse (MQLONG)-salida
La respuesta de la salida, inicializada al entrar en cada función de salida para:

MQXCC_Correcto
Continúe con normalidad.

La función de salida debe establecer este campo para comunicar al gestor de colas el resultado de la ejecución de la función de salida. El valor debe ser uno de los siguientes:

MQXCC_Correcto
La función de salida se ha completado satisfactoriamente. Continúe con normalidad.

Este valor lo pueden establecer todas las funciones de salida MQXR_ *. ExitResponse2 se utiliza para decidir si se deben invocar las funciones de salida más adelante en la cadena.

MQXCC_FAILED
La función de salida ha fallado debido a un error.

Este valor lo pueden establecer todas las funciones de salida MQXR_ *. El gestor de colas establece CompCode en MQCC_FAILED y Reason en:

- MQRC_API_EXIT_INIT_ERROR si la función es MQ_INIT_EXIT
- MQRC_API_EXIT_TERM_ERROR si la función es MQ_TERM_EXIT
- MQRC_API_EXIT_ERROR para todas las demás funciones de salida

Los valores establecidos pueden ser alterados por una función de salida más adelante en la cadena.

ExitResponse2 se ignora; el gestor de colas continúa el proceso como si se hubiera devuelto MQXR2_SUPPRESS_CHAIN .

MQXCC_SUPPRESS_FUNCTION
Suprima la función de la API WebSphere MQ .

Este valor sólo lo puede establecer una función de salida MQXR_BEFORE. Omite la llamada de API. Si lo devuelve MQ_DATA_CONV_ON_GET_EXIT, se omite la conversión de datos. El gestor de colas establece CompCode en MQCC_FAILED, y Reason en MQRC_SUPPRESSED_BY_EXIT, pero los valores establecidos se pueden modificar mediante una función de salida más adelante en la cadena. Otros parámetros para la llamada permanecen cuando la salida los deja. ExitResponse2 se utiliza para decidir si se deben invocar las funciones de salida más adelante en la cadena.

Si este valor lo establece una función de salida MQXR_AFTER o MQXR_CONNECTION, el gestor de colas continúa el proceso como si se hubiera devuelto MQXCC_FAILED.

MQXCC_SKIP_FUNCTION
Omita la función de API WebSphere MQ .

Este valor sólo lo puede establecer una función de salida MQXR_BEFORE. Omite la llamada de API. Si lo devuelve MQ_DATA_CONV_ON_GET_EXIT, se omite la conversión de datos. La función de salida debe establecer CompCode y Reason en los valores que se van a devolver a la aplicación, pero los valores establecidos pueden ser alterados por una función de salida más adelante en la cadena. Otros parámetros para la llamada permanecen cuando la salida los deja. ExitResponse2 se utiliza para decidir si se deben invocar las funciones de salida más adelante en la cadena.

Si este valor lo establece una función de salida MQXR_AFTER o MQXR_CONNECTION, el gestor de colas continúa el proceso como si se hubiera devuelto MQXCC_FAILED.

MQXCC_SUPPRESS_EXIT
Suprimir todas las funciones de salida que pertenecen al conjunto de salidas.

Este valor sólo lo pueden establecer las funciones de salida MQXR_BEFORE y MQXR_AFTER. Omite *todas* las invocaciones posteriores de las funciones de salida que pertenecen a este conjunto de salidas para esta conexión lógica. Esta omisión continúa hasta que se produce la solicitud de desconexión lógica, cuando se invoca la función MQ_TERM_EXIT con una ExitReason de MQXR_CONNECTION.

La función de salida debe establecer CompCode y Reason en los valores que se van a devolver a la aplicación, pero los valores establecidos pueden ser alterados por una función de salida más adelante en la cadena. Otros parámetros para la llamada permanecen cuando la salida los deja. ExitResponse2 se ignora.

Si este valor lo establece una función de salida MQXR_CONNECTION, el gestor de colas continúa el proceso como si se hubiera devuelto MQXCC_FAILED.

Para obtener información sobre la interacción entre ExitResponse y ExitResponse2, y su efecto en el proceso de salida, consulte “Cómo procesan los gestores de colas las funciones de salida” en la página 1097.

ExitResponse2 (MQLONG)-salida
Este es un código de respuesta de salida secundario que califica el código de respuesta de salida primario para las funciones de salida MQXR_BEFORE. Se inicializa para:

```
MQXR2_DEFAULT_CONTINUATION
```

en la entrada a una función de salida de llamada de API de WebSphere MQ . A continuación, se puede establecer en uno de los valores:

MQXR2_DEFAULT_CONTINUATION
Indica si se debe continuar con la siguiente salida de la cadena, en función del valor de ExitResponse.

Si ExitResponse es MQXCC_SUPPRESS_FUNCTION o MQXCC_SKIP_FUNCTION, omita las funciones de salida más adelante en la cadena MQXR_BEFORE y las funciones de salida coincidentes en la cadena MQXR_AFTER. Invoque las funciones de salida de la cadena MQXR_AFTER que coincidan con las funciones de salida anteriores de la cadena MQXR_BEFORE.

De lo contrario, invoque la siguiente salida de la cadena.

MQXR2_SUPPRESS_CHAIN
Suprima la cadena.

Omita las funciones de salida más adelante en la cadena MQXR_BEFORE y las funciones de salida coincidentes en la cadena MQXR_AFTER para esta invocación de llamada de API. Invoque las funciones de salida de la cadena MQXR_AFTER que coincidan con las funciones de salida anteriores de la cadena MQXR_BEFORE.

MQXR2_CONTINUE_CHAIN
Continúe con la siguiente salida de la cadena.

Para obtener información sobre la interacción entre ExitResponse y ExitResponse2, y su efecto en el proceso de salida, consulte “Cómo procesan los gestores de colas las funciones de salida” en la página 1097.

Comentarios (MQLONG)-entrada/salida
Comunicar códigos de retroalimentación entre invocaciones de función de salida. Se inicializa para:

```
MQFB_NONE (0)
```

antes de invocar la primera función de la primera salida de una cadena.

Las salidas pueden establecer este campo en cualquier valor, incluido cualquier valor MQFB_ * o MQRC_ * válido. Las salidas también pueden establecer este campo en un valor de comentarios definido por el usuario en el rango MQFB_APPL_FIRST a MQFB_APPL_LAST.

APICallerType (MQLONG)-entrada
El tipo de interlocutor de la API, que indica si el interlocutor de la API de WebSphere MQ es externo o interno del gestor de colas: MQXACT_EXTERNAL o MQXACT_INTERNAL.

ExitUserArea (MQBYTE16)-entrada/salida
Un área de usuario, disponible para todas las salidas asociadas con un objeto ExitInfodeterminado. Se inicializa en MQXUA_NONE (ceros binarios para la longitud del área ExitUser) antes de invocar la primera función de salida (MQ_INIT_EXIT) para el hconn. A partir de entonces, los cambios realizados en este campo por una función de salida se conservan entre invocaciones de funciones de la misma salida.

Este campo está alineado con un múltiplo de 4 MQLONG.

Las salidas también pueden anclar cualquier almacenamiento que asignen desde esta área.

Para cada hconn, cada salida de una cadena de salidas tiene un área ExitUserdiferente. Las salidas de una cadena no pueden compartir el área ExitUsery el contenido del área ExitUserde una salida no está disponible para otra salida de una cadena.

Para programas C, la constante MQXUA_NONE_ARRAY también se define con el mismo valor que MQXUA_NONE, pero como una matriz de caracteres en lugar de una serie.

La longitud de este campo la proporciona MQ_EXIT_USER_AREA_LENGTH.

ExitData (MQCHAR32)-entrada
Datos de salida, establecidos en la entrada de cada función de salida en los 32 caracteres de datos específicos de salida que se proporcionan en la salida. Si no define ningún valor en la salida, este campo estará en blanco.

La longitud de este campo la proporciona MQ_EXIT_DATA_LENGTH.

ExitInfoNombre (MQCHAR48)-entrada
El nombre de información de salida, establecido en la entrada de cada función de salida en el ApiExit_name especificado en las definiciones de salida en las stanzas.

ExitPDArea (MQBYTE48)-entrada/salida
Un área de determinación de problemas, inicializada en MQXPDA_NONE (ceros binarios para la longitud del campo) para cada invocación de una función de salida.

Para programas C, la constante MQXPDA_NONE_ARRAY también se define con el mismo valor que MQXPDA_NONE, pero como una matriz de caracteres en lugar de una serie.

El manejador de salida siempre escribe esta área en el rastreo de WebSphere MQ al final de una salida, incluso cuando la función es satisfactoria.

La longitud de este campo la proporciona MQ_EXIT_PD_AREA_LENGTH.

QMgrName (MQCHAR48)-entrada
El nombre del gestor de colas al que está conectada la aplicación, que ha invocado una salida como resultado del proceso de una llamada de API WebSphere MQ .

Si el nombre de un gestor de colas proporcionado en una llamada MQCONN o MQCONNX está en blanco, este campo sigue establecido en el nombre del gestor de colas al que está conectada la aplicación, tanto si la aplicación es servidor como si es cliente.

El manejador de salida establece este campo en la entrada para cada función de salida.

La longitud de este campo la proporciona MQ_Q_MGR_NAME_LENGTH.

ExitChainAreaPtr (PMQACH)-entrada/salida
Se utiliza para comunicar datos entre invocaciones de distintas salidas de una cadena. Se establece en un puntero NULL antes de invocar la primera función (MQ_INIT_EXIT con ExitReason MQXR_CONNECTION) de la primera salida de una cadena de salidas. El valor devuelto por la salida en una invocación se pasa a la siguiente invocación.

Consulte “El área de cadena de salida y la cabecera de área de cadena de salida (MQACH)” en la página 1100 para obtener más detalles sobre cómo utilizar el área de cadena de salida.

Hconfig (MQHCONFIG)-entrada
El descriptor de contexto de configuración, que representa el conjunto de funciones que se están inicializando. Este valor lo genera el gestor de colas en la función MQ_INIT_EXIT y se pasa posteriormente a la función de salida de API. Se establece en la entrada para cada función de salida.

Puede utilizar Hconfig como puntero a la estructura MQIEP para realizar llamadas MQI y DCI. Debe comprobar que los 4 primeros bytes de HConfig coinciden con el StrucId de la estructura MQIEP antes de utilizar el parámetro HConfig como puntero a la estructura MQIEP.

Función (MQLONG)-entrada
El identificador de función, cuyos valores válidos son las constantes MQXF_ * descritas en “Constantes externas” en la página 1102.

El manejador de salida establece este campo en el valor correcto, al entrar en cada función de salida, en función de la llamada de API WebSphere MQ que ha dado como resultado la invocación de la salida.

ExitMsgDescriptor de contexto (MQHMSG)-entrada/salida
Cuando la función es MQXF_GET y ExitReason es MQXR_AFTER, se devuelve un descriptor de mensaje válido en este campo que permite a la salida de API acceder a los campos del descriptor de mensaje y a cualquier otra propiedad que coincida con la serie ExitProperties especificada en la estructura MQXEPO al registrar la salida de API.

Cualquier propiedad de descriptor que no sea de mensaje que se devuelva en el descriptor de contexto ExitMsgno estará disponible en el MsgHandle en la estructura MQGMO si se ha especificado una, o en los datos del mensaje.

Cuando la función es MQXF_GET y ExitReason es MQXR_BEFORE, si el programa de salida establece este campo en MQHM_NONE, suprimirá el llenado de las propiedades de manejador ExitMsg.

Este campo no se establece si la versión es menor que MQAXP_VERSION_2.

Cómo procesan los gestores de colas las funciones de salida

El proceso realizado por el gestor de colas al devolver una función de salida depende de ExitResponse y de ExitResponse2.

La Tabla 593 en la página 1097 resume las combinaciones posibles y sus efectos para una función de salida MQXR_BEFORE, mostrando:

- Quién establece los parámetros CompCode y Reason de la llamada de API
- Si se invocan las funciones de salida restantes de la cadena MQXR_BEFORE y las funciones de salida coincidentes de la cadena MQXR_AFTER
- Si se invoca la llamada de API

Para una función de salida MQXR_AFTER:

- CompCode y Reason se establecen de la misma forma que MQXR_BEFORE
- ExitResponse2 se ignora (las funciones de salida restantes de la cadena MQXR_AFTER siempre se invocan)
- MQXCC_SUPPRESS_FUNCTION y MQXCC_SKIP_FUNCTION no son válidos

Para una función de salida MQXR_CONNECTION:

- CompCode y Reason se establecen de la misma forma que MQXR_BEFORE
- ExitResponse2 se ignora
- MQXCC_SUPPRESS_FUNCTION, MQXCC_SKIP_FUNCTION, MQXCC_SUPPRESS_EXIT no son válidos

En todos los casos, en los que una salida o el gestor de colas establece CompCode y Reason, los valores establecidos se pueden cambiar mediante una salida invocada más adelante, o mediante la llamada de API (si la llamada de API se invoca más tarde).

Tabla 593. Proceso de salida MQXR_BEFORE

Valor de ExitResponse	**CompCode y Reason establecidos por**	**Valor de ExitResponse2 (continuación predeterminada) Cadena**	**Valor de la API ExitResponse2 (continuación predeterminada)**
MQXCC_Correcto	salida	Y	Y
MQXCC_SUPPRESS_EXIT	salida	Y	Y
MQXCC_SUPPRESS_FUNCTION	gestor de colas	N	N
MQXCC_SKIP FUNCIÓN	salida	N	N
MQXCC_FAILED	gestor de colas	N	N

Cómo procesan los clientes las funciones de salida

En general, los clientes procesan las funciones de salida de la misma forma que las aplicaciones de servidor, y el atributo *QMgrName* de esta estructura se aplica tanto si la función está en un servidor como si está en un cliente.

Sin embargo, el cliente no tiene ningún concepto del archivo *mqs.ini* , por lo que las stanzas *ApiExitCommon* y *APIExitTemplate* no se aplican. Solo se aplica la stanza *ApiExitLocal* y esta stanza se configura en el archivo *mqclient.ini* .

Estructura de contexto de salida de API de IBM WebSphere MQ (MQAXC)

La estructura MQAXC, un bloque de control externo, se utiliza como parámetro de entrada para una salida de API.

MQAXC tiene la siguiente declaración C:

```
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 */
 };
```

Los parámetros para MQAXC son:

StrucId (MQCHAR4)-entrada
El identificador de estructura de contexto de salida, con un valor de MQAXC_STRUC_ID. Para los programas C, también se define la constante MQAXC_STRUC_ID_ARRAY, con el mismo valor que MQAXC_STRUC_ID, pero como una matriz de caracteres en lugar de una serie.

El manejador de salida establece este campo en la entrada para cada función de salida.

Versión (MQLONG)-entrada
El número de versión de la estructura, con un valor de:

MQAXC_VERSION_2
Número de versión para la estructura de contexto de salida.

MQAXC_VERSIÓN_ACTUAL
Número de versión actual para la estructura de contexto de salida.

El manejador de salida establece este campo en la entrada para cada función de salida.

Entorno (MQLONG)-entrada
El entorno desde el que se ha emitido una llamada de API de WebSphere MQ que ha dado como resultado la ejecución de una función de salida. Los valores válidos para este campo son:

MQXE_OTHER
Este valor es coherente con las invocaciones que una salida de API ve si se llama a la salida desde una aplicación de servidor. Esto significa que una salida de API se ejecuta sin cambios en un cliente y no ve nada diferente.

Si la salida realmente necesita determinar si se está ejecutando en el cliente, puede hacerlo consultando los campos *ChannelName* y *ChannelDefinition* .

MQXE_MCA
Agente de canal de mensajes

MQXE_MCA_SVRCONN
Un agente de canal de mensajes que actúa en nombre de un cliente

MQXE_COMMAND_SERVER
El servidor de mandatos

MQXE_MQSC
El intérprete de mandatos runmqsc

El manejador de salida establece este campo en la entrada para cada función de salida.

UserId (MQCHAR12)-entrada
El ID de usuario asociado a la aplicación. En particular, en el caso de las conexiones de cliente, este campo contiene el ID de usuario del usuario adoptado en contraposición al ID de usuario bajo el que se ejecuta el código de canal. Si un ID de usuario en blanco fluye desde el cliente, no se realiza ningún cambio en el ID de usuario que ya se está utilizando. Es decir, no se adopta ningún ID de usuario nuevo.

El manejador de salida establece este campo en la entrada para cada función de salida. La longitud de este campo la proporciona MQ_USER_ID_LENGTH.

En el caso de un cliente, es el ID de usuario enviado desde el cliente al servidor. Tenga en cuenta que es posible que no sea el ID de usuario efectivo con el que se ejecuta el cliente en el gestor de colas, ya que podría haber una configuración MCAUser o CHLAUTH que cambie el ID de usuario.

SecurityId (MQBYTE40)-entrada
Una extensión del ID de usuario que ejecuta la aplicación. Su longitud la proporciona MQ_SECURITY_ID_LENGTH.

En el caso de un cliente, es el ID de usuario enviado desde el cliente al servidor. Tenga en cuenta que es posible que no sea el ID de usuario efectivo con el que se ejecuta el cliente en el gestor de colas, ya que podría haber una configuración MCAUser o CHLAUTH que cambie el ID de usuario.

ConnectionName (MQCHAR264)-entrada
El campo de nombre de conexión, establecido en la dirección del cliente. Por ejemplo, para TCP/IP, sería la dirección IP del cliente.

La longitud de este campo la proporciona MQ_CONN_NAME_LENGTH.

En el caso de un cliente, es la dirección asociada del gestor de colas.

LongMCAUserIdLength (MQLONG)-entrada
Longitud del identificador de usuario de MCA largo.

Cuando MCA se conecta al gestor de colas, este campo se establece en la longitud del identificador de usuario de MCA largo (o cero si no existe dicho identificador).

En el caso de un cliente, es el identificador de usuario largo del cliente.

LongRemoteUserIdLength (MQLONG)-entrada
La longitud del identificador de usuario remoto largo.

Cuando MCA se conecta al gestor de colas, este campo se establece en la longitud del identificador de usuario remoto largo. De lo contrario, este campo se establecerá en cero

En el caso de un cliente, establezca este campo en cero.

LongMCAUserIdPtr (MQPTR)-entrada
Dirección del identificador de usuario de MCA largo.

Cuando MCA se conecta al gestor de colas, este campo se establece en la dirección del identificador de usuario de MCA largo (o en un puntero nulo si no existe dicho identificador).

En el caso de un cliente, es el identificador de usuario largo del cliente.

LongRemoteUserIdPtr (MQPTR)-entrada
La dirección del identificador de usuario remoto largo.

Cuando MCA se conecta al gestor de colas, este campo se establece en la dirección del identificador de usuario remoto largo (o en un puntero nulo si no existe dicho identificador).

En el caso de un cliente, establezca este campo en cero.

ApplName (MQCHAR28)-entrada
El nombre de la aplicación o componente que ha emitido la llamada de API WebSphere MQ .

Las reglas para generar el ApplName son las mismas que para generar el nombre predeterminado para un MQPUT.

El valor de este campo se encuentra consultando el nombre del programa en el sistema operativo. Su longitud la proporciona MQ_APPL_NAME_LENGTH.

ApplType (MQLONG)-entrada
El tipo de aplicación o componente que ha emitido la llamada de API WebSphere MQ .

El valor es MQAT_DEFAULT para la plataforma en la que se compila la aplicación, o equivale a uno de los valores MQAT_ * definidos.

El manejador de salida establece este campo en la entrada para cada función de salida.

ProcessId (MQPID)-entrada
Identificador de proceso del sistema operativo.

Cuando sea aplicable, el manejador de salida establece este campo en la entrada para cada función de salida.

ThreadId (MQTID)-entrada
El identificador de hebra de MQ . Es el mismo identificador que se utiliza en el rastreo de MQ y en los volcados FFST , pero puede ser diferente del identificador de hebra del sistema operativo.

Cuando sea aplicable, el manejador de salida establece este campo en la entrada para cada función de salida.

ChannelName (MQCHAR)-entrada
El nombre del canal, rellenado con espacios en blanco, si es aplicable y conocido.

Si no es aplicable, este campo se establece en caracteres NULL.

Reserved1 (MQBYTE4)-entrada
Este campo está reservado.

ChanneDefinition (PMQCD)-entrada
Puntero a la definición de canal que se utiliza, si es aplicable y conocida.

Si no es aplicable, este campo se establece en caracteres NULL.

Tenga en cuenta que el puntero sólo se completa si la conexión se está procesando en nombre de un canal de WebSphere MQ y esa definición de canal se ha leído.

En concreto, la definición de canal no se proporciona en el servidor cuando se realiza la primera llamada MQCONN para el canal. Además, si el puntero se llena, la estructura (y cualquier subestructura) apuntada por el puntero debe ser tratada como de sólo lectura; cualquier actualización de la estructura conduciría a resultados impredecibles y no está soportada.

En el caso de un cliente, los campos distintos de aquellos con un valor especificado para un cliente, contienen valores que son adecuados para una aplicación cliente.

El área de cadena de salida y la cabecera de área de cadena de salida (MQACH)

Si es necesario, una función de salida puede adquirir almacenamiento para un área de cadena de salida y establecer la ExitChainAreaPtr en MQAXP para que apunte a este almacenamiento.

Las salidas (las mismas o diferentes funciones de salida) pueden adquirir varias áreas de cadena de salida y enlazarlas. Las áreas de cadena de salida sólo deben añadirse o eliminarse de esta lista mientras se llama desde el manejador de salida. Esto garantiza que no haya problemas de serialización causados por distintas hebras que añadan o eliminen áreas de la lista al mismo tiempo.

Un área de cadena de salida debe empezar con una estructura de cabecera MQACH, cuya declaración C es:

```
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 */
 };
```

Los campos de la cabecera de área de cadena de salida son:

StrucId (MQCHAR4)-entrada
El identificador de estructura de área de cadena de salida, con un valor inicial, definido por MQACH_DEFAULT, de MQACH_STRUC_ID.

Para programas C, la constante MQACH_STRUC_ID_ARRAY también está definida; tiene el mismo valor que MQACH_STRUC_ID, pero como una matriz de caracteres en lugar de una serie.

Versión (MQLONG)-entrada
El número de versión de la estructura, tal como se indica a continuación:

MQACH_VERSION_1
El número de versión para la estructura del parámetro de salida.

VERSIÓN_ACTUAL_MQACH_VERSIÓN
El número de versión actual para la estructura de contexto de salida.

El valor inicial de este campo, definido por MQACH_DEFAULT, es MQACH_CURRENT_VERSION.

Nota: Si introduce una nueva versión de esta estructura, el diseño de la parte existente no cambia. Las funciones de salida deben comprobar que el número de versión es igual o mayor que la versión más baja que contiene los campos que la función de salida necesita utilizar.

StrucLength (MQLONG)-entrada
Longitud de la estructura MQACH. Las salidas pueden utilizar este campo para determinar el inicio de los datos de salida, estableciéndolos en la longitud de la estructura creada por la salida.

El valor inicial de este campo, definido por MQACH_DEFAULT, es MQACH_CURRENT_LENGTH.

ChainAreaLength (MQLONG)-entrada
La longitud del área de cadena de salida, establecida en la longitud global del área de cadena de salida actual, incluida la cabecera MQACH.

El valor inicial de este campo, definido por MQACH_DEFAULT, es cero.

ExitInfoNombre (MQCHAR48)-entrada
El nombre de información de salida.

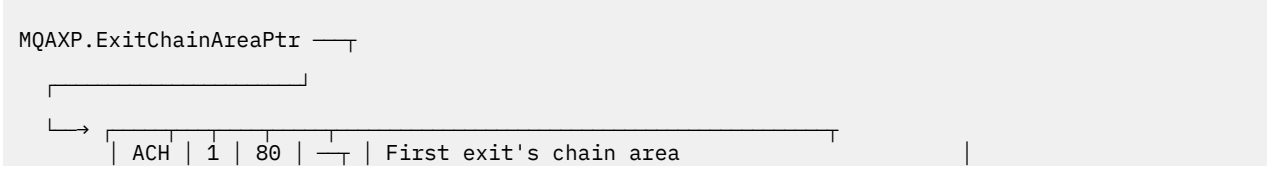
Cuando una salida crea una estructura MQACH, debe inicializar este campo con su propio nombre ExitInfo, para que posteriormente esta estructura MQACH pueda ser encontrada por otra instancia de esta salida o por una salida cooperante.

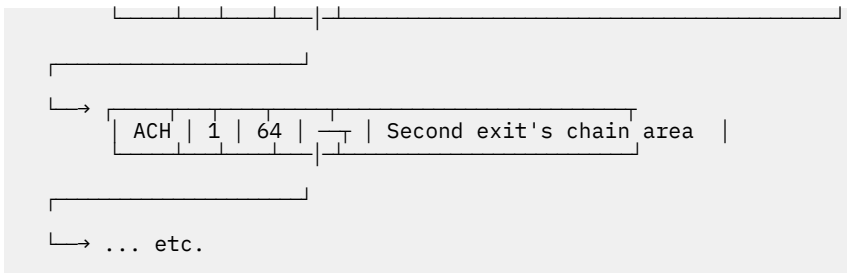
El valor inicial de este campo, definido por MQACH_DEFAULT, es una serie de longitud cero ({""}).

NextChainAreaPtr (PMQACH)-entrada
Puntero al siguiente área de cadena de salida con un valor inicial, definido por MQACH_DEFAULT, de puntero nulo (NULL).

Las funciones de salida deben liberar el almacenamiento para las áreas de cadena de salida que adquieran y manipular los punteros de cadena para eliminar sus áreas de cadena de salida de la lista.

Un área de cadena de salida se puede construir de la siguiente manera:





Constantes externas

Utilice este tema como información de referencia para las constantes externas disponibles para la API.

Las siguientes constantes externas están disponibles para las salidas de API:

MQXF_ * (identificadores de función de salida)

```
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_ * (razones de salida)

```
MQXR_BEFORE                     1       X'00000001'
MQXR_AFTER                      2       X'00000002'
MQXR_CONNECTION                 3       X'00000003'
```

MQXE_ * (entornos)

```
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* _ * (constantes adicionales)

```
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* _ * (constantes nulas)

```
MQXPDA_NONE                    X'00...00' (48 nulls)
MQXPDA_NONE_ARRAY              '\0','\0',...,'\0','\0'
```

MQXCC_ * (códigos de terminación)

```
MQXCC_FAILED                  -8
```

MQRC_ * (códigos de razón)

MQRC_API_EXIT_ERROR 2374 X'00000946'
Una invocación de función de salida ha devuelto un código de respuesta no válido, o ha fallado de algún modo, y el gestor de colas no puede determinar la siguiente acción a realizar.

Examine los campos ExitResponse y ExitResponse2 de MQAXP para determinar el código de respuesta incorrecto y cambie la salida para que devuelva un código de respuesta válido.

MQRC_API_EXIT_INIT_ERROR 2375 X'00000947'
El gestor de colas ha encontrado un error al inicializar el entorno de ejecución para una función de salida de API.

MQRC_API_EXIT_TERM_ERROR 2376 X'00000948'
El gestor de colas ha encontrado un error al cerrar el entorno de ejecución para una función de salida de API.

MQRC_EXIT_REASON_ERROR 2377 X'00000949'
El valor del campo ExitReason proporcionado en una llamada de registro de punto de entrada de salida (MQXEP) es erróneo.

Examine el valor del campo ExitReason para determinar y corregir el valor de razón de salida incorrecta.

MQRC_RESERVED_VALUE_ERROR 2378 X'0000094A'
El valor del campo Reservado es erróneo.

Examine el valor del campo Reservado para determinar y corregir el valor Reservado.

Definiciones de tipo de idioma C

Este tema proporciona información sobre las typedefs asociadas con salidas de API disponibles en lenguaje C.

A continuación se muestran las definiciones de tipo de lenguaje C asociadas con las salidas de 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;
```

Llamada de registro de punto de entrada de salida (MQXEP)

Utilice esta información para obtener información sobre MQXEP, invocación de lenguaje C MQXEP y prototipo de función C MQXEP.

Utilice la llamada MQXEP para:

1. Registrar los puntos de invocación de salida de API antes y después de WebSphere MQ en los que invocar las funciones de salida
2. Especificar los puntos de entrada de la función de salida
3. Anular el registro de los puntos de entrada de la función de salida

Normalmente codificaría las llamadas MQXEP en la función de salida MQ_INIT_EXIT, pero puede especificarlas en cualquier función de salida posterior.

Si utiliza una llamada MQXEP para registrar una función de salida ya registrada, la segunda llamada MQXEP se completa correctamente, sustituyendo la función de salida registrada.

Si utiliza una llamada MQXEP para registrar una función de salida NULL, la llamada MQXEP se completa correctamente y se anula el registro de la función de salida.

Si se utilizan llamadas MQXEP para registrar, anular el registro y volver a registrar una función de salida determinada durante la vida de una solicitud de conexión, se reactiva la función de salida registrada anteriormente. Cualquier almacenamiento todavía asignado y asociado con esta instancia de función de salida está disponible para que lo utilicen las funciones de la salida. (Este almacenamiento normalmente se libera durante la invocación de la función de salida de terminación.)

La interfaz con MQXEP es:

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason)
```

donde:

Hconfig (MQHCONFIG)-entrada
El descriptor de contexto de configuración, que representa la salida de API que incluye el conjunto de funciones que se están inicializando. Este valor lo genera el gestor de colas inmediatamente antes de invocar la función MQ_INIT_EXIT y se pasa en MQAXP a cada función de salida de API.

ExitReason (MQLONG)-entrada
Razón por la que se está registrando el punto de entrada, por las razones siguientes:

- Inicialización o terminación de nivel de conexión (MQXR_CONNECTION)
- Antes de una llamada de API de WebSphere MQ (MQXR_BEFORE)
- Después de una llamada de API de WebSphere MQ (MQXR_AFTER)

Función (MQLONG)-entrada
El identificador de función, cuyos valores válidos son las constantes MQXF_ * (consulte “Constantes externas” en la página 1102).

EntryPoint (PMQFUNC)-entrada
La dirección del punto de entrada para la función de salida que se va a registrar. El valor NULL indica que no se ha proporcionado la función de salida o que se está anulando el registro de un registro anterior de la función de salida.

ExitOpts(MQXEPO)
Las salidas de API pueden especificar opciones que controlan cómo se registran las salidas de API. Si se especifica un puntero nulo para este campo, se asumen los valores predeterminados de la estructura MQXEPO.

CompCode (MQLONG)-salida
El código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón (MQLONG)-salida
El código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED:

MQRC_HCONFIG_ERROR
(2280, X'8E8') El descriptor de contexto de configuración proporcionado no es válido. Utilice el descriptor de contexto de configuración de MQAXP.

MQRC_EXIT_REASON_ERROR
(2377, X' 949 ') La razón de invocación de función de salida proporcionada no es válida o no es válida para el identificador de función de salida proporcionado.

Utilice una de las razones de invocación de función de salida válidas (valor MQXR_ *) o utilice una combinación válida de identificador de función y razón de salida. (Consulte Tabla 594 en la página 1105.)

MQRC_FUNCTION_ERROR
(2281, X'8E9') El identificador de función proporcionado no es válido por razón de salida de API. La tabla siguiente muestra combinaciones válidas de identificadores de función y ExitReasons.

Tabla 594. Combinaciones válidas de identificadores de función y ExitReasons

Función	**ExitReason**
MQXF_INIT MQXF_TERM	MQXR_CONNECTION

Tabla 594. Combinaciones válidas de identificadores de función y ExitReasons (continuación)	
Función	**ExitReason**
MQXF_CONN MQXF_CONNX MQXF_DISC MQXF_OPEN MQXF_CLOSE MQXF_PUT1 MQXF_PUT MQXF_GET MQXF_INQ MQXF_SET MQXF_INICIO MQXF_CMIT MQXF_BACK MQXF_STAT MQXF_CB MQXF_CTL MQXF_CALLBACK MQXF_SUB MQXF_SUBRQ	MQXR_ANTES MQXR_AFTER
MQXF_DATA_CONV_ON_GET	MQXR_ANTES

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Un intento de registrar o anular el registro de una función de salida ha fallado debido a un problema de recurso.

MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Un intento de registrar o anular el registro de una función de salida ha fallado inesperadamente.

MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Nombre de ExitProperties no válido.

MQRC_XEPO_ERROR
(2507, X'09CB') La estructura de opciones de salida no es válida.

Invocación de lenguaje C MQXEP

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason);
```

Declaración para lista de parámetros:

```
    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 de función C MQXEP

```
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 */
```

Funciones de salida

Esta sección proporciona información general para ayudarle a utilizar las llamadas de función y describe cómo invocar las funciones de salida individuales.

Utilice esta información para comprender las reglas generales para las rutinas de salida de API y para configurar y limpiar el entorno de ejecución de salida.

Reglas generales para rutinas de salida de API

Las siguientes reglas generales se aplican al invocar rutinas de salida de API.

- En todos los casos, las funciones de salida de API se controlan antes de validar los parámetros de llamada de API y antes de cualquier comprobación de seguridad (en el caso de MQCONN, MQCONNX o MQOPEN).
- Los valores de los campos introducidos y la salida de una rutina de salida son:
  - En la entrada a una función de salida de API *anterior a* WebSphere MQ , el valor de un campo puede ser establecido por el programa de aplicación o por una invocación de función de salida anterior.
  - En la salida de una función de salida de API *anterior a* WebSphere MQ , el valor de un campo se puede dejar sin modificar o establecer en algún otro valor por parte de la función de salida.
  - En la entrada a una función de salida de API de *después de* WebSphere MQ , el valor de un campo puede ser el valor establecido por el gestor de colas después de procesar la llamada de API de WebSphere MQ , o puede establecerse en un valor mediante una invocación de función de salida anterior en la cadena de funciones de salida.
  - En la salida de una función de salida de llamada de API *después de* WebSphere MQ , el valor de un campo se puede dejar sin modificar o establecer en algún otro valor por parte de la función de salida.
- Las funciones de salida deben comunicarse con el gestor de colas utilizando los campos ExitResponse y ExitResponse2 .
- Los campos CompCode y Código de razón se comunican de nuevo con la aplicación. El gestor de colas y las funciones de salida pueden establecer los campos CompCode y Código de razón.
- La llamada MQXEP devuelve nuevos códigos de razón a las funciones de salida que llaman a MQXEP. Sin embargo, las funciones de salida pueden convertir estos nuevos códigos de razón a cualquier código de razón existente que las aplicaciones existentes y nuevas puedan comprender.
- Cada prototipo de función de salida tiene parámetros similares a la función de la API con un nivel adicional de indirección, excepto CompCode y Reason.
- Las salidas de API pueden emitir llamadas MQI (excepto MQDISC), pero estas llamadas MQI no invocan ellas mismas salidas de API.

Tenga en cuenta que si la aplicación está en un servidor o un cliente, no puede predecir la secuencia de las llamadas de salida de API. Es posible que una llamada BEFORE de salida de API no vaya seguida inmediatamente de una llamada AFTER .

La llamada BEFORE puede ir seguida de otra llamada BEFORE . Por ejemplo:

MQCTL ANTES
ANTES de devolución de llamada
ANTES DE MQPUT
DESPUÉS de MQPUT
DESPUÉS de devolución de llamada
DESPUÉS DE MQCTL

o

ANTES DE XAOPEN
ANTES DE MQCONNX
DESPUÉS de MQCONNX
DESPUÉS DE XAOPEN

En el cliente, hay una salida que puede modificar el comportamiento de la llamada MQCONN o MQCONNX, denominada salida `PreConnect` . La salida `PreConnect` puede modificar cualquiera de los parámetros de la llamada MQCONN o MQCONNX, incluido el nombre del gestor de colas. El cliente llama primero a esta salida y, a continuación, invoca la llamada MQCONN o MQCONNX. Tenga en cuenta que sólo la llamada MQCONN o MQCONNX inicial invoca la salida de API; las llamadas de reconexión posteriores no tienen ningún efecto.

Entorno de ejecución

En general, todos los errores de las funciones de salida se comunican de nuevo al manejador de salida utilizando los campos ExitResponse y ExitResponse2 en MQAXP.

Estos errores a su vez se convierten en valores MQCC_ * y MQRC_ * y se comunican de nuevo a la aplicación en los campos CompCode y Reason. Sin embargo, los errores encontrados en la lógica del manejador de salida se comunican de nuevo a la aplicación como valores MQCC_ * y MQRC_ * en los campos CompCode y Reason.

Si una función MQ_TERM_EXIT devuelve un error:

- La llamada MQDISC ya ha tenido lugar
- No hay otra oportunidad de controlar la función de salida *after* MQ_TERM_EXIT (y, por lo tanto, realizar la limpieza del entorno de ejecución de salida)
- La limpieza del entorno de ejecución de salida *no* se realiza

La salida no se puede descargar porque puede que todavía esté en uso. Además, otras salidas registradas más abajo en la cadena de salida para las que la salida *anterior* ha sido satisfactoria, se activarán en el orden inverso.

Configuración del entorno de ejecución de salida

Al procesar una llamada MQCONN o MQCONNX explícita, la lógica de manejo de salida configura el entorno de ejecución de salida antes de invocar la función de inicialización de salida (MQ_INIT_EXIT). La configuración del entorno de ejecución de salida implica cargar la salida, adquirir almacenamiento para e inicializar las estructuras de parámetros de salida. También se asigna el descriptor de contexto de configuración de salida.

Si se producen errores durante esta fase, la llamada MQCONN o MQCONNX falla con CompCode MQCC_FAILED y uno de los siguientes códigos de razón:

MQRC_API_EXIT_LOAD_ERROR
Ha fallado un intento de cargar un módulo de salida de API.

MQRC_API_EXIT_NOT_FOUND
No se ha podido encontrar una función de salida de API en el módulo de salida de API.

MQRC_STORAGE_NOT_AVAILABLE
Ha fallado un intento de inicializar el entorno de ejecución para una función de salida de API porque no había suficiente almacenamiento disponible.

MQRC_API_EXIT_INIT_ERROR
Se ha encontrado un error al inicializar el entorno de ejecución para una función de salida de API.

Limpieza del entorno de ejecución de salida

Al procesar una llamada MQDISC explícita, o una solicitud de desconexión implícita como resultado de la finalización de una aplicación, es posible que la lógica de manejo de salidas tenga que limpiar el entorno

de ejecución de salida después de invocar la función de terminación de salida (MQ_TERM_EXIT), si está registrada.

La limpieza del entorno de ejecución de salida implica la liberación de almacenamiento para estructuras de parámetros de salida, posiblemente la supresión de los módulos cargados previamente en la memoria.

Si se producen errores durante esta fase, una llamada MQDISC explícita falla con CompCode MQCC_FAILED y el siguiente código de razón (los errores no se resaltan en las solicitudes de desconexión implícitas):

MQRC_API_EXIT_TERM_ERROR
Se ha encontrado un error al cerrar el entorno de ejecución para una función de salida de API. La salida *no* debe devolver ningún error de MQDISC antes o después de las llamadas a la función de salida de la API MQ_TERM*.

Salidas de API en clientes

Un cliente utiliza la salida PreConnect para modificar el comportamiento de las llamadas MQCONN y MQCONNX y no da soporte a las propiedades de salida de API.

Salida PreConnect

En un cliente, la salida PreConnect se puede utilizar para buscar la definición de canal desde un repositorio central, como un servidor LDAP.

La salida PreConnect también puede modificar cualquier parámetro, o todos los parámetros, en una llamada MQCONN o MQCONNX, por ejemplo, el nombre del gestor de colas.

En el caso de las aplicaciones cliente, se debe llamar a la salida PreConnect antes de la salida de API porque la salida de API MQCONN o MQCONNX sólo se llama una vez que se conoce el nombre del gestor de colas y la salida PreConnect puede cambiar este nombre.

Tenga en cuenta que sólo la llamada MQCONN o MQCONNX inicial invoca la salida.

Propiedades de salida de API

En un servidor, las salidas de API pueden registrar una estructura MQXEPO en el momento de la inicialización. La estructura MQXEPO contiene el campo ExitProperties que detalla el grupo de propiedades en las que está interesada la salida. Esto tiene el efecto de generar un descriptor de contexto de propiedad de mensaje independiente que la salida puede manipular por separado de cualquier descriptor de contexto de propiedad de mensaje de aplicación.

En un cliente, las propiedades de salida de API no están soportadas. Si se intenta registrar un nombre de grupo de propiedades en un cliente, la función falla con un código de razón de MQRC_EXIT_PROPS_NOT_SUPPORTED.

Restitución-MQ_BACK_EXIT

MQ_BACK_EXIT proporciona una función de salida de restitución para realizar *antes* y *después* del proceso de restitución. Utilice el identificador de función MQXF_BACK con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada de restitución.

La interfaz para esta función es:

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Inicio-MQ_BEGIN_EXIT

MQ_BEGIN_EXIT proporciona una función de salida de inicio para realizar *antes* y *después del proceso de llamada* MQBEGIN. Utilice el identificador de función MQXF_BEGIN con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada MQBEGIN.

La interfaz para esta función es:

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pBeginOptions (PMQBO)-entrada/salida
Puntero a las opciones de inicio.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason);
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Devolución de llamada-MQ_CALLBACK_EXIT

MQ_CALLBACK_EXIT proporciona una función de salida para realizar *antes* y *después* del proceso de devolución de llamada. Utilice el identificador de función MQXF_CALLBACK con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada *before* y *after* .

La interfaz para esta función es:

```
MQ_CALLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts,
    &pBuffer, &pMQCBContext)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida

ExitContext (MQAXC)-entrada/salida
Estructura de contexto de salida

Hconn (MQHCONN)-entrada/salida
Descriptor de contexto de conexión

pMsgDesc
Descriptor de mensaje

pGetMsgOpts
Opciones que controlan la acción de MQGET

pBuffer
Área para contener los datos del mensaje

pMQCBContext
Datos de contexto para la devolución de llamada

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts, &pBuffer,
 &pContext);
```

La salida debe coincidir con el siguiente prototipo de función 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 conatin the message data */
PPMQCBC   ppContext;)     /* Context data for the callback */
```

Notas de uso

1. La salida de devolución de llamada se invoca antes de que se invoque al consumidor y después de que se haya completado la función de consumidor del consumidor. Aunque las estructuras MQMD y MQGMO son modificables, el cambio de los valores de la salida anterior no reconduce la recuperación de un mensaje de la cola, ya que el mensaje ya se ha eliminado de la cola para entregarlo a la función de consumidor.

Gestionar funciones de devolución de llamada-MQ_CB_EXIT

MQ_CB_EXIT proporciona una función de salida para realizar *antes* y *después* de la llamada MQCB. Utilice el identificador de función MQXF_CB con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada MQCB.

La interfaz para esta función es:

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &pCallbackDesc,
            &Hobj, &pMsgDesc, &pGetMsgOpts, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida

ExitContext (MQAXC)-entrada/salida
Estructura de contexto de salida

Hconn (MQHCONN)-entrada/salida
Descriptor de contexto de conexión

Operación (MQLONG)-entrada/salida
Valor de operación

pCallbackDesc (PMQCBD)-entrada/salida
Descriptor de devolución de llamada

Hobj (MQHOBJ)-entrada/salida
Descriptor de contexto del objeto

pMsgDesc (PMQMD)-entrada/salida
Descriptor de mensaje

pGetMsgOpts (PMQGMO)-entrada/salida
Opciones que controlan la acción de MQCB

CompCode (MQLONG)-entrada/salida
Código de terminación

Razón (MQLONG)-entrada/salida
Código de razón que califica a CompCode

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
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.
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &Hobj, &pMsgDesc,
           &pGetMsgOpts, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Cerrar-MQ_CLOSE_EXIT

MQ_CLOSE_EXIT proporciona una función de salida de cierre para realizar *antes* y *después del proceso de la llamada MQCLOSE de* . Utilice el identificador de función MQXF_CLOSE con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de* las funciones de salida de llamada MQCLOSE.

La interfaz para esta función es:

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHobj,
               &Options, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pHobj (PMQHOBJ)-entrada
Puntero al descriptor de contexto de objeto.

Opciones (MQLONG)-entrada/salida
Opciones de cierre.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED, la función de salida puede establecer el campo de código de razón en cualquier valor MQRC_ * válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext,&Hconn, &pHobj, &Options,
               &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Confirmar-MQ_CMIT_EXIT

MQ_CMIT_EXIT proporciona una función de salida de confirmación para realizar *antes* y *después* del proceso de confirmación. Utilice el identificador de función MQXF_CMIT con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada de confirmación.

Si una operación de confirmación falla y la transacción se restituye, la llamada MQCMIT falla con MQCC_WARNING y MQRC_BACKED_OUT. Estos códigos de retorno y de razón se pasan a cualquier *después* de las funciones de salida MQCMIT para dar a las funciones de salida una indicación de que la unidad de trabajo se ha restituido.

La interfaz para esta función es:

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext,&Hconn, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Notas de uso

1. La interfaz de función MQ_GET_EXIT descrita aquí se utiliza tanto para la función de salida MQXF_GET como para la función de salida “MQXF_DATA_CONV_ON_GET” en la página 1122 .

   Se definen puntos de entrada separados para estas dos funciones de salida, por lo que para interceptar *ambos* la llamada MQXEP debe utilizarse dos veces; para esta llamada utilice el identificador de función MQXF_GET.

   Puesto que la interfaz MQ_GET_EXIT es la misma para MQXF_GET y MQXF_DATA_CONV_ON_GET, se puede utilizar una única función de salida para ambos; el campo *Function* de la estructura MQAXP indica qué función de salida se ha invocado. De forma alternativa, se puede utilizar la llamada MQXEP para registrar distintas funciones de salida para los dos casos.

Extensión de conexión y conexión-MQ_CONNX_EXIT

MQ_CONNX_EXIT proporciona:

- Función de salida de conexión para realizar *antes* y *después* del proceso MQCONN
- Función de salida de extensión de conexión para realizar *antes* y *después* del proceso MQCONNX

La misma interfaz, descrita aquí, se invoca para las funciones de salida de llamada MQCONN y MQCONNX.

Cuando el agente de canal de mensajes (MCA) responde a una conexión de cliente de entrada, el MCA puede conectarse y realizar un número de llamadas de API de WebSphere MQ antes de que se conozca completamente el estado del cliente. Estas llamadas de API llaman a las funciones de salida de API con MQAXC basadas en el propio programa MCA (por ejemplo, en los campos UserId y ConnectionName de MQAXC).

Cuando el MCA responde a las llamadas de API de cliente de entrada posteriores, la estructura MQAXC se basa en el cliente de entrada, estableciendo los campos UserId y ConnectionName de forma adecuada.

El nombre del gestor de colas establecido por la aplicación en una llamada MQCONN o MQCONNX se pasa a la llamada de conexión subyacente. Cualquier intento de un *anterior a* MQ_CONNX_EXIT de cambiar el nombre del gestor de colas no tiene ningún efecto.

Utilice los identificadores de función MQXF_CONN y MQXF_CONNX con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de las funciones de salida de llamada MQCONN y MQCONNX de* .

Una salida MQ_CONNX_EXIT llamada por la razón MQXR_BEFORE *no* debe emitir ninguna llamada de API WebSphere MQ , ya que no se ha configurado el entorno correcto en este momento.

Un MQ_CONNX_EXIT no puede llamar a MQDISC desde una llamada de salida de API para la conexión para la que se está llamando. Esta restricción es aplicable a las salidas de API de cliente y servidor.

La interfaz con MQCONN y MQCONNX es idéntica:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOpts,
               &pHconn, &CompCode, &Reason);
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

pQMgrNombre (PMQCHAR)-entrada
Puntero al nombre de gestor de colas proporcionado en la llamada MQCONNX. La salida no debe cambiar este nombre en la llamada MQCONN o MQCONNX.

pConnectOpts (PMQCNO)-entrada/salida
Puntero a las opciones que controlan la acción de la llamada MQCONNX.

Para obtener más detalles, consulte "MQCNO - Opciones de conexión" en la página 297.

Para la función de salida MQXF_CONN, pConnectOpts apunta a la estructura de opciones de conexión predeterminada (MQCNO_DEFAULT).

pHconn (PMQHCONN)-entrada
Puntero al descriptor de conexión.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (terminación parcial)

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOps,
               &pHconn, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Notas de uso

1. La interfaz de función MQ_CONNX_EXIT descrita aquí se utiliza tanto para la llamada MQCONN como para la llamada MQCONNX. Sin embargo, se definen puntos de entrada separados para estas dos llamadas. Para interceptar *ambas* llamadas, la llamada MQXEP debe utilizarse al menos dos veces, una con el identificador de función MQXF_CONN y otra con MQXF_CONNX.

   Puesto que la interfaz MQ_CONNX_EXIT es la misma para MQCONN y MQCONNX, se puede utilizar una única función de salida para ambas llamadas; el campo *Function* de la estructura MQAXP indica qué llamada está en curso. De forma alternativa, la llamada MQXEP se puede utilizar para registrar distintas funciones de salida para las dos llamadas.
2. Cuando un agente de canal de mensajes (MCA) responde a una conexión de cliente de entrada, el MCA puede emitir una serie de llamadas MQ antes de que se conozca completamente el estado del cliente. Estas llamadas MQ hacen que las funciones de salida de API se invoquen con la estructura MQAXC que contiene datos relacionados con el MCA, y no con el cliente (por ejemplo, el identificador de usuario y el nombre de conexión). Sin embargo, una vez que el estado del cliente es totalmente conocido, las llamadas MQ posteriores dan como resultado que se invoquen las funciones de salida de API con los datos de cliente adecuados en la estructura MQAXC.
3. Todas las funciones de salida MQXR_BEFORE se invocan antes de que el gestor de colas realice cualquier validación de parámetro. Por lo tanto, es posible que los parámetros no sean válidos (incluidos los punteros no válidos para las direcciones de los parámetros).

   La función MQ_CONNX_EXIT se invoca antes de que el gestor de colas realice comprobaciones de autorización.
4. La función de salida no debe cambiar el nombre del gestor de colas especificado en la llamada MQCONN o MQCONNX. Si la función de salida cambia el nombre, los resultados no están definidos.
5. Una función de salida MQXR_BEFORE para MQ_CONNX_EXIT no puede emitir llamadas de MQ distintas de MQXEP.

Devolución de llamada de control-MQ_CTL_EXIT

MQ_CTL_EXIT proporciona una función de salida de solicitud de suscripción para realizar *antes* y *después* del proceso de devolución de llamada de control. Utilice el identificador de función MQXF_CTL con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada de retorno de control.

La interfaz para esta función es:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason)
```

donde los parámetros son:

Hconn (MQHCONN)-entrada/salida
Descriptor de conexión.

Entrada/salida de operación (MQLONG)
La operación que se está procesando en la devolución de llamada definida para el descriptor de objeto especificado

Entrada/salida de ControlOpts (MQCTLO)
Opciones que controlan la acción de MQCTL

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Desconectar-MQ_DISC_EXIT

MQ_DISC_EXIT proporciona una función de salida de desconexión para realizar *antes* y *después* del proceso de salida MQDISC. Utilice el identificador de función MQXF_DISC con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de* las funciones de salida de llamada MQDISC.

La interfaz para esta función es

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
              &CompCode, &Reason);
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

pHconn (PMQHCONN)-entrada
Puntero al descriptor de conexión.

Para la llamada MQDISC anterior, el valor de este campo es uno de los siguientes:

- El descriptor de conexión devuelto en la llamada MQCONN o MQCONNX
- Cero, para entornos en los que un adaptador específico del entorno se ha conectado al gestor de colas
- Un valor establecido por una invocación de función de salida anterior

Para la llamada MQDISC posterior, el valor de este campo es cero o un valor establecido por una invocación de función de salida anterior.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Terminación parcial

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
PMQHCONN        pHconn;         /* Ptr to Connection handle */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
              &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Obtener-MQ_GET_EXIT

MQ_GET_EXIT proporciona una función de obtención de salida para realizar *antes* y *después del proceso de la llamada MQGET de* .

Hay dos identificadores de función:

1. Utilice MQXF_GET con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada MQGET.
2. Consulte “MQXF_DATA_CONV_ON_GET” en la página 1122 para obtener información sobre cómo utilizar el identificador de función MQXF_DATA_CONV_ON_GET.

La interfaz para esta función es:

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

Hobj (MQHOBJ)-entrada/salida
El manejador del objeto.

pMsgDesc (PMQMD)-entrada/salida
Puntero al descriptor de mensaje.

pGetMsgOpts (PMQGMO)-entrada/salida
Puntero para obtener opciones de mensaje.

BufferLength (MQLONG)-entrada/salida
Longitud del almacenamiento intermedio de mensajes.

pBuffer (PMQBYTE)-entrada/salida
Puntero al almacenamiento intermedio de mensajes.

pDataLength (PMQLONG)-entrada/salida
Puntero al campo de longitud de datos.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Notas de uso

1. La interfaz de función MQ_GET_EXIT descrita aquí se utiliza tanto para la función de salida MQXF_GET como para la función de salida “MQXF_DATA_CONV_ON_GET” en la página 1122 .

   Se definen puntos de entrada separados para estas dos funciones de salida, por lo que para interceptar *ambos* la llamada MQXEP debe utilizarse dos veces; para esta llamada utilice el identificador de función MQXF_GET.

   Puesto que la interfaz MQ_GET_EXIT es la misma para MQXF_GET y MQXF_DATA_CONV_ON_GET, se puede utilizar una única función de salida para ambos; el campo *Function* de la estructura MQAXP indica qué función de salida se ha invocado. De forma alternativa, se puede utilizar la llamada MQXEP para registrar distintas funciones de salida para los dos casos.

MQXF_DATA_CONV_ON_GET

Consulte MQ_GET_EXIT para obtener información sobre la interfaz para esta llamada y una declaración de lenguaje C de ejemplo.

Notas de uso

Si está registrado, se llama a este punto de entrada cuando llegan mensajes a la aplicación pero antes de que se haya producido cualquier conversión de datos. Esto puede ser útil si la salida de API necesita realizar el proceso, como el descifrado o la descompresión, antes de que el mensaje se pase a la conversión de datos. La salida puede, si es necesario, hacer que se elude la conversión de datos devolviendo MQXCC_SUPPRESS_FUNCTION; para obtener más información, consulte la estructura MQAXP .

El registro para este punto de entrada en un cliente tiene el efecto de hacer que la conversión de datos se realice localmente en la máquina cliente. Por lo tanto, para que el funcionamiento sea correcto, es posible que sea necesario instalar las salidas de conversión de aplicaciones en el cliente. Tenga en cuenta que MQXF_DATA_CONV_ON_GET también se utiliza para el consumo asíncrono.

Cuando utilice la llamada MQ_GET_EXIT, utilice MQXF_DATA_CONV_ON_GET, con la razón de salida MQXR_BEFORE, para registrar una función de salida de conversión de datos MQGET *anterior a* .

No hay ninguna función de salida MQXR_AFTER para MQXF_DATA_CONV_ON_GET; la función de salida MQXR_AFTER para MQXF_GET proporciona la capacidad necesaria para el proceso de salida después de la conversión de datos.

Se definen puntos de entrada separados para la llamada MQ_GET_EXIT, por lo que para interceptar *ambas* funciones de salida, la llamada MQXEP debe utilizarse dos veces; para esta llamada, utilice el identificador de función MQXF_DATA_CONV_ON_GET.

Puesto que la interfaz MQ_GET_EXIT es la misma para MQXF_GET y MQXF_DATA_CONV_ON_GET, se puede utilizar una única función de salida para ambos; el campo *Function* de la estructura MQAXP indica qué función de salida se ha invocado. De forma alternativa, se puede utilizar la llamada MQXEP para registrar distintas funciones de salida para los dos casos.

Inicialización-MQ_INIT_EXIT

MQ_INIT_EXIT proporciona inicialización de nivel de conexión, indicada estableciendo ExitReason en MQAXP en MQXR_CONNECTION.

Durante la inicialización, tenga en cuenta lo siguiente:

- La función MQ_INIT_EXIT llama a MQXEP para registrar los verbos de la API WebSphere MQ y los puntos ENTRY y EXIT en los que está interesado.
- Las salidas no necesitan interceptar todos los verbos de la API de WebSphere MQ . Las funciones de salida sólo se invocan si se ha registrado un interés.
- El almacenamiento que va a utilizar la salida se puede adquirir al inicializarlo.
- Si falla una llamada a esta función, la llamada MQCONN o MQCONNX que la ha invocado también falla con un CompCode y una razón que dependen del valor del campo ExitResponse en MQAXP.
- Una salida MQ_INIT_EXIT no debe emitir llamadas de API de WebSphere MQ , porque el entorno correcto no se ha configurado en este momento.
- Si un MQ_INIT_EXIT falla con MQXCC_FAILED, el gestor de colas vuelve de la llamada MQCONN o MQCONNX que le ha llamado con MQCC_FAILED y MQRC_API_EXIT_ERROR.
- Si el gestor de colas encuentra un error al inicializar el entorno de ejecución de la función de salida de API antes de invocar la primera MQ_INIT_EXIT, el gestor de colas devuelve la llamada MQCONN o MQCONNX que ha invocado MQ_INIT_EXIT con MQCC_FAILED y MQRC_API_EXIT_INIT_ERROR.

La interfaz con MQ_INIT_EXIT es:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

CompCode (MQLONG)-entrada/salida
Puntero a código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Puntero al código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

CompCode y Reason devueltos a la aplicación dependen del valor del campo ExitResponse en MQAXP.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        MQAXP           ExitParms;      /* Exit parameter structure */
        MQAXC           ExitContext;    /* Exit context structure */
```

```
        MQLONG          CompCode;       /* Completion code */
        MQLONG          Reason;         /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Notas de uso

1. La función MQ_INIT_EXIT puede emitir la llamada MQXEP para registrar las direcciones de las funciones de salida para las llamadas MQ concretas que se van a interceptar. No es necesario interceptar todas las llamadas de MQ o interceptar las llamadas MQXR_BEFORE y MQXR_AFTER. Por ejemplo, una suite de salida podría elegir interceptar sólo la llamada MQXR_BEFORE de MQPUT.
2. El almacenamiento que deben utilizar las funciones de salida en la suite de salida puede ser adquirido por la función MQ_INIT_EXIT. De forma alternativa, las funciones de salida pueden adquirir almacenamiento cuando se invocan, como y cuando sea necesario. Sin embargo, todo el almacenamiento debe liberarse antes de que termine la suite de salida; la función MQ_TERM_EXIT puede liberar el almacenamiento o una función de salida invocada anteriormente.
3. Si MQ_INIT_EXIT devuelve MQXCC_FAILED en el campo *ExitResponse* de MQAXP, o falla de alguna otra forma, la llamada MQCONN o MQCONNX que ha hecho que se invoque MQ_INIT_EXIT también falla, con los parámetros *CompCode* y *Reason* establecidos en los valores adecuados.
4. Una función MQ_INIT_EXIT no puede emitir llamadas MQ que no sean MQXEP.

Consultar-MQ_INQ_EXIT

MQ_INQ_EXIT proporciona una función de salida de consulta para realizar *antes* y *después del proceso de la llamada MQINQ de* . Utilice el identificador de función MQXF_INQ con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de* las funciones de salida de llamada MQINQ.

La interfaz para esta función es:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

Hobj (MQHOBJ)-entrada
El manejador del objeto.

SelectorCount (MQLONG)-entrada
Recuento de selectores

pSelectors (PMQLONG)-entrada/salida
Puntero a matriz de valores de selector.

IntAttrCount (MQLONG)-entrada
Recuento de atributos enteros.

pIntAttrs (PMQLONG)-entrada/salida
Puntero a matriz de valores de atributo de entero.

CharAttrLength (MQLONG)-entrada/salida
Longitud de matriz de atributos de caracteres.

pCharAttrs (PMQCHAR)-entrada/salida
Puntero a matriz de atributos de carácter.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Abrir-MQ_OPEN_EXIT

MQ_OPEN_EXIT proporciona una función de salida abierta para realizar *antes* y *después* del proceso de llamada MQOPEN. Utilice el identificador de función MQXF_OPEN con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de* las funciones de salida de llamada MQOPEN.

La interfaz para esta función es

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
              &pHobj, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pObjDesc (PMQOD)-entrada/salida
Puntero al descriptor de objeto.

Opciones (MQLONG)-entrada/salida
Opciones de apertura.

pHobj (PMQHOBJ)-entrada
Puntero al descriptor de contexto de objeto.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Terminación parcial

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
              &pHobj, &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Put-MQ_PUT_EXIT

MQ_PUT_EXIT proporciona una función de salida de transferencia para realizar *antes* y *después* del proceso de llamada MQPUT. Utilice el identificador de función MQXF_PUT con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de* las funciones de salida de llamada MQPUT.

La interfaz para esta función es:

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

Hobj (MQHOBJ)-entrada/salida
El manejador del objeto.

pMsgDesc (PMQMD)-entrada/salida
Puntero al descriptor de mensaje.

pPutMsgOpts (PMQPMO)-entrada/salida
Puntero para colocar opciones de mensaje.

BufferLength (MQLONG)-entrada/salida
Longitud del almacenamiento intermedio de mensajes.

pBuffer (PMQBYTE)-entrada/salida
Puntero al almacenamiento intermedio de mensajes.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Notas de uso

- Los mensajes de informe generados por el gestor de colas se saltan el proceso de llamada normal. Como resultado, la función MQ_PUT_EXIT o la función MQPUT1 no pueden interceptar dichos mensajes. Sin embargo, los mensajes de informe generados por el agente de canal de mensajes se procesan normalmente y, por lo tanto, pueden ser interceptados por la función MQ_PUT_EXIT o la función MQ_PUT1_EXIT . Para asegurarse de interceptar todos los mensajes de informe generados por el MCA, deben utilizarse MQ_PUT_EXIT y MQ_PUT1_EXIT .

Put1 - MQ_PUT1_EXIT

MQ_PUT1_EXIT proporciona una función de salida *put one message only* para realizar *antes* y *después del proceso de llamada* MQPUT1 . Utilice el identificador de función MQXF_PUT1 con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de las funciones de salida de llamada* MQPUT1 .

La interfaz para esta función es:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
              &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pObjDesc (PMQOD)-entrada/salida
Puntero al descriptor de objeto.

pMsgDesc (PMQMD)-entrada/salida
Puntero al descriptor de mensaje.

pPutMsgOpts (PMQPMO)-entrada/salida
Puntero para colocar opciones de mensaje.

BufferLength (MQLONG)-entrada/salida
Longitud del almacenamiento intermedio de mensajes.

pBuffer (PMQBYTE)-entrada/salida
Puntero al almacenamiento intermedio de mensajes.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
              &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Establecer-MQ_SET_EXIT

MQ_SET_EXIT proporciona una función de salida de conjunto para realizar *antes* y *después del proceso de la llamada MQSET de* . Utilice el identificador de función MQXF_SET con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después de* las funciones de salida de llamada MQSET.

La interfaz para esta función es:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttr, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

Hobj (MQHOBJ)-entrada
El manejador del objeto.

SelectorCount (MQLONG)-entrada
Recuento de selectores

pSelectors (PMQLONG)-entrada/salida
Puntero a matriz de valores de selector.

IntAttrCount (MQLONG)-entrada
Recuento de atributos enteros.

pIntAttrs (PMQLONG)-entrada/salida
Puntero a matriz de valores de atributo de entero.

CharAttrLength (MQLONG)-entrada/salida
Longitud de matriz de atributos de caracteres.

pCharAttrs (PMQCHAR)-entrada/salida
Puntero a valores de atributo de carácter.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Estado-MQ_STAT_EXIT

MQ_STAT_EXIT proporciona una función de salida de estado para realizar *antes* y *después* del proceso de llamada MQSTAT. Utilice el identificador de función MQXF_STAT con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de llamada MQSTAT.

La interfaz para esta función es:

```
MQ_STAT_EXIT (&ExitParms, &ExitContext, &Hconn, &Type, &pStatus
              &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

Tipo (MQLONG)-entrada
Tipo de información de estado a recuperar.

pStatus (PMQSTS)-salida
Puntero al almacenamiento intermedio de estado.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Terminación-MQ_TERM_EXIT

MQ_TERM_EXIT proporciona terminación a nivel de conexión, registrada con un identificador de función de MQXF_TERM y ExitReason MQXR_CONNECTION. Si está registrado, se llama a MQ_TERM_EXIT una vez para cada solicitud de desconexión.

Como parte de la terminación, el almacenamiento que ya no necesita la salida se puede liberar y se puede realizar cualquier limpieza necesaria.

Si un MQ_TERM_EXIT falla con MQXCC_FAILED, el gestor de colas devuelve el MQDISC que lo ha llamado con MQCC_FAILED y MQRC_API_EXIT_ERROR.

Si el gestor de colas encuentra un error al terminar el entorno de ejecución de la función de salida de API después de invocar el último MQ_TERM_EXIT, el gestor de colas devuelve la llamada MQDISC que ha invocado MQ_TERM_EXIT con MQCC_FAILED y MQRC_API_EXIT_TERM_ERROR

La interfaz para esta función es:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED, la función de salida puede establecer el campo de código de razón en cualquier valor MQRC_ * válido.

CompCode y Reason devueltos a la aplicación dependen del valor del campo ExitResponse en MQAXP.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Notas de uso

1. La función MQ_TERM_EXIT es opcional. No es necesario que una suite de salida registre una salida de terminación si no hay que realizar ningún proceso de terminación.

   Si las funciones pertenecientes a la suite de salida adquieren recursos durante la conexión, una función MQ_TERM_EXIT es un punto conveniente en el que liberar esos recursos, por ejemplo, liberando almacenamiento obtenido dinámicamente.
2. Si se registra una función MQ_TERM_EXIT cuando se emite la llamada MQDISC, la función de salida se invoca después de que se hayan invocado todas las funciones de salida MQDISC.
3. Si MQ_TERM_EXIT devuelve MQXCC_FAILED en el campo *ExitResponse* de MQAXP, o falla de alguna otra forma, la llamada MQDISC que ha hecho que se invoque MQ_TERM_EXIT también falla, con los parámetros *CompCode* y *Reason* establecidos en los valores adecuados.

Registrar suscripción-MQ_SUB_EXIT

MQ_SUB_EXIT proporciona una función de salida para realizar *antes* y *después* del proceso de reregistro de suscripción. Utilice el identificador de función MQXF_SUB con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar *antes* y *después* de las funciones de salida de registrationcall de suscripción.

La interfaz para esta función es:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub, &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada/salida
Descriptor de conexión.

pSubDesc-entrada/salida
Matriz de selectores de atributos.

pHobj -entrada/salida
Descriptor de contexto del objeto

pHsub (MQHOBJ) entrada/salida
Manejador de suscripciones

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub,
 &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Solicitud de suscripción-MQ_SUBRQ_EXIT

MQ_SUBRQ_EXIT proporciona una función de salida de solicitud de suscripción para realizar *antes* y *después* del proceso de solicitud de suscripción. Utilice el identificador de función MQXF_SUBRQ con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada de solicitud de suscripción *before* y *after* .

La interfaz para esta función es:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
    &CompCode, &Reason)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada/salida
Descriptor de conexión.

pHsub (MQHOBJ) entrada/salida
Manejador de suscripciones

Entrada/salida de acción (MQLONG)
Acción

pSubRqOpts (MQSRO) entrada/salida

CompCode (MQLONG)-entrada/salida
Código de terminación, cuyos valores válidos son:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
Llamada fallida

Razón (MQLONG)-entrada/salida
Código de razón que califica el código de terminación.

Si el código de terminación es MQCC_OK, el único valor válido es:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si el código de terminación es MQCC_FAILED o MQCC_WARNING, la función de salida puede establecer el campo de código de razón a cualquier valor de MQRC_* válido.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
        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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
 &CompCode, &Reason);
```

La salida debe coincidir con el siguiente prototipo de función 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 proporciona una función de salida xa_close para realizar antes y después del proceso xa_close. Utilice el identificador de función MQXF_XACLOSE con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_close.

La interfaz para esta función es:

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXa_info (PMQCHAR)-entrada/salida
Información del gestor de recursos específico de la instancia.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función 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 proporciona una función de salida xa_commit para realizar antes y después del proceso xa_commit. Utilice el identificador de función MQXF_XACOMMIT con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_commit.

La interfaz para esta función es:

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función 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 proporciona una función de salida xa_complete para realizar antes y después del proceso xa_complete. Utilice el identificador de función MQXF_XACOMPLETE con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_complete.

La interfaz para esta función es:

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetval, &Rmid, &Flags,
&XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pHandle (PMQLONG)-entrada/salida
Puntero a operación asíncrona.

pRetVal (PMQLONG)-entrada/salida
Valor de retorno de la operación asíncrona.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetval, &Rmid, &Flags,
&XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función 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 proporciona una función de salida xa_end para realizar antes y después del proceso xa_end. Utilice el identificador de función MQXF_XAEND con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_end.

La interfaz para esta función es:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función 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_OLVIDO_SALIDA

XA_OLVI_EXIT proporciona una función de salida xa_olvidar para realizar antes y después del proceso xa_olvidar. Utilice el identificador de función MQXF_XAFORGET con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_forget.

La interfaz para esta función es:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función 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 proporciona una función de salida xa_open para realizar antes y después del proceso xa_open. Utilice el identificador de función MQXF_XAOPEN con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_open.

La interfaz para esta función es:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXa_info (PMQCHAR)-entrada/salida
Información del gestor de recursos específico de la instancia.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función 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 proporciona una función de salida xa_prepare para realizar antes y después del proceso xa_prepare. Utilice el identificador de función MQXF_XAPREPARE con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_prepare.

La interfaz para esta función es:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función 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 proporciona una función de salida xa_recover para realizar antes y después del proceso xa_recover. Utilice el identificador de función MQXF_XARECOVER con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_recover.

La interfaz para esta función es:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Recuento (MQLONG)-entrada/salida
Máximo de XID en matriz XID

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función 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 proporciona una función de salida xa_rollback para realizar antes y después del proceso xa_rollback. Utilice el identificador de función MQXF_XAROLLBACK con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_rollback.

La interfaz para esta función es:

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función 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 proporciona una función de salida xa_start para realizar antes y después del proceso xa_start. Utilice el identificador de función MQXF_XASTART con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de xa_start.

La interfaz para esta función es:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función 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 proporciona una función de salida ax_reg para realizar antes y después del proceso ax_reg. Utilice el identificador de función MQXF_AXREG con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de ax_reg.

La interfaz para esta función es:

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode)
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Hconn (MQHCONN)-entrada
Descriptor de conexión.

pXID (MQPTR)-entrada/salida
ID de rama de transacción.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función 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 proporciona una función de salida ax_unreg para realizar antes y después del proceso ax_unreg. Utilice el identificador de función MQXF_AXUNREG con las razones de salida MQXR_BEFORE y MQXR_AFTER para registrar las funciones de salida de llamada antes y después de ax_unreg.

La interfaz para esta función es:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

donde los parámetros son:

ExitParms (MQAXP)-entrada/salida
Estructura de parámetros de salida.

ExitContext (MQAXC)-entrada/salida
Salir de la estructura de contexto.

Rmid (MQLONG)-entrada/salida
Identificador del gestor de recursos.

Distintivos (MQLONG)-entrada/salida
Opciones del gestor de recursos.

XARetCode (MQLONG)-entrada/salida
Respuesta de la llamada XA.

Invocación de lenguaje C

El gestor de colas define lógicamente las variables siguientes:

```
     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 */
```

A continuación, el gestor de colas llama lógicamente a la salida como se indica a continuación:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

La salida debe coincidir con el siguiente prototipo de función 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 */
```

Información general sobre la invocación de funciones de salida

Este tema proporciona algunas directrices generales para ayudarle a planificar las salidas, especialmente relacionadas con el manejo de errores y sucesos inesperados.

Error de salida

Si una función de salida termina de forma anómala después de una llamada MQGET destructiva, fuera de punto de sincronismo, pero antes de que el mensaje se haya pasado a la aplicación, el manejador de salida puede recuperarse de la anomalía y pasar el control a la aplicación.

En este caso, es posible que se pierda el mensaje. Esto es como lo que sucede cuando una aplicación falla inmediatamente después de recibir un mensaje de una cola.

La llamada MQGET puede completarse con MQCC_FAILED y MQRC_API_EXIT_ERROR.

Si una función de salida de llamada de API *before* termina de forma anómala, el manejador de salida puede recuperarse de la anomalía y pasar el control a la aplicación sin procesar la llamada de API. En este caso, la función de salida debe recuperar los recursos que posee.

Si las salidas encadenadas están en uso, las salidas de llamada de API *after* para cualquier salida de llamada de API *before* que se haya controlado correctamente pueden ser controladas por sí mismas. La llamada de API puede fallar con MQCC_FAILED y MQRC_API_EXIT_ERROR.

Manejo de errores de ejemplo para funciones de salida

El diagrama siguiente muestra los puntos (e*N*) en los que se pueden producir errores. Es sólo un ejemplo para mostrar cómo se comportan las salidas y debe leerse junto con la tabla siguiente. En este ejemplo, se invocan dos funciones de salida antes y después de cada llamada de API para mostrar el comportamiento con salidas encadenadas.

```
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 tabla siguiente lista las acciones que se deben realizar en cada punto de error. Sólo se ha cubierto un subconjunto de los puntos de error, ya que las reglas que se muestran aquí se pueden aplicar a todos los demás. Son las acciones que especifican el comportamiento previsto en cada caso.

Tabla 595. Errores de salida de API y acciones adecuadas a realizar

Err Pt	**Descripción**	**Acciones**
e1	Error al configurar el entorno.	1. Deshacer configuración de entorno según sea necesario 2. No hay funciones de salida de unidad 3. Error de MQCONN con MQCC_FAILED, MQRC_API_EXIT_LOAD_ERROR
e2	La función MQ_INIT_EXIT se completa con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED: 1. Limpiar entorno 2. Error de MQCONN con MQCC_FAILED, MQRC_API_EXIT_INIT_ERROR • Para MQXCC_ * 1. Actuar como para los valores de MQXCC_ * y MQXR2_ *[1] 2. Limpiar entorno
e3	*Antes* , la función MQ_CONNX_EXIT 1 finaliza con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED: 1. Unidad MQ_TERM_EXIT, función 2. Limpiar entorno 3. Error de llamada MQCONN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ * 1. Actuar como para los valores de MQXCC_ * y MQXR2_ *[1] 2. Función MQ_TERM_EXIT de unidad, si es necesario 3. Limpiar entorno si es necesario

Tabla 595. Errores de salida de API y acciones adecuadas a realizar (continuación)

Err Pt	Descripción	Acciones
e4	*Antes* , la función MQ_CONNX_EXIT 2 se completa con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED: 1. Unidad *después de la función* MQ_CONNX_EXIT 1 2. Unidad MQ_TERM_EXIT, función 3. Limpiar entorno 4. Error de llamada MQCONN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ * 1. Actuar como para los valores de MQXCC_ * y MQXR2_ *[1] 2. Unidad *después* de la función MQ_CONNX_EXIT 1 si la salida no se suprime 3. Función MQ_TERM_EXIT de unidad, si es necesario 4. Limpiar entorno si es necesario
e5	La llamada MQCONN falla.	1. Pasar MQCONN CompCode y Razón 2. Unidad *después* de la función MQ_CONNX_EXIT 2 si *antes* MQ_CONNX_EXIT 2 se ha ejecutado correctamente y la salida no se suprime 3. Unidad *después* de la función MQ_CONNX_EXIT 1 si *antes* MQ_CONNX_EXIT 1 se ha ejecutado correctamente y la salida no se suprime 4. Unidad MQ_TERM_EXIT, función 5. Limpiar entorno
e6	*Después de* la función MQ_CONNX_EXIT 2 se completa con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED: 1. Unidad *después de la función* MQ_CONNX_EXIT 1 2. Complete la llamada MQCONN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ * 1. Actuar como para los valores de MQXCC_ * y MQXR2_ *[1] 2. Unidad *después de la función* MQ_CONNX_EXIT 1 si es necesario
e7	*Después de que la función* MQ_CONNX_EXIT 1 se complete con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED, complete la llamada MQCONN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ *, actúe como para los valores de MQXCC_ * y MQXR2_ *[1]
e8	*Antes* la función MQ_OPEN_EXIT 1 se completa con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED, complete la llamada MQOPEN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ *, actúe como para los valores de MQXCC_ * y MQXR2_ *[1]

Tabla 595. Errores de salida de API y acciones adecuadas a realizar (continuación)		
Err Pt	**Descripción**	**Acciones**
e9	*Antes* , la función MQ_OPEN_EXIT 2 finaliza con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED: 1. Unidad *después de la función* MQ_OPEN_EXIT 1 2. Complete la llamada MQOPEN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ *, actúe como para los valores de MQXCC_ * y MQXR2_ *[1]
e10	La llamada MQOPEN falla	1. Pasar MQOPEN CompCode y Razón 2. Unidad *después de la función* MQ_OPEN_EXIT 2 si la salida no se suprime 3. Unidad *después de la función* MQ_OPEN_EXIT 1 si la salida no se ha suprimido y si las salidas encadenadas no se han suprimido
e11	*Después de que la función MQ_OPEN_EXIT 2 de* se complete con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED: 1. Unidad *después de la función* MQ_OPEN_EXIT 1 2. Complete la llamada MQOPEN con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ * 1. Actuar como para los valores de MQXCC_ * y MQXR2_ *[1] 2. Unidad *después de la función* MQ_OPEN_EXIT 1 si no se suprime la salida
e25	*Después de* la función MQ_DISC_EXIT 2 se completa con: • MQXCC_FAILED • MQXCC_ *	• Para MQXCC_FAILED: 1. Unidad *después de la función* MQ_DISC_EXIT 1 2. Unidad MQ_TERM_EXIT, función 3. Entorno de ejecución de salida de limpieza 4. Completar llamada MQDISC con MQCC_FAILED, MQRC_API_EXIT_ERROR • Para MQXCC_ * 1. Actuar como para los valores de MQXCC_ * y MQXR2_ *[1] 2. Unidad MQ_TERM_EXIT, función 3. Entorno de ejecución de salida de limpieza

Nota:

1. Los valores de MQXCC_ * y MQXR2_* y sus acciones correspondientes se definen en Cómo procesan los gestores de colas las funciones de salida.

Los campos ExitResponse se han establecido incorrectamente

Este tema proporciona información sobre lo que sucedería cuando el campo ExitResponse se establece en cualquier valor que no sean los valores soportados.

Si el campo ExitResponse se establece en un valor distinto de uno de los valores soportados, se aplican las acciones siguientes:

- Para una función de salida de API MQCONN o MQDISC *anterior a* :

  - El valor ExitResponse2 se ignora.
  - No se invocan más *antes* de las funciones de salida en la cadena de salida (si las hay); no se emite la propia llamada de API.
  - Para las salidas *anteriores* que se han llamado correctamente, las salidas *posteriores* se llaman en orden inverso.
  - Si está registrado, las funciones de salida de terminación para las funciones de salida *antes* MQCONN o MQDISC de la cadena que se han invocado correctamente se controlan para que se limpien después de estas funciones de salida.
  - La llamada MQCONN o MQDISC falla con MQRC_API_EXIT_ERROR.
- Para una función de salida de API *antes de* WebSphere MQ que no sea MQCONN o MQDISC:
  - El valor ExitResponse2 se ignora.
  - No se invocan más *antes* o *después* de las funciones de conversión de datos en la cadena de salida (si las hay).
  - Para las salidas *anteriores* que se han llamado correctamente, las salidas *posteriores* se llaman en orden inverso.
  - La propia llamada de API de WebSphere MQ no se emite.
  - La llamada de API WebSphere MQ falla con MQRC_API_EXIT_ERROR.
- Para una función de salida de API MQCONN o MQDISC *después de* :
  - El valor ExitResponse2 se ignora.
  - Las funciones de salida restantes que se han llamado correctamente antes de la llamada de API se llaman en orden inverso.
  - Si está registrado, las funciones de salida de terminación para las funciones de salida *antes* o *después* MQCONN o MQDISC de la cadena que se han invocado correctamente se controlan para que se limpien después de la salida.
  - Se devuelve a la aplicación un CompCode del más grave de MQCC_WARNING y el CompCode devuelto por la salida.
  - Se devuelve una razón de MQRC_API_EXIT_ERROR a la aplicación.
  - La llamada de API WebSphere MQ se ha emitido correctamente.
- Para una función de salida de llamada de API *después de* WebSphere MQ distinta de MQCONN o MQDISC:
  - El valor ExitResponse2 se ignora.
  - Las funciones de salida restantes que se han llamado correctamente antes de la llamada de API se llaman en orden inverso.
  - Se devuelve a la aplicación un CompCode del más grave de MQCC_WARNING y el CompCode devuelto por la salida.
  - Se devuelve una razón de MQRC_API_EXIT_ERROR a la aplicación.
  - La llamada de API WebSphere MQ se ha emitido correctamente.
- Para la conversión de datos *before* al obtener la función de salida:
  - El valor ExitResponse2 se ignora.
  - Las funciones de salida restantes que se han llamado correctamente antes de la llamada de API se llaman en orden inverso.
  - El mensaje no se convierte y el mensaje no convertido se devuelve a la aplicación.
  - Se devuelve a la aplicación un CompCode del más grave de MQCC_WARNING y el CompCode devuelto por la salida.
  - Se devuelve una razón de MQRC_API_EXIT_ERROR a la aplicación.
  - La llamada de API WebSphere MQ se ha emitido correctamente.

Nota: Dado que el error es con la salida, es mejor devolver MQRC_API_EXIT_ERROR que devolver MQRC_NOT_CONVERT.

Si una función de salida establece el campo ExitResponse2 en un valor distinto de uno de los valores soportados, en su lugar se asume un valor de MQXR2_DEFAULT_CONTINUATION .

Información de consulta sobre la interfaz de servicios instalables

Esta colección de temas proporciona información de referencia para los servicios instalables.

Las funciones y los tipos de datos se listan en orden alfabético dentro del grupo para cada tipo de servicio.

Cómo se muestran las funciones

Cómo se documentan las funciones de servicios instalables.

Para cada función hay una descripción, incluido el identificador de función (para MQZEP).

Los *parámetros* se muestran en el orden en el que deben aparecer. Todos ellos deben estar presentes.

Cada nombre de parámetro va seguido de su tipo de datos. Estos son los tipos de datos elementales descritos en "Tipos de datos elementales" en la página 218.

También se proporciona la invocación del lenguaje C, después de la descripción de los parámetros.

MQZ_AUTHENTICATE_USER-Autenticar usuario

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 y la invoca el gestor de colas para autenticar un usuario o para establecer campos de contexto de identidad. Se invoca cuando se establece el contexto de aplicación de usuario de WebSphere MQ.

El contexto de aplicación se establece durante las llamadas de conexión en el punto en el que se inicializa el contexto de usuario de la aplicación y en cada punto en el que se cambia el contexto de usuario de la aplicación. Cada vez que se realiza una llamada de conexión, la información de contexto de usuario de la aplicación se vuelve a adquirir en el campo *IdentityContext* .

El identificador de función para esta función (para MQZEP) es MQZID_AUTHENTICATE_USER.

Sintaxis

MQZ_AUTHENTICATE_USER (*QMgrName*, *SecurityParms*, *ApplicationContext*, *IdentityContext*, *CorrelationPtr*, *ComponentData*, *Continuation*, *CompCode*, *Motivo*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

SecurityParms

Tipo: MQCSP-entrada

Parámetros de seguridad. Datos relacionados con el ID de usuario, la contraseña y el tipo de autenticación. Si el atributo AuthenticationType de la estructura MQCSP se especifica como MQCSP_AUTH_USER_ID_AND_PWD, tanto el ID de usuario como la contraseña se comparan con los campos equivalentes del parámetro IdentityContext (MQZIC) para determinar si coinciden con. Para obtener más información, consulte "MQCSP-Parámetros de seguridad" en la página 313.

Durante una llamada MQCONN MQI, este parámetro contiene valores nulos o predeterminados.

ApplicationContext
Tipo: MQZAC-entrada

Contexto de aplicación. Datos relacionados con la aplicación de llamada. Consulte MQZAC-Contexto de aplicación para obtener más detalles.

Durante cada llamada MQCONN o MQCONNX MQI, se vuelve a adquirir la información de contexto de usuario de la estructura MQZAC.

IdentityContext
Tipo: MQZIC-entrada/salida

Contexto de identidad. En input para la función de usuario de autenticación, this identifica el contexto de identidad actual. La función autenticar usuario puede cambiar este, momento en el que el gestor de colas adopta el nuevo contexto de identidad. Consulte MQZIC-Contexto de identidad para obtener más detalles sobre la estructura MQZIC.

CorrelationPtr
Tipo: MQPTR-salida

Puntero de correlación. Especifica la dirección de cualquier dato de correlación. Este puntero se pasa posteriormente a otras llamadas de OAM.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos de en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente's .

La longitud de esta área de datos la pasa el gestor de colas en el parámetro de longitud ComponentDatade la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Distintivo de continuación. Puede especificar los valores siguientes:

MQZCI_DEFAULT
Continuación dependiente de otros componentes.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información en estos códigos de razón, consulte Códigos de razón.

Invocación en C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                   IdentityContext, &CorrelationPtr, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Declare los parámetros pasados al servicio como se indica a continuación:

```
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-Comprobar autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la inicia el gestor de colas para comprobar si una entidad tiene autorización para realizar una acción determinada, o acciones, en un objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_CHECK_AUTHORITY.

Sintaxis

MQZ_CHECK_AUTHORITY(*QMgrName*, *EntityName*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName
Tipo: MQCHAR12 -entrada

Nombre de entidad. El nombre de la entidad cuya autorización para el objeto se va a comprobar. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

No es esencial que esta entidad sea conocida por el servicio de seguridad subyacente. Si no se conoce, las autorizaciones del grupo **nobody** especial (al que se supone que pertenecen todas las entidades) se utilizan para la comprobación. Un nombre totalmente en blanco es válido y se puede utilizar de esta forma.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por EntityName. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto al que se necesita acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

authority
Tipo: MQLONG - entrada

Autorización que se debe comprobar. Si se está comprobando una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si se está comprobando más de una autorización, es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

Las autorizaciones siguientes se aplican al uso de las llamadas MQI:

MQZAO_CONNECT
Posibilidad de utilizar la llamada MQCONN.

MQZAO_BROWSE
Posibilidad de utilizar la llamada MQGET con una opción de examinar.

Esto permite especificar la opción MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR o MQGMO_BROWSE_NEXT en la llamada MQGET.

MQZAO_INPUT
Principal. Posibilidad de utilizar la llamada MQGET con una opción de entrada.

Esto permite especificar la opción MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE o MQOO_INPUT_AS_Q_DEF en la llamada MQOPEN.

MQZAO_OUTPUT
Posibilidad de utilizar la llamada MQPUT.

Esto permite especificar la opción MQOO_OUTPUT en la llamada MQOPEN.

MQZAO_INQUIRE
Posibilidad de utilizar la llamada MQINQ.

Esto permite especificar la opción MQOO_INQUIRE en la llamada MQOPEN.

MQZAO_SET
Posibilidad de utilizar la llamada MQSET.

Esto permite especificar la opción MQOO_SET en la llamada MQOPEN.

MQZAO_PASS_IDENTITY_CONTEXT
Capacidad de pasar contexto de identidad.

Esto permite especificar la opción MQOO_PASS_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_PASS_ALL_CONTEXT
Capacidad de pasar todo el contexto.

Esto permite especificar la opción MQOO_PASS_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_ IDENTITY_ CONTEXT
Capacidad para establecer el contexto de identidad.

Esto permite especificar la opción MQOO_SET_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_ ALL_CONTEXT
Capacidad para establecer todo el contexto.

Esto permite especificar la opción MQOO_SET_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY
Capacidad para utilizar la autorización de usuario alternativo.

Esto permite especificar la opción MQOO_ALTERNATE_USER_AUTHORITY en la llamada MQOPEN y especificar la opción MQPMO_ALTERNATE_USER_AUTHORITY en la llamada MQPUT1 .

MQZAO_ALL_MQI
Todas las autorizaciones MQI.

Esto habilita todas las autorizaciones.

Las autorizaciones siguientes se aplican a la administración de un gestor de colas:

MQZAO_CREAR
Posibilidad de crear objetos de un tipo especificado.

MQZAO_DELETE
Posibilidad de suprimir un objeto especificado.

MQZAO_DISPLAY
Capacidad para visualizar los atributos de un objeto especificado.

MQZAO_CHANGE
Posibilidad de cambiar los atributos de un objeto especificado.

MQZAO_CLEAR
Posibilidad de suprimir todos los mensajes de una cola especificada.

MQZAO_AUTORIZAR
Posibilidad de autorizar a otros usuarios para un objeto especificado.

MQZAO_CONTROL
Capacidad para iniciar o detener un objeto de canal de escucha, servicio o no cliente, y la capacidad de hacer ping a un objeto de canal no cliente.

MQZAO_CONTROL_EXTENDED
Capacidad para restablecer un número de secuencia o resolver un mensaje dudoso en un objeto de canal no de cliente.

MQZAO_ALL_ADMIN
Capacidad para establecer el contexto de identidad.

Todas las autorizaciones de administración, excepto MQZAO_CREATE.

Las autorizaciones siguientes se aplican tanto al uso de la MQI como a la administración de un gestor de colas:

MQZAO_TODOS
Todas las autorizaciones, excepto MQZAO_CREATE.

MQZAO_NONE
Sin autorizaciones.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

Si la llamada a un componente falla (es decir, *CompCode* devuelve MQCC_FAILED), y el parámetro *Continuación* es MQZCI_DEFAULT o MQZCI_CONTINUE, el gestor de colas continúa llamando a otros componentes si los hay.

Si la llamada es satisfactoria (es decir, *CompCode* devuelve MQCC_OK) no se llama a ningún otro componente independientemente del valor de *Continuación* .

Si la llamada falla y el parámetro *Continuación* es MQZCI_STOP, no se llama a ningún otro componente y se devuelve el error al gestor de colas. Los componentes no conocen las llamadas anteriores, por lo que el parámetro *Continuación* siempre se establece en MQZCI_DEFAULT antes de la llamada.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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 -Comprobar autorización (ampliada)

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_2 y la inicia el gestor de colas para comprobar si una entidad tiene autorización para realizar una acción determinada, o acciones, en un objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_CHECK_AUTHORITY.

MQZ_CHECK_AUTHORITY_2 es como MQZ_CHECK_AUTHORITY, pero con el parámetro *EntityName* sustituido por el parámetro *EntityData* .

Sintaxis

MQZ_CHECK_AUTHORITY_2(*QMgrName*, *EntityData*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData
Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad con autorización sobre el objeto que se va a comprobar. Para obtener más detalles, consulte "MQZED-Descriptor de entidad" en la página 1211.

No es esencial que esta entidad sea conocida por el servicio de seguridad subyacente. Si no se conoce, las autorizaciones del grupo **nobody** especial (al que se supone que pertenecen todas las entidades) se utilizan para la comprobación. Un nombre totalmente en blanco es válido y se puede utilizar de esta forma.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityData*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto al que se necesita acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

authority
Tipo: MQLONG - entrada

Autorización que se debe comprobar. Si se está comprobando una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si se está comprobando más de una autorización, es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

Las autorizaciones siguientes se aplican al uso de las llamadas MQI:

MQZAO_CONNECT
Posibilidad de utilizar la llamada MQCONN.

MQZAO_BROWSE
Posibilidad de utilizar la llamada MQGET con una opción de examinar.

Esto permite especificar la opción MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR o MQGMO_BROWSE_NEXT en la llamada MQGET.

MQZAO_INPUT
Principal. Posibilidad de utilizar la llamada MQGET con una opción de entrada.

Esto permite especificar la opción MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE o MQOO_INPUT_AS_Q_DEF en la llamada MQOPEN.

MQZAO_OUTPUT
Posibilidad de utilizar la llamada MQPUT.

Esto permite especificar la opción MQOO_OUTPUT en la llamada MQOPEN.

MQZAO_INQUIRE
Posibilidad de utilizar la llamada MQINQ.

Esto permite especificar la opción MQOO_INQUIRE en la llamada MQOPEN.

MQZAO_SET
Posibilidad de utilizar la llamada MQSET.

Esto permite especificar la opción MQOO_SET en la llamada MQOPEN.

MQZAO_PASS_IDENTITY_CONTEXT
Capacidad de pasar contexto de identidad.

Esto permite especificar la opción MQOO_PASS_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_PASS_ALL_CONTEXT
Capacidad de pasar todo el contexto.

Esto permite especificar la opción MQOO_PASS_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_PASS_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_ IDENTITY_ CONTEXT
Capacidad para establecer el contexto de identidad.

Esto permite especificar la opción MQOO_SET_IDENTITY_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_IDENTITY_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_SET_ ALL_CONTEXT
Capacidad para establecer todo el contexto.

Esto permite especificar la opción MQOO_SET_ALL_CONTEXT en la llamada MQOPEN y especificar la opción MQPMO_SET_ALL_CONTEXT en las llamadas MQPUT y MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY
Capacidad para utilizar la autorización de usuario alternativo.

Esto permite especificar la opción MQOO_ALTERNATE_USER_AUTHORITY en la llamada MQOPEN y especificar la opción MQPMO_ALTERNATE_USER_AUTHORITY en la llamada MQPUT1 .

MQZAO_ALL_MQI
Todas las autorizaciones MQI.

Esto habilita todas las autorizaciones.

Las autorizaciones siguientes se aplican a la administración de un gestor de colas:

MQZAO_CREAR
Posibilidad de crear objetos de un tipo especificado.

MQZAO_DELETE
Posibilidad de suprimir un objeto especificado.

MQZAO_DISPLAY
Capacidad para visualizar los atributos de un objeto especificado.

MQZAO_CHANGE
Posibilidad de cambiar los atributos de un objeto especificado.

MQZAO_CLEAR
Posibilidad de suprimir todos los mensajes de una cola especificada.

MQZAO_AUTORIZAR
Posibilidad de autorizar a otros usuarios para un objeto especificado.

MQZAO_CONTROL
Capacidad para iniciar o detener un objeto de canal de escucha, servicio o no cliente, y la capacidad de hacer ping a un objeto de canal no cliente.

MQZAO_CONTROL_EXTENDED
Capacidad para restablecer un número de secuencia o resolver un mensaje dudoso en un objeto de canal no de cliente.

MQZAO_ALL_ADMIN
Capacidad para establecer el contexto de identidad.

Todas las autorizaciones de administración, excepto MQZAO_CREATE.

Las autorizaciones siguientes se aplican tanto al uso de la MQI como a la administración de un gestor de colas:

MQZAO_TODOS
Todas las autorizaciones, excepto MQZAO_CREATE.

MQZAO_NONE
Sin autorizaciones.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_CHECK_AUTHORITY_2 (QMgrName, &EntityData, EntityType,
                       ObjectName, ObjectType, Authority, ComponentData,
                       &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Comprobar si el usuario tiene privilegios

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_6 y la invoca el gestor de colas para determinar si un usuario especificado es un usuario privilegiado.

El identificador de función para esta función (para MQZEP) es MQZID_CHECK_PRIVILEGED.

Sintaxis

MQZ_CHECK_PRIVILEGED(*QMgrName*, *EntityData*, *EntityType*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData
Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad que se va a comprobar. Para obtener más información, consulte "MQZED-Descriptor de entidad" en la página 1211.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por EntityData. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

Si la llamada a un componente falla (es decir, *CompCode* devuelve MQCC_FAILED), y el parámetro *Continuación* es MQZCI_DEFAULT o MQZCI_CONTINUE, el gestor de colas continúa llamando a otros componentes si los hay.

Si la llamada es satisfactoria (es decir, *CompCode* devuelve MQCC_OK) no se llama a ningún otro componente independientemente del valor de *Continuación* .

Si la llamada falla y el parámetro *Continuación* es MQZCI_STOP, no se llama a ningún otro componente y se devuelve el error al gestor de colas. Los componentes no conocen las llamadas anteriores, por lo que el parámetro *Continuación* siempre se establece en MQZCI_DEFAULT antes de la llamada.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NO_PRIVILEGIADO
(2584, X'A18') Este usuario no es un ID de usuario privilegiado.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,
                      ComponentData, &Continuation,
                      &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Copiar todas las autorizaciones

Esta función la proporciona un componente de servicio de autorización. Lo inicia el gestor de colas para copiar todas las autorizaciones que están actualmente en vigor para un objeto de referencia en otro objeto.

El identificador de función para esta función (para MQZEP) es MQZID_COPY_ALL_AUTHORITY.

Sintaxis

MQZ_COPY_ALL_AUTHORITY(*QMgrName*, *RefObjectName*, *ObjectName*, *ObjectType*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

RefObjectNombre
Tipo: MQCHAR48 -entrada

Nombre de objeto de referencia. El nombre del objeto de referencia, cuyas autorizaciones se van a copiar. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se van a establecer los accesos. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *RefObjectName* y *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro de longitud ComponentDatade la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_REF_OBJECT
(2294, X'8F6') Objeto de referencia desconocido.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,
                        ComponentData, &Continuation, &CompCode,
                        &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Suprimir autorización

Esta función la proporciona un componente de servicio de autorización y la inicia el gestor de colas para suprimir todas las autorizaciones asociadas con el objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_DELETE_AUTHORITY.

Sintaxis

MQZ_DELETE_AUTHORITY(*QMgrName*, *ObjectName*, *ObjectType*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se van a suprimir los accesos. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro de longitud ComponentDatade la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,
                      &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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_AUTORITY_DATA-Enumerar datos de autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_4 y la inicia repetidamente el gestor de colas para recuperar todos los datos de autorización que coinciden con los criterios de selección especificados en la primera invocación.

El identificador de función para esta función (para MQZEP) es MQZID_ENUMERATE_AUTORITY_DATA.

Sintaxis

MQZ_ENUMERATE_AUTHORITY_DATA(*QMgrName*, *StartEnumeration*, *Filter*, *AuthorityBufferLength*, *AuthorityBuffer*, *AuthorityDataLength*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

StartEnumeration
Tipo: MQLONG - entrada

Distintivo que indica si la llamada puede iniciar la enumeración. Esto indica si la llamada puede iniciar la enumeración de datos de autorización o continuar la enumeración de datos de autorización iniciada por una llamada anterior a MQZ_ENUMERATE_AUTORITY_DATA. El valor es uno de los valores siguientes:

MQZSE_START
Iniciar enumeración. La llamada se inicia con este valor para iniciar la enumeración de datos de autorización. El parámetro *Filter* especifica los criterios de selección que se deben utilizar para seleccionar los datos de autorización devueltos por esta y por llamadas sucesivas.

MQZSE_CONTINUE
Continuar enumeración. La llamada se inicia con este valor para continuar la enumeración de datos de autorización. El parámetro *Filter* se ignora en este caso y se puede especificar como puntero nulo (los criterios de selección se determinan mediante el parámetro *Filter*especificado por la llamada que tenía *StartEnumeration* establecido en MQZSE_START).

Filter
Tipo: MQZAD-entrada

Filtro. Si *StartEnumeration* es MQZSE_START, *Filter* especifica los criterios de selección que se van a utilizar para seleccionar los datos de autorización que se van a devolver. Si *Filter* es el puntero nulo, no se utiliza ningún criterio de selección, es decir, se devuelven todos los datos de autorización. Consulte "MQZAD-Datos de autorización" en la página 1208 para obtener detalles de los criterios de selección que se pueden utilizar.

Si *StartEnumeration* es MQZSE_CONTINUE, *Filter* se ignora y se puede especificar como puntero nulo.

AuthorityBufferLongitud
Tipo: MQLONG - entrada

Longitud de *AuthorityBuffer*. Es la longitud en bytes del parámetro *AuthorityBuffer* . El almacenamiento intermedio de autorización debe ser lo suficientemente grande para dar cabida a los datos que se van a devolver.

AuthorityBuffer
Tipo: MQZAD-salida

Datos de autorización. Es el almacenamiento intermedio en el que se devuelven los datos de autorización. El almacenamiento intermedio debe ser lo suficientemente grande para acomodar una estructura MQZAD, una estructura MQZED, más el nombre de entidad más largo y el nombre de dominio más largo definido.

Nota: Nota: Este parámetro se define como un MQZAD, ya que el MQZAD siempre se produce al principio del almacenamiento intermedio. Sin embargo, si el almacenamiento intermedio se declara como MQZAD, el almacenamiento intermedio será demasiado pequeño; debe ser mayor que un MQZAD para que pueda acomodar los nombres de entidad y dominio MQZAD, MQZED, más.

AuthorityDataLongitud
Tipo: MQLONG - salida

Longitud de los datos devueltos en *AuthorityBuffer*. Si el almacenamiento intermedio de autorización es demasiado pequeño, *AuthorityDataLength* se establece en la longitud del almacenamiento intermedio necesaria y la llamada devuelve el código de terminación MQCC_FAILED y el código de razón MQRC_BUFFER_LENGTH_ERROR.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro de longitud ComponentDatade la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_ENUMERATE_AUTORITY_DATA, tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') El parámetro de longitud de almacenamiento intermedio no es válido.

MQRC_NO_DATA_AVAILABLE
(2379, X'94B') No hay datos disponibles.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,
                              AuthorityBufferLength,
                              &AuthorityBuffer,
                              &AuthorityDataLength, ComponentData,
                              &Continuation, &CompCode,
                              &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Usuario libre

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 y la inicia el gestor de colas para liberar el recurso asignado asociado.

Se inicia cuando una aplicación ha terminado de ejecutarse en todos los contextos de usuario, por ejemplo, durante una llamada MQI MQDISC.

El identificador de función para esta función (para MQZEP) es MQZID_FREE_USER.

Sintaxis

MQZ_FREE_USER(*QMgrName*, *FreeParms*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

FreeParms

Tipo: MQZFP-entrada

Parámetros libres. Estructura que contiene datos relacionados con el recurso que se va a liberar. Consulte "MQZFP-Parámetros libres" en la página 1213 para obtener los detalles.

ComponentData

Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro de longitud ComponentDatade la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Distintivo de continuación. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente de otros componentes.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón

Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                   IdentityContext, CorrelationPtr, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Obtener autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la inicia el gestor de colas para recuperar la autorización que una entidad tiene para acceder al objeto especificado, incluidas (si la entidad es un principal) las autorizaciones que poseen los grupos de los que el principal es miembro. Las autorizaciones de perfiles genéricos se incluyen en el conjunto de autorizaciones devuelto.

El identificador de función para esta función (para MQZEP) es MQZID_GET_AUTHORITY.

Sintaxis

MQZ_GET_AUTHORITY(*QMgrName*, *EntityName*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName
Tipo: MQCHAR12 -entrada

Nombre de entidad. El nombre de la entidad cuyo acceso al objeto se va a recuperar. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityName*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto al que se va a recuperar el acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

authority
Tipo: MQLONG - entrada

Autoridad de entidad. Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_GET_AUTHORITY, tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, &Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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 -Obtener autorización (ampliada)

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_2 y la inicia el gestor de colas para recuperar la autorización que una entidad tiene para acceder al objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_GET_AUTHORITY.

MQZ_GET_AUTHORITY_2 es como MQZ_GET_AUTHORITY, pero con el parámetro *EntityName* sustituido por el parámetro *EntityData* .

Sintaxis

MQZ_GET_AUTHORITY_2(*QMgrName*, *EntityData*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData
Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad para la que se va a recuperar la autorización para el objeto. Para obtener más detalles, consulte "MQZED-Descriptor de entidad" en la página 1211.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityData*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se va a recuperar la autorización de entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

authority
Tipo: MQLONG - entrada

Autoridad de entidad. Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Sintaxis

MQZ_GET_AUTHORITY_2*(QMgrName, EntityData, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

Invocación en C

```
MQZ_GET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                     ObjectType, &Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Obtener autorización explícita

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la inicia el gestor de colas para recuperar la autorización que un grupo con nombre tiene para acceder a un objeto especificado (pero sin la autorización adicional del grupo **nobody**), o la autorización que el grupo primario del principal con nombre tiene para acceder a un objeto especificado.

En plataformas UNIX, para el gestor de autorizaciones sobre objetos (OAM) de WebSphere MQ incorporado, la autorización devuelta es la que sólo posee el grupo primario del principal.

El identificador de función para esta función (para MQZEP) es MQZID_GET_EXPLICIT_AUTHORITY.

Sintaxis

MQZ_GET_EXPLICIT_AUTHORITY(*QMgrName*, *EntityName*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName
Tipo: MQCHAR12 -entrada

Nombre de entidad. El nombre de la entidad para la que se va a recuperar el acceso al objeto. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityName*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se va a recuperar la autorización de entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

authority
Tipo: MQLONG - entrada

Autoridad de entidad. Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_GET_AUTHORITY, tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,
                            ObjectName, ObjectType, &Authority,
                            ComponentData, &Continuation,
                            &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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 -Obtener autorización explícita (ampliada)

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_2 y la inicia el gestor de colas para recuperar la autorización que un grupo con nombre tiene para acceder a un objeto especificado (pero sin la autorización adicional del grupo **nobody**), o la autorización que tiene el grupo primario del principal con nombre para acceder a un objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_GET_EXPLICIT_AUTHORITY.

MQZ_GET_EXPLICIT_AUTHORITY_2 es como MQZ_GET_EXPLICIT_AUTHORITY, pero con el parámetro *EntityName* sustituido por el parámetro *EntityData* .

Sintaxis

MQZ_GET_EXPLICIT_AUTHORITY_2(*QMgrName*, *EntityData*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData
Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad cuya autorización al objeto se va a recuperar. Para obtener más detalles, consulte “MQZED-Descriptor de entidad” en la página 1211.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityData*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto para el que se va a recuperar la autorización de entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

authority
Tipo: MQLONG - entrada

Autoridad de entidad. Si la entidad tiene una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si tiene más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_GET_EXPLICIT_AUTHORITY_2 (QMgrName, &EntityData, EntityType,
                              ObjectName, ObjectType, &Authority,
                              ComponentData, &Continuation,
                              &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Inicializar servicio de autorización

Esta función la proporciona un componente de servicio de autorización y la inicia el gestor de colas durante la configuración del componente. Se espera que llame a MQZEP para proporcionar información al gestor de colas.

El identificador de función para esta función (para MQZEP) es MQZID_INIT_AUTHORITY.

Sintaxis

```
MQZ_INIT_AUTHORITY( Hconfig, Options, QMgrName, ComponentDataLength,
ComponentData, Version, CompCode, Reason)
```

Parámetros

Hconfig
Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente concreto que se está inicializando. Debe ser utilizado por el componente al llamar al gestor de colas con la función MQZEP.

Opciones
Tipo: MQLONG - entrada

Opciones de inicialización. Tiene que ser uno de los valores siguientes:

MQZIO_PRIMARY
Inicialización primaria.

MQZIO_SECUNDARIO
Inicialización secundaria.

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentDataLongitud
Tipo: MQLONG - entrada

Longitud de los datos de componente. Longitud en bytes del área *ComponentData* . Esta longitud se define en los datos de configuración del componente.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. Se inicializa con todos los ceros antes de llamar a la función de inicialización primaria del componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

version
Tipo: MQLONG-entrada/salida

Número de versión. En la entrada de la función de inicialización, identifica el número de versión más alto al que da soporte el gestor de colas. La función de inicialización debe cambiar esto, si es necesario, a la versión de la interfaz a la que da soporte. Si en la devolución el gestor de colas

no soporta la versión devuelta por el componente, llama a la función MQZ_TERM_AUTHORITY del componente y no hace más uso de este componente.

Se admiten los valores siguientes:

MQZAS_VERSION_1
Versión 1.

MQZAS_VERSION_2
Versión 2.

MQZAS_VERSION_3
Versión 3.

MQZAS_VERSION_4
Versión 4.

MQZAS_VERSION_5
Versión 5.

MQZAS_VERSION_6
Versión 6.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_INITIALIZATION_FAILED
(2286, X'8EE') La inicialización ha fallado por una razón no definida.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,
                    ComponentData, &Version, &CompCode,
                    &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Consultar servicio de autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_5 y la inicia el gestor de colas para consultar la funcionalidad soportada.

Cuando se utilizan varios componentes de servicio, se llama a los componentes de servicio en orden inverso al orden en el que se instalaron.

El identificador de función para esta función (para MQZEP) es MQZID_INQUIRE.

Sintaxis

MQZ_INQUIRE(*QMgrName*, *SelectorCount* , *Selectors* , *IntAttrCount* , *IntAttrs* , *CharAttrLength* , *CharAttrs* , *SelectorReturned* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

SelectorCount

Tipo: MQLONG - entrada

Número de selectores. Número de selectores proporcionados en el parámetro *Selectors* .

El valor debe estar en el rango de 0 a 256.

Selectores

Tipo: MQLONGxSelectorRecuento-entrada

Matriz de selectores. Cada selector identifica un atributo necesario y debe ser uno de los siguientes:

- MQIACF_INTERFACE_VERSION (entero)
- MQIACF_USER_ID_SUPPORT (entero)
- MQCACF_SERVICE_COMPONENT (carácter)

Los selectores se pueden especificar en cualquier orden. El número de selectores de la matriz se indica mediante el parámetro *SelectorCount* .

Los atributos enteros identificados por los selectores se devuelven en el parámetro *IntAttrs* en el mismo orden en que aparecen en *Selectors* .

Los atributos de carácter identificados por los selectores se devuelven en el parámetro *CharAttrs* en el mismo orden en que aparecen *Selectors* .

IntAttrCount

Tipo: MQLONG - entrada

Número de atributos enteros proporcionados en el parámetro IntAttrs .

El valor debe estar en el rango de 0 a 256.

IntAttrs

Tipo: MQLONG ×IntAttrRecuento-salida

Atributos enteros. Matriz de atributos enteros. Los atributos de entero se devuelven en el mismo orden que los selectores de entero correspondientes en la matriz *Selectors* .

Recuento deCharAttr
Tipo: MQLONG - entrada

Longitud del almacenamiento intermedio de atributos de caracteres. Longitud en bytes del parámetro *CharAttrs* .

El valor debe ser como mínimo la suma de las longitudes de los atributos de caracteres solicitados. Si no se solicita ningún atributo de carácter, cero es un valor válido.

CharAttrs
Tipo: MQLONG ×CharAttrRecuento-salida

Almacenamiento intermedio de atributos de caracteres. Almacenamiento intermedio que contiene atributos de caracteres, concatenados entre sí. Los atributos de carácter se devuelven en el mismo orden que los selectores de caracteres correspondientes en la matriz *Selectors* .

La longitud del almacenamiento intermedio la proporciona el parámetro de recuento CharAttr.

SelectorReturned
Tipo: MQLONG ×SelectorCount -entrada

Se ha devuelto el selector. Matriz de valores que identifican qué atributos se han devuelto del conjunto solicitado por los selectores en el parámetro Selectores. El número de valores de esta matriz se indica mediante el parámetro *SelectorCount* . Cada valor en la matriz se relaciona con el selector desde la posición correspondiente en la matriz de selectores. Cada valor es uno de los siguientes:

MQZSL_DEVUELTO
Se ha devuelto el atributo solicitado por el selector correspondiente en el parámetro *Selectors* .

MQZSL_NO_DEVUELTO
El atributo solicitado por el selector correspondiente en el parámetro *Selectors* no se ha devuelto.

La matriz se inicializa con todos los valores como *MQZSL_NOT_RETURNED*. Cuando un componente de servicio de autorización devuelve un atributo, establece el valor adecuado en la matriz en *MQZSL_NOT_RETURNED* . Esto permite que cualquier otro componente de servicio de autorización, al que se realiza la llamada de consulta, identifique qué atributos ya se han devuelto.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Finalización parcial.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode* .

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT
No hay espacio suficiente para los atributos de carácter.

MQRC_INT_COUNT_TOO_SMALL
No hay suficiente espacio para atributos enteros.

Si *CompCode* es MQCC_FAILED:

MQRC_SELECTOR_COUNT_ERROR
El número de selectores no es válido.

MQRC_SELECTOR_ERROR
Selector de atributo no válido.

MQRC_SELECTOR_LIMIT_EXCEEDED
Se han especificado demasiados selectores.

MQRC_INT_ATTR_COUNT_ERROR
El número de atributos enteros no es válido.

MQRC_INT_ATTRS_ARRAY_ERROR
Matriz de atributos enteros no válida.

MQRC_CHAR_ATTR_LENGTH_ERROR
El número de atributos de tipo carácter no es válido.

MQRC_CHAR_ATTRS_ERROR
La serie de atributos de tipo carácter no es válida.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
                  &IntAttrs, CharAttrLength, &CharAttrs,
                  SelectorReturned, ComponentData, &Continuation,
                  &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Renovar todas las autorizaciones

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_3 y la invoca el gestor de colas para renovar la lista de autorizaciones mantenidas internamente por el componente.

El identificador de función para esta función (para MQZEP) es MQZID_REFRESH_CACHE (8L).

Sintaxis

MQZ_REFRESH_CACHE(*QMgrName*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName

Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentData

Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de las funciones de este componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

continuación

Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode

Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

Invocación en C

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
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-Establecer autorización

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_1 y la inicia el gestor de colas para establecer la autorización que una entidad tiene para acceder al objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_SET_AUTHORITY.

Nota: Esta función altera temporalmente las autorizaciones existentes. Para conservar las autorizaciones existentes, debe volver a establecerlas con esta función.

Sintaxis

MQZ_SET_AUTHORITY(*QMgrName*, *EntityName*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityName
Tipo: MQCHAR12 -entrada

Nombre de entidad. El nombre de la entidad para la que se va a recuperar el acceso al objeto. La longitud máxima de la serie es de 12 caracteres; si es más corta que esa, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityName*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto al que se necesita acceso. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

authority
Tipo: MQLONG - entrada

Autoridad de entidad. Si se establece una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si se establece más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_GET_AUTHORITY, tiene el mismo efecto que MQZCI_CONTINUE.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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 -Establecer autorización (ampliado)

Esta función la proporciona un componente de servicio de autorización MQZAS_VERSION_2 y la inicia el gestor de colas para establecer la autorización que una entidad tiene para acceder al objeto especificado.

El identificador de función para esta función (para MQZEP) es MQZID_SET_AUTHORITY.

Nota: Esta función altera temporalmente las autorizaciones existentes. Para conservar las autorizaciones existentes, debe volver a establecerlas con esta función.

MQZ_SET_AUTHORITY_2 es como MQZ_SET_AUTHORITY, pero con el parámetro *EntityName* sustituido por el parámetro *EntityData* .

Sintaxis

MQZ_SET_AUTHORITY_2(*QMgrName*, *EntityData*, *EntityType*, *ObjectName*, *ObjectType*, *Authority*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

EntityData
Tipo: MQZED-entrada

Datos de entidad. Datos relacionados con la entidad cuya autorización para el objeto se va a establecer. Para obtener más detalles, consulte “MQZED-Descriptor de entidad” en la página 1211.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad. El tipo de entidad especificado por *EntityData*. Tiene que ser uno de los valores siguientes:

MQZAET_PRINCIPAL
Principal.

GRUPO_MQZAC
group.

ObjectName
Tipo: MQCHAR48 -entrada

El nombre del objeto. El nombre del objeto sobre el que se va a establecer la autorización de entidad. La longitud máxima de la serie es de 48 caracteres; si es más corta, se rellena a la derecha con espacios en blanco. El nombre no termina con un carácter nulo.

Si *ObjectType* es MQOT_Q_MGR, este nombre es el mismo que *QMgrName*.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto. El tipo de entidad especificado por *ObjectName*. Tiene que ser uno de los valores siguientes:

MQOT_AUTH_INFO
Información de autenticación.

MQOT_CHANNEL
Canal.

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente.

MQOT_ESCUCHA
Escucha.

MQOT_NAMELIST
Lista de nombres.

MQOT_PROCESS
.

MQOT_Q
Cola.

MQOT_Q_MGR
Gestor de colas.

SERVICIO MQOT_SERVICE
.

MQOT_TOPIC
.

authority
Tipo: MQLONG - entrada

Autoridad de entidad. Si se establece una autorización, este campo es igual a la operación de autorización adecuada (constante MQZAO_ *). Si se establece más de una autorización, este campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Se pueden especificar los siguientes valores:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

Para MQZ_CHECK_AUTHORITY, tiene el mismo efecto que MQZCI_STOP.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') No autorizado para el acceso.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entidad desconocida para el servicio.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_SET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Terminar servicio de autorización

Esta función la proporciona un componente de servicio de autorización y la inicia el gestor de colas cuando ya no necesita los servicios de este componente. La función debe realizar cualquier limpieza necesaria para el componente.

El identificador de función para esta función (para MQZEP) es MQZID_TERM_AUTHORITY.

Sintaxis

MQZ_TERM_AUTHORITY(*Hconfig*, *Options*, *QMgrName*, *ComponentData*, *CompCode*, *Reason*)

Parámetros

Hconfig
Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente concreto que se está terminando. Debe ser utilizado por el componente al llamar al gestor de colas con la función MQZEP.

Opciones
Tipo: MQLONG - entrada

Opciones de terminación. Tiene que ser uno de los valores siguientes:

MQZTO_PRIMARY
Terminación primaria.

MQZTO_SECUNDARIO
Terminación secundaria.

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro de longitud ComponentDataen la llamada MQZ_INIT_AUTHORITY.

Cuando se ha completado la llamada MQZ_TERM_AUTHORITY, el gestor de colas descarta estos datos.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_TERMINATION_FAILED
(2287, X'8FF') La terminación ha fallado por una razón no definida.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,
                    &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Suprimir nombre

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas para suprimir una entrada para la cola especificada.

El identificador de función para esta función (para MQZEP) es MQZID_DELETE_NAME.

Sintaxis

MQZ_DELETE_NAME(*QMgrName*, *QName*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

nombre_cola
Tipo: MQCHAR48 -entrada

Nombre de cola. El nombre de la cola para la que se va a suprimir una entrada. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro de longitud ComponentDataen la llamada MQZ_INIT_NAME.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Tiene que ser uno de los valores siguientes:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

MQZCI_STOP
No continúe con el componente siguiente.

Para el mandato **MQZ_DELETE_NAME** , el gestor de colas no intenta iniciar otro componente, independientemente de lo que se devuelva en el parámetro **Continuation** .

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_WARNING
Aviso (finalización parcial).

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_WARNING:

MQRC_UNKNOWN_NAME
(2288, X'8F0') No se ha encontrado el nombre de cola.

Nota: Es posible que no sea posible devolver este código si el servicio subyacente responde correctamente en este caso.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_DELETE_NAME (QMgrName, QName, ComponentData, &Continuation,
                 &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Inicializar servicio de nombres

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas durante la configuración del componente. Se espera que llame a MQZEP para proporcionar información al gestor de colas.

El identificador de función para esta función (para MQZEP) es MQZID_INIT_NAME.

Sintaxis

MQZ_INIT_NAME(*Hconfig*, *Options*, *QMgrName*, *ComponentDataLength*, *ComponentData*, *Version*, *CompCode*, *Reason*)

Parámetros

Hconfig
Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente concreto que se está inicializando. Debe ser utilizado por el componente al llamar al gestor de colas con la función MQZEP.

Opciones
Tipo: MQLONG - entrada

Opciones de inicialización. Tiene que ser uno de los valores siguientes:

MQZIO_PRIMARY
Inicialización primaria.

MQZIO_SECUNDARIO
Inicialización secundaria.

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentDataLongitud
Tipo: MQLONG - entrada

Longitud de los datos de componente. Longitud en bytes del área *ComponentData* . Esta longitud se define en los datos de configuración del componente.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. Se inicializa con todos los ceros antes de llamar a la función de inicialización primaria del componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_AUTHORITY.

version
Tipo: MQLONG-entrada/salida

Número de versión. En la entrada de la función de inicialización, identifica el número de versión más alto al que da soporte el gestor de colas. La función de inicialización debe cambiar esto, si es necesario, a la versión de la interfaz a la que da soporte. Si en el momento de la devolución el gestor de colas no soporta la versión devuelta por el componente, llama a la función MQZ_TERM_NAME del componente y no hace más uso de este componente.

Se admiten los valores siguientes:

MQZAS_VERSION_1
Versión 1.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_INITIALIZATION_FAILED
(2286, X'8EE') La inicialización ha fallado por una razón no definida.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_INIT_NAME (Hconfig, Options, QMgrName, ComponentDataLength,
               ComponentData, &Version, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Insertar nombre

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas para insertar una entrada para la cola especificada, que contiene el nombre del gestor de colas propietario de la cola. Si la cola ya está definida en el servicio, la llamada falla.

El identificador de función para esta función (para MQZEP) es MQZID_INSERT_NAME.

Sintaxis

MQZ_INSERT_NAME(*QMgrName*, *QName*, *ResolvedQMgrName*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

nombre_cola
Tipo: MQCHAR48 -entrada

Nombre de cola. El nombre de la cola para la que se va a insertar una entrada. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

ResolvedQMgrName
Tipo: MQCHAR48 -entrada

Nombre de gestor de colas resuelto. El nombre del gestor de colas en el que se resuelve la cola. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_NAME.

continuación
Tipo: MQLONG-entrada/salida

Indicador de continuación establecido por componente. Para MQZ_INSERT_NAME, el gestor de colas no intenta iniciar otro componente, lo que se devuelva en el parámetro *Continuation* .

Se admiten los valores siguientes:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_Q_ALREADY_EXISTS
(2290, X'8F2') El objeto de cola ya existe.

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_INSERT_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,
                 &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Nombre de búsqueda

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas para recuperar el nombre del gestor de colas propietario, para una cola especificada.

El identificador de función para esta función (para MQZEP) es MQZID_LOOKUP_NAME.

Sintaxis

MQZ_LOOKUP_NAME(*QMgrName*, *QName*, *ResolvedQMgrName*, *ComponentData*, *Continuation*, *CompCode*, *Reason*)

Parámetros

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

nombre_cola
Tipo: MQCHAR48 -entrada

Nombre de cola. El nombre de la cola para la que se va a resolver una entrada. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

ResolvedQMgrName
Tipo: MQCHAR48 -salida

Nombre de gestor de colas resuelto. Si la función se completa correctamente, este es el nombre del gestor de colas propietario de la cola.

El nombre devuelto por el componente de servicio debe rellenarse a la derecha con espacios en blanco hasta la longitud completa del parámetro; el nombre no debe terminar con un carácter nulo, ni contener espacios en blanco iniciales o intercalados.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_NAME.

continuación
Tipo: MQLONG - salida

Indicador de continuación establecido por componente. Para MQZ_LOOKUP_NAME, el gestor de colas especifica si se debe iniciar otro componente de servicio de nombres, como se indica a continuación:

- Si *CompCode* es MQCC_OK, no se inician más componentes, independientemente del valor que se devuelva en *Continuación*.
- Si *CompCode* no es MQCC_OK, se inicia un componente adicional, a menos que *Continuation* sea MQZCI_STOP.

Se admiten los valores siguientes:

MQZCI_DEFAULT
Continuación dependiente del gestor de colas.

MQZCI_CONTINUE
Continúe con el componente siguiente.

MQZCI_STOP
No continúe con el componente siguiente.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Se ha producido un error inesperado al acceder al servicio.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

MQRC_UNKNOWN_Q_NAME
(2288, X'8F0') No se ha encontrado el nombre de cola.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_LOOKUP_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,
                 &Continuation, &CompCode, &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Terminar servicio de nombres

Esta función la proporciona un componente de servicio de nombres y la inicia el gestor de colas cuando ya no necesita los servicios de este componente. La función debe realizar cualquier limpieza necesaria para el componente.

El identificador de función para esta función (para MQZEP) es MQZID_TERM_NAME.

Sintaxis

MQZ_TERM_NAME(*Hconfig*, *Options*, *QMgrName*, *ComponentData*, *CompCode*, *Reason*)

Parámetros

Hconfig
Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente concreto que se está terminando. Lo utiliza el componente al llamar al gestor de colas con la función MQZEP.

Opciones
Tipo: MQLONG - entrada

Opciones de terminación. Tiene que ser uno de los valores siguientes:

MQZTO_PRIMARY
Terminación primaria.

MQZTO_SECUNDARIO
Terminación secundaria.

QMgrName
Tipo: MQCHAR48 -entrada

Nombre del gestor de colas. El nombre del gestor de colas que llama al componente. Este nombre se rellena con espacios en blanco hasta la longitud completa del parámetro; el nombre no termina con un carácter nulo.

El nombre del gestor de colas se pasa al componente para obtener información; la interfaz del servicio de autorización no requiere que el componente lo utilice de ninguna manera definida.

ComponentData
Tipo: MQBYTE ×ComponentDataLongitud-entrada/salida

Datos de componente. El gestor de colas conserva estos datos en nombre de este componente en particular; los cambios realizados en él por cualquiera de las funciones (incluida la función de inicialización) proporcionadas por este componente se conservan y se presentan la próxima vez que se llame a una de estas funciones de componente.

Los datos de componente están en memoria compartida accesible para todos los procesos.

La longitud de esta área de datos la pasa el gestor de colas en el parámetro *ComponentDataLength* de la llamada MQZ_INIT_NAME.

Cuando se ha completado la llamada MQZ_TERM_NAME, el gestor de colas descarta estos datos.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode*.

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_TERMINATION_FAILED
(2287, X'8FF') La terminación ha fallado por una razón no definida.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') El servicio subyacente no está disponible.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZ_TERM_NAME (Hconfig, Options, QMgrName, ComponentData, &CompCode,
               &Reason);
```

Los parámetros pasados al servicio se declaran de la siguiente manera:

```
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-Contexto de aplicación

La estructura MQZAC se utiliza en la llamada MQZ_AUTHENTICATE_USER para el parámetro *ApplicationContext* . Este parámetro especifica los datos relacionados con la aplicación de llamada.

Tabla 1 resume los campos de la estructura.

Tabla 596. Campos en MQZAC	
Campo	**Descripción**
StrucId	Identificador de la estructura
Versión	Número de versión de la estructura
ProcessId	Identificador proceso
ThreadId	Identificador de hebra
ApplName	Nombre de la aplicación
UserID	Identificador de usuario
ID deEffectiveUser	Identificador de usuario efectivo
Entorno	Entorno
CallerType	Tipo de interlocutor
AuthenticationType	Tipo de autenticación
BindType	tipo de enlace

Campos

StrucId
Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQZAC_STRUC_ID
Identificador de la estructura de contexto de aplicación.

Para el lenguaje de programación C, también se define la constante MQZAC_STRUC_ID_ARRAY; tiene el mismo valor que MQZAC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

version
Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZAC_VERSION_1
Estructura de contexto de aplicación Version-1 . La constante MQZAC_CURRENT_VERSION especifica el número de versión de la versión actual.

ProcessId
Tipo: MQPID-entrada

Identificador de proceso de la aplicación.

ThreadId
Tipo: MQTID-entrada

Identificador de hebra de la aplicación.

ApplName
Tipo: MQCHAR28 -entrada

Nombre de aplicación.

UserID
Tipo: MQCHAR12 -entrada

Identificador de usuario. En sistemas UNIX , este campo especifica el ID de usuario real de la aplicación. En Windows este campo especifica el ID de usuario de la aplicación.

ID deEffectiveUser
Tipo: MQCHAR12 -entrada

Identificador de usuario efectivo. En sistemas UNIX , este campo especifica el ID de usuario efectivo de la aplicación. En Windows , este campo está en blanco.

entorno
Tipo: MQLONG - entrada

Medio ambiente. Este campo especifica el entorno desde el que se ha realizado la llamada. El campo es uno de los valores siguientes:

MQXE_COMMAND_SERVER
Servidor de mandatos

MQXE_MQSC
Intérprete de mandatos **runmqsc**

MQXE_MCA
Agente de canal de mensajes MQXE_OTHER

MQXE_OTHER
Entorno no definido

CallerType
Tipo: MQLONG - entrada

Tipo de interlocutor. Este campo especifica el tipo de programa que ha realizado la llamada. El campo es uno de los valores siguientes:

MQXACT_EXTERNAL
La llamada es externa al gestor de colas.

MQXACT_INTERNAL
La llamada es interna al gestor de colas.

AuthenticationType
Tipo: MQLONG - entrada

Tipo de autenticación. Este campo especifica el tipo de autenticación que se está realizando. El campo es uno de los valores siguientes:

MQZAT_INITIAL_CONTEXT
La llamada de autenticación se debe a que se está inicializando el contexto de usuario. Este valor se utiliza durante una llamada MQCONN o MQCONNX.

CONTEXTO_CAMBIO_MQZAT_CONTEXTO
La llamada de autenticación se debe a que se ha cambiado el contexto de usuario. Este valor se utiliza cuando el MCA cambia el contexto de usuario. Tema principal: MQZAC-

BindType
Tipo: MQLONG - entrada

Tipo de enlace. Este campo especifica el tipo de enlace en uso. El campo es uno de los valores siguientes:

MQCNO_FASTPATH_BINDING
Enlace de vía de acceso rápida.

MQCNO_SHARED_BINDING
Enlace compartido.

MQCNO_ISOLATED_BINDING
Enlace aislado.

Declaración C

Declare los campos de la estructura como se indica a continuación:

```
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-Datos de autorización

La estructura MQZAD se utiliza en la llamada MQZ_ENUMERATE_AUTORITY_DATA para dos parámetros, uno de entrada y otro de salida.

- MQZAD se utiliza para el parámetro *Filter* que se especifica en la llamada. Este parámetro especifica los criterios de selección que deben utilizarse para seleccionar los datos de autorización devueltos por la llamada.
- MQZAD también se utiliza para el parámetro *AuthorityBuffer* que es la salida de la llamada. Este parámetro especifica las autorizaciones para una combinación de nombre de perfil, tipo de objeto y entidad.

Tabla 1. resume los campos de la estructura.

Tabla 597. Campos en MQZAD

Campo	Descripción
StrucId	Identificador de la estructura
Versión	Número de versión de la estructura
ProfileName	Identificador proceso
ObjectType	Identificador de hebra
Autorización	Nombre de la aplicación
EntityDataPtr	Identificador de usuario
EntityType	Entorno
Opciones	Tipo de interlocutor

Campos

StrucId

Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQZAC_STRUC_ID

Identificador de la estructura de contexto de aplicación.

Para el lenguaje de programación C, también se define la constante MQZAC_STRUC_ID_ARRAY; tiene el mismo valor que MQZAC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

version
Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZAC_VERSION_1
Estructura de contexto de aplicación Version-1 . La constante MQZAC_CURRENT_VERSION especifica el número de versión de la versión actual.

La constante siguiente especifica el número de versión de la versión actual:

MQZAD_CURRENT_VERSION
Versión actual de la estructura de datos de autorización.

ProfileName
Tipo: MQCHAR48 -entrada

Nombre de perfil.

Para el parámetro *Filtro* , este campo es el nombre de perfil para el que se necesitan datos de autorización. Si el nombre está completamente en blanco hasta el final del campo o el primer carácter nulo, se devuelven los datos de autorización para todos los nombres de perfil.

Para el parámetro *AuthorityBuffer* , este campo es el nombre de un perfil que coincide con los criterios de selección especificados.

ObjectType
Tipo: MQLONG - entrada

Tipo de objeto.

Para el parámetro *Filtro* , este campo es el tipo de objeto para el que se necesitan datos de autorización. Si el valor es MQOT_ALL, se devuelven los datos de autorización para todos los tipos de objeto.

Para el parámetro *AuthorityBuffer* , este campo es el tipo de objeto al que se aplica el perfil identificado por el parámetro *ProfileName* .

El valor es uno de los siguientes; para el parámetro *Filtro* , el valor MQOT_ALL también es válido:

MQOT_AUTH_INFO
Información de autenticación

MQOT_CHANNEL
Canal

MQOT_CLNTCONN_CHANNEL
Canal de conexión de cliente

MQOT_ESCUCHA
Escucha

MQOT_NAMELIST
Lista de nombres

MQOT_PROCESS
Definición de proceso

MQOT_Q
Cola

MQOT_Q_MGR
Gestor de colas

SERVICIO MQOT_SERVICE
Servicio

Autorización
Tipo: MQLONG - entrada

Autorización.

Para el parámetro *Filtro* , este campo se ignora.

Para el parámetro *AuthorityBuffer* , este campo representa las autorizaciones que la entidad tiene sobre los objetos identificados por *ProfileName* y *ObjectType*. Si la entidad sólo tiene una autorización, el campo es igual al valor de autorización adecuado (constante MQZAO_ *). Si la entidad tiene más de una autorización, el campo es el OR a nivel de bit de las constantes MQZAO_ * correspondientes.

EntityDataPtr
Tipo: PMQZED-entrada

Dirección de la estructura MQZED que identifica una entidad.

Para el parámetro *Filtro* , este campo apunta a una estructura MQZED que identifica la entidad para la que se necesitan datos de autorización. Si *EntityDataPtr* es el puntero nulo, se devuelven los datos de autorización para todas las entidades.

Para el parámetro *AuthorityBuffer* , este campo apunta a una estructura MQZED que identifica la entidad para la que se han devuelto datos de autorización.

EntityType
Tipo: MQLONG - entrada

Tipo de entidad.

Para el parámetro *Filtro* , este campo especifica el tipo de entidad para el que se necesitan datos de autorización. Si el valor es MQZAET_NONE, se devuelven los datos de autorización para todos los tipos de entidad.

Para el parámetro *AuthorityBuffer* , este campo especifica el tipo de la entidad identificada por la estructura MQZED a la que apunta el parámetro *EntityDataPtr* .

El valor es uno de los siguientes; para el parámetro *Filtro* , el valor MQZAET_NONE también es válido:

MQZAET_PRINCIPAL
Entidad de seguridad

GRUPO_MQZAC
Grupo

Opciones
Tipo: MQAUTHOPT-entrada

Opciones. Este campo especifica las opciones que dan control sobre los perfiles que se visualizan. Se debe especificar uno de los valores siguientes:

MQAUTHOPT_NAME_ALL_MATCHING
Muestra todos los perfiles

MQAUTHOPT_NAME_EXPLICIT
Muestra perfiles que tienen exactamente el mismo nombre que el especificado en el campo *ProfileName* .

Además, también debe especificarse uno de los siguientes:

MQAUTHOPT_ENTITY_SET
Visualizar todos los perfiles que se utilizan para calcular la autorización acumulativa que la entidad tiene sobre el objeto especificado por el parámetro *ProfileName* . El parámetro *ProfileName* no debe contener caracteres comodín.

Si la entidad especificada es un principal, para cada miembro del conjunto {entity, groups} se muestra el perfil más aplicable que se aplica al objeto.

Si la entidad especificada es un grupo, se visualiza el perfil más aplicable del grupo que se aplica al objeto.

Si se especifica este valor, los valores de *ProfileName*, *ObjectType*, *EntityType*y el nombre de entidad que se especifica en la estructura MQZED *EntityDataPtr* , no deben estar en blanco.

Si ha especificado MQAUTHOPT_NAME_ALL_MATCHING, también puede especificar el valor siguiente:

MQAUTHOPT_ENTITY_EXPLICIT
Muestra perfiles que tienen exactamente el mismo nombre de entidad que el nombre de entidad especificado en la estructura MQZED *EntityDataPtr* .

Declaración 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 */
};
```

Campos

MQZED-Descriptor de entidad

La estructura MQZED se utiliza en un número de llamadas de servicio de autorización para especificar la entidad para la que se va a comprobar la autorización.

Tabla 1. resume los campos de la estructura.

Tabla 598. Campos en MQZED

Campo	Descripción
StrucId	Identificador de la estructura
Versión	Versión
EntityName Ptr	Nombre de entidad
EntityDomainPtr	Puntero de dominio de entidad
SecurityId	Identificador de seguridad
CorrelationPtr	Puntero de correlación

Campos

StrucId
Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQZED_STRUC_ID
Identificador de la estructura del descriptor de entidad.

Para el lenguaje de programación C, también se define la constante MQZED_STRUC_ID_ARRAY; tiene el mismo valor que MQZED_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

version
Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZED_VERSION_1
Estructura del descriptor de entidad Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQZED_CURRENT_VERSION
Versión actual de la estructura del descriptor de entidad.

EntityNamePtr
Tipo: PMQCHAR-entrada

Nombre de perfil.

Dirección del nombre de entidad. Se trata de un puntero al nombre de la entidad cuya autorización se va a comprobar.

EntityDomainPtr
Tipo: PMQCHAR-entrada

Dirección del nombre de dominio de entidad. Es un puntero al nombre del dominio que contiene la definición de la entidad cuya autorización se va a comprobar.

SecurityId
Tipo: MQBYTE40 -entrada

Autorización.

Identificador de seguridad. Es el identificador de seguridad cuya autorización se debe comprobar.

CorrelationPtr
Tipo: MQPTR-entrada

Puntero de correlación. Esto facilita el paso de datos correlacionales entre la función de autenticar usuario y otras funciones de OAM adecuadas.

Declaración 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 */
```

Campos

MQZEP-Añadir punto de entrada de componente

Un componente de servicio inicia esta función, durante la inicialización, para añadir un punto de entrada al vector de punto de entrada para dicho componente de servicio.

Sintaxis

MQZEP (*Hconfig*, *Función*, *EntryPoint*, *CompCode*, *Razón*)

Parámetros

Hconfig
Tipo: MQHCONFIG-entrada

Descriptor de contexto de configuración. Este descriptor de contexto representa el componente que se está configurando para este servicio instalable en particular. Debe ser el mismo que el componente

que el gestor de colas ha pasado a la función de configuración de componente en la llamada de inicialización de componente.

Función
Tipo: MQLONG - entrada

Identificador de función. Los valores válidos para esto se definen para cada servicio instalable.

Si se llama a MQZEP más de una vez para la misma función, la última llamada realizada proporciona el punto de entrada que se utiliza.

EntryPoint
Tipo: PMQFUNC-entrada

Punto de entrada de función. Es la dirección del punto de entrada proporcionado por el componente para realizar la función.

El valor NULL es válido e indica que este componente no proporciona la función. Se asume NULL para puntos de entrada que no están definidos utilizando MQZEP.

CompCode
Tipo: MQLONG - salida

Código de terminación. Tiene que ser uno de los valores siguientes:

MQCC_OK
Realización satisfactoria.

MQCC_FAILED
La llamada no se ha realizado satisfactoriamente.

Razón
Tipo: MQLONG - salida

Código de razón que califica *CompCode* .

Si *CompCode* es MQCC_OK:

MQRC_NONE
(0, X'000') No hay ninguna razón sobre la que informar.

Si *CompCode* es MQCC_FAILED:

MQRC_FUNCTION_ERROR
(2281, X'8E9') El identificador de función no es válido.

MQRC_HCONFIG_ERROR
(2280, X'8E8') El descriptor de contexto de configuración no es válido.

Para obtener más información sobre estos códigos de razón, consulte Códigos de razón de API.

Invocación en C

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

Declare los parámetros como se indica a continuación:

```
MQHCONFIG  Hconfig;     /* Configuration handle */
MQLONG     Function;    /* Function identifier */
PMQFUNC    EntryPoint;  /* Function entry point */
MQLONG     CompCode;    /* Completion code */
MQLONG     Reason;      /* Reason code qualifying CompCode */
```

MQZFP-Parámetros libres

La estructura MQZFP se utiliza en la llamada MQZ_FREE_USER para el parámetro *FreeParms* . Este parámetro especifica los datos relacionados con el recurso que se va a liberar.

Tabla 1. resume los campos de la estructura.

Tabla 599. Campos en MQZFP	
Campo	**Descripción**
StrucId	Identificador de la estructura
Versión	Versión
Reserved	Reservado, campo
CorrelationPtr	Puntero de correlación

Campos

StrucId
Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQZIC_STRUC_ID
Identificador de la estructura de contexto de identidad. Para el lenguaje de programación C, también se define la constante MQZIC_STRUC_ID_ARRAY; tiene el mismo valor que MQZIC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

version
Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZFP_VERSION_1
Estructura de parámetros libres Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQZFP_CURRENT_VERSION
Versión actual de la estructura de parámetros libres.

Reserved
Tipo: MQBYTE8 -entrada

Campo reservado. El valor inicial es nulo.

CorrelationPtr
Tipo: MQPTR-entrada

Puntero de correlación. Dirección de los datos de correlación relacionados con el recurso que se va a liberar.

Declaración 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 */
};
```

Campos

MQZIC-Contexto de identidad

La estructura MQZIC se utiliza en la llamada MQZ_AUTHENTICATE_USER para el parámetro *IdentityContext* .

La estructura MQZIC contiene información de contexto de identidad, que identifica al usuario de la aplicación que colocó primero el mensaje en una cola:

- El gestor de colas rellena el campo *UserIdentifier* con un nombre que identifica al usuario, la forma en que el gestor de colas puede hacerlo depende del entorno en el que se ejecuta la aplicación.
- El gestor de colas rellena el campo *AccountingToken* con una señal o un número que ha determinado a partir de la aplicación que ha colocado el mensaje.
- Las aplicaciones pueden utilizar el campo *Datos deApplIdentity* para cualquier información adicional que deseen incluir sobre el usuario (por ejemplo, una contraseña cifrada).

Las aplicaciones debidamente autorizadas pueden establecer el contexto de identidad utilizando la función MQZ_AUTHENTICATE_USER.

Un identificador de seguridad (SID) de sistemas Windows se almacena en el campo *AccountingToken* cuando se crea un mensaje en WebSphere MQ para Windows. El SID se puede utilizar para complementar el campo *UserIdentifier* y para establecer las credenciales de un usuario.

Tabla 1. resume los campos de la estructura.

Tabla 600. Campos en MQZIC

Campo	Descripción
StrucId	Identificador de la estructura
Versión	Versión
UserIdentifier	Identificador de usuario
AccountingToken	Señal de contabilidad
ApplIdentityData	Datos de identidad de la aplicación

Campos

StrucId

Tipo: MQCHAR4 -entrada

Identificador de estructura. El valor es el siguiente:

MQZIC_STRUC_ID
Identificador de la estructura de contexto de identidad. Para el lenguaje de programación C, también se define la constante MQZIC_STRUC_ID_ARRAY; tiene el mismo valor que MQZIC_STRUC_ID, pero es una matriz de caracteres en lugar de una serie.

version

Tipo: MQLONG - entrada

Número de versión de la estructura. El valor es el siguiente:

MQZIC_VERSION_1
Estructura de contexto de identidad Version-1 .

La constante siguiente especifica el número de versión de la versión actual:

MQZIC_CURRENT_VERSION
Versión actual de la estructura de contexto de identidad.

UserIdentifier

Tipo: MQCHAR12 -entrada

Identificador de usuario. Esto forma parte del contexto de identidad del mensaje. *UserIdentifier* especifica el identificador de usuario de la aplicación que ha originado el mensaje. El gestor de colas trata esta información como datos de tipo carácter, pero no define su formato. Para obtener más información sobre el campo *UserIdentifier* , consulte “UserIdentifier (MQCHAR12)” en la página 439.

AccountingToken
Tipo: MQBYTE32 -entrada

Señal de contabilidad. Esto forma parte del contexto de identidad del mensaje. *AccountingToken* permite que una aplicación haga que el trabajo realizado como resultado del mensaje se cargue correctamente. El gestor de colas trata esta información como una serie de bits y no comprueba su contenido. Para obtener más información sobre el campo *AccountingToken* , consulte “AccountingToken (MQBYTE32)” en la página 396.

ApplIdentityData
Tipo: MQCHAR32 -entrada

Datos de la aplicación relacionados con la identidad. Esto forma parte del contexto de identidad del mensaje. ApplIdentityLos datos son información definida por la suite de aplicaciones que se puede utilizar para proporcionar información adicional sobre el origen del mensaje. Por ejemplo, las aplicaciones que se ejecutan con autorización de usuario adecuada podrían establecerla para indicar si los datos de identidad son de confianza. Para obtener más información sobre el campo de datos ApplIdentity, consulte “Datos de ApplIdentity(MQCHAR32)” en la página 398.

Declaración 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 */
};
```

Campos

Material de referencia para el puente IBM WebSphere MQ para HTTP

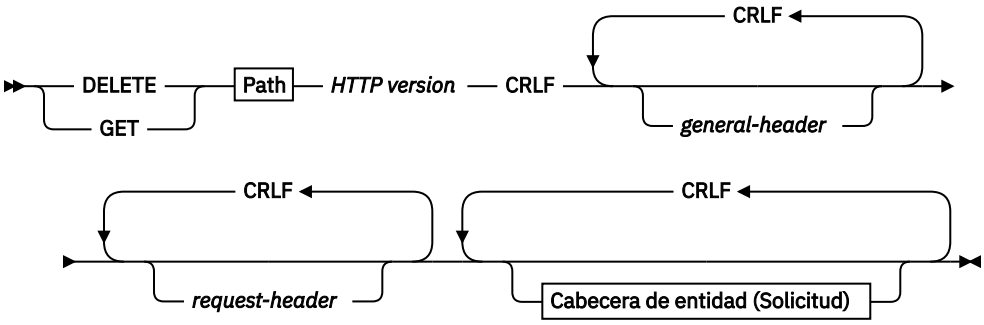
Temas de referencia para el puente IBM WebSphere MQ para HTTP, ordenados alfabéticamente

HTTP DELETE: WebSphere MQ Bridge for HTTP, mandato

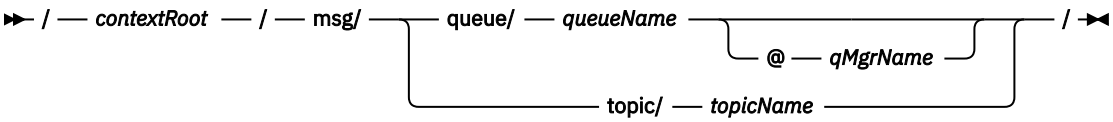
La operación HTTP **DELETE** obtiene un mensaje de una cola de WebSphere MQ o recupera una publicación de un tema. El mensaje se elimina de la cola. Si la publicación se retiene, no se elimina. Un mensaje de respuesta se devuelve al cliente incluyendo información sobre el mensaje.

Sintaxis

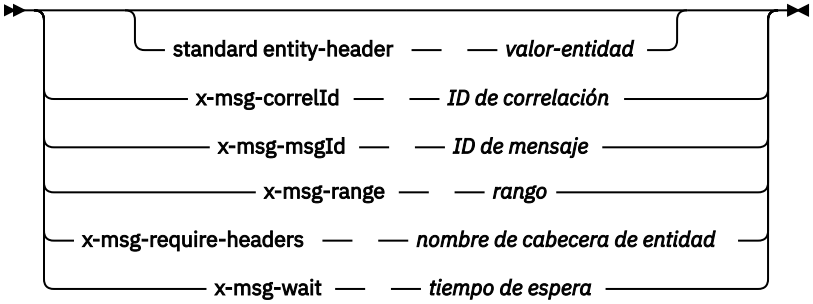
Solicitud



Path



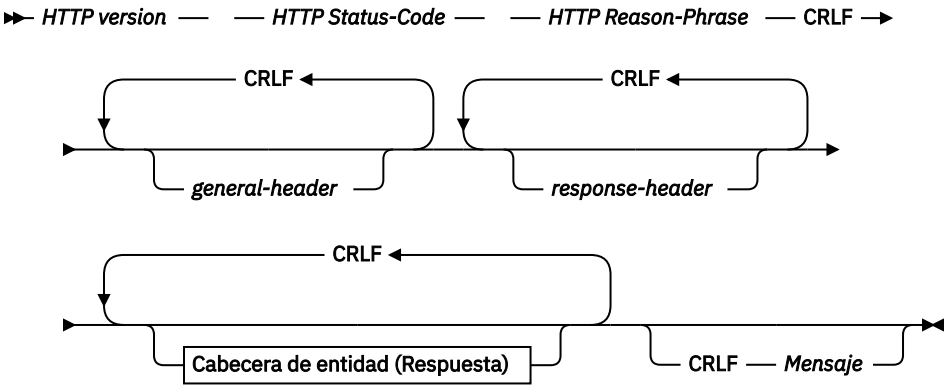
cabecera de entidad (Solicitud)



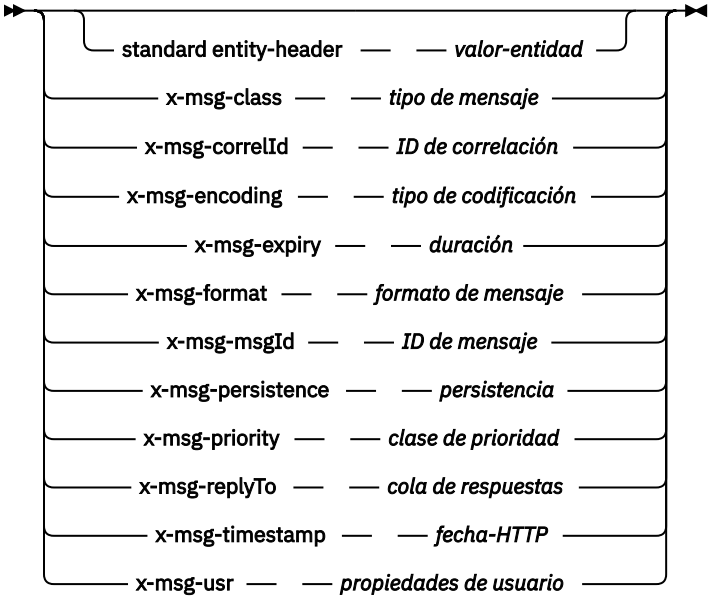
Nota:

1. If a question mark (?) is used it must be substituted with `%3f`. For example, `orange?topic` should be specified as `orange%3ftopic`.
2. @*qMgrName* is only valid on an HTTP **POST**

Respuesta



cabecera de entidad (Respuesta)



Parámetros de solicitud

path
Consulte “Formato de URI” en la página 1251.

Versión de HTTP
Versión HTTP; por ejemplo, `HTTP/1.1`

cabecera-general
Consulte HTTP/1.1 - 4.5 Campos de cabecera general.

cabecera-solicitud
Consulte HTTP/1.1 - 5.3 Campos de cabecera de solicitud. El campo Host es obligatorio en una solicitud HTTP/1.1 . A menudo se inserta automáticamente mediante la herramienta que utiliza para crear una solicitud de cliente.

cabecera-entidad (Solicitud)
Consulte HTTP/1.1 - 7.1 Campos de cabecera de entidad. Una de las cabeceras de entidad listadas en el diagrama de sintaxis de solicitud.

Parámetros de respuesta

path
Consulte “Formato de URI” en la página 1251.

Versión de HTTP
Versión HTTP; por ejemplo, `HTTP/1.1`

cabecera-general
Consulte HTTP/1.1 - 4.5 Campos de cabecera general.

cabecera-respuesta
Consulte HTTP/1.1 - 6.2 Campos de cabecera de respuesta.

cabecera de entidad (Respuesta)
Consulte HTTP/1.1 - 7.1 Campos de cabecera de entidad. Una de las cabeceras de entidad o respuesta listadas en el diagrama de sintaxis de respuesta. La `Longitud de contenido` siempre está presente en una respuesta. Se establece en cero si no hay ningún cuerpo de mensaje.

message
Cuerpo del mensaje.

Descripción

Si la solicitud HTTP **DELETE** es satisfactoria, el mensaje de respuesta contiene los datos recuperados de la cola WebSphere MQ . El número de bytes en el cuerpo del mensaje se devuelve en la cabecera HTTP `Content-Length` . El código de estado para la respuesta HTTP se establece en `200 OK`. Si `x-msg-range` se especifica como `0`o `0-0`, el código de estado de la respuesta HTTP es `204 No Content`.

Si la solicitud HTTP **DELETE** no es satisfactoria, la respuesta incluye un mensaje de error WebSphere MQ puente para HTTP y un código de estado HTTP.

Ejemplo de HTTP DELETE

HTTP **DELETE** obtiene un mensaje de una cola y suprime el mensaje o recupera y suprime una publicación. El ejemplo Java **HTTPDELETE** es un ejemplo de petición HTTP **DELETE** que lee un mensaje de una cola. También se puede crear una petición HTTP **DELETE** utilizando un formulario de navegador o un kit de herramientas AJAX en lugar de Java.

Figura 37 en la página 1219 es una solicitud HTTP para suprimir el siguiente mensaje en la cola denominada `myQueue`. En respuesta, se devuelve el cuerpo del mensaje al cliente. En términos de WebSphere MQ , HTTP **DELETE** es una obtención destructiva.

La solicitud contiene la cabecera de solicitud HTTP `x-msg-wait`, que indica al puente de WebSphere MQ para HTTP cuánto tiempo debe esperar a que llegue un mensaje a la cola. La solicitud también contiene

la cabecera de solicitud `x-msg-require-headers`, que especifica que el cliente debe recibir el ID de correlación de mensaje en la respuesta.

```
DELETE /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Figura 37. Ejemplo de una solicitud HTTP **DELETE***

Figura 38 en la página 1219, es la respuesta devuelta al cliente. Se devuelve el ID de correlación al cliente, tal como se ha solicitado en `x-msg-require-headers` de la solicitud.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 50
Content-Type: text/plain; charset=utf-8
x-msg-correlId: 1234567890

Here is my message body that is retrieved from the queue.
```

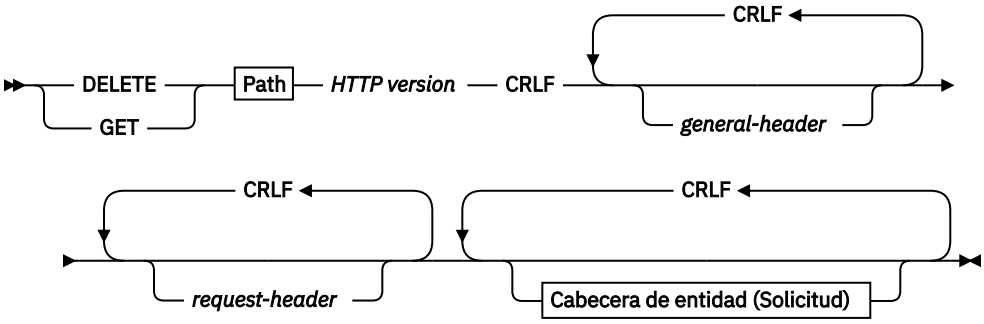
*Figura 38. Ejemplo de una respuesta de HTTP **DELETE***

HTTP GET: WebSphere MQ Bridge for HTTP, mandato

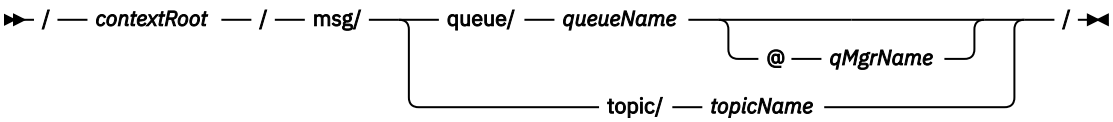
La operación HTTP **GET** obtiene un mensaje de una cola de WebSphere MQ . El mensaje se deja en la cola. La operación HTTP **GET** es equivalente a examinar una cola de WebSphere MQ .

Sintaxis

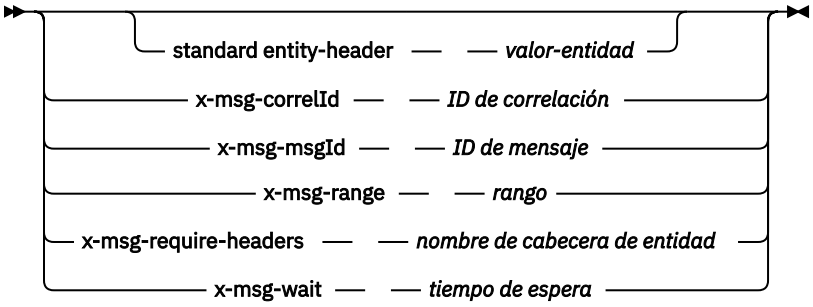
Solicitud



Path



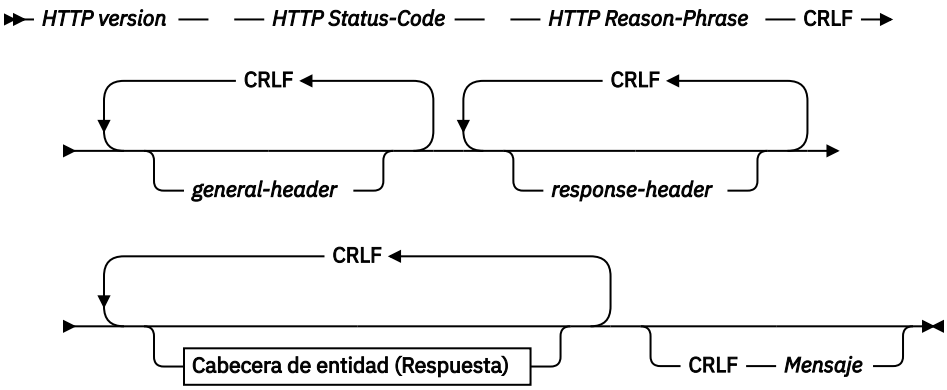
cabecera de entidad (Solicitud)



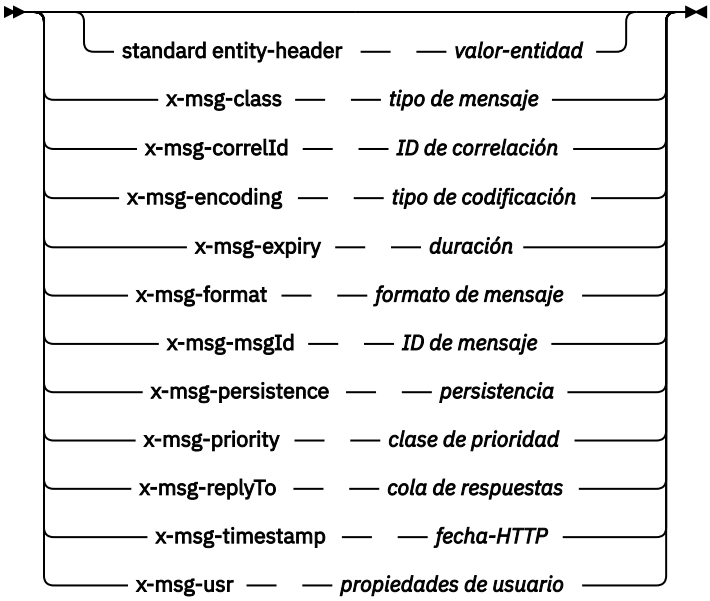
Nota:

1. If a question mark (?) is used it must be substituted with `%3f`. For example, `orange?topic` should be specified as `orange%3ftopic`.
2. @*qMgrName* is only valid on an HTTP **POST**

Respuesta



cabecera de entidad (Respuesta)



Parámetros de solicitud

path

Consulte “Formato de URI” en la página 1251.

Versión de HTTP
Versión HTTP; por ejemplo, `HTTP/1.1`

cabecera-general
Consulte HTTP/1.1 - 4.5 Campos de cabecera general.

cabecera-solicitud
Consulte HTTP/1.1 - 5.3 Campos de cabecera de solicitud. El campo Host es obligatorio en una solicitud HTTP/1.1 . A menudo se inserta automáticamente mediante la herramienta que utiliza para crear una solicitud de cliente.

cabecera-entidad (Solicitud)
Consulte HTTP/1.1 - 7.1 Campos de cabecera de entidad. Una de las cabeceras de entidad listadas en el diagrama de sintaxis de solicitud.

Parámetros de respuesta

path
Consulte “Formato de URI” en la página 1251.

Versión de HTTP
Versión HTTP; por ejemplo, `HTTP/1.1`

cabecera-general
Consulte HTTP/1.1 - 4.5 Campos de cabecera general.

cabecera-respuesta
Consulte HTTP/1.1 - 6.2 Campos de cabecera de respuesta.

cabecera de entidad (Respuesta)
Consulte HTTP/1.1 - 7.1 Campos de cabecera de entidad. Una de las cabeceras de entidad o respuesta listadas en el diagrama de sintaxis de respuesta. La `Longitud de contenido` siempre está presente en una respuesta. Se establece en cero si no hay ningún cuerpo de mensaje.

message
Cuerpo del mensaje.

Descripción

Si la solicitud HTTP **GET** es satisfactoria, el mensaje de respuesta contiene los datos recuperados de la cola WebSphere MQ . El número de bytes en el cuerpo del mensaje se devuelve en la cabecera HTTP `Content-Length` . El código de estado para la respuesta HTTP se establece en `200 OK`. Si `x-msg-range` se especifica como `0`o `0-0`, el código de estado de la respuesta HTTP es `204 No Content`.

Si la solicitud HTTP **GET** no es satisfactoria, la respuesta incluye un mensaje de error WebSphere MQ puente para HTTP y un código de estado HTTP.

Ejemplo de HTTP GET

HTTP **GET** obtiene un mensaje de una cola. El mensaje permanece en la cola. En términos de WebSphere MQ , HTTP **GET** es una solicitud de examen. Una petición HTTP **GET** se puede crear utilizando un cliente Java, un formulario de navegador o un kit de herramientas AJAX.

Figura 39 en la página 1222 es una solicitud HTTP para examinar el siguiente mensaje en la cola denominada `myQueue`.

La solicitud contiene la cabecera de solicitud HTTP `x-msg-wait`, que indica al puente de WebSphere MQ para HTTP cuánto tiempo debe esperar a que llegue un mensaje a la cola. La solicitud también contiene la cabecera de solicitud `x-msg-require-headers`, que especifica que el cliente debe recibir el `ID` de correlación de mensaje en la respuesta.

```
GET /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
x-msg-wait: 10
x-msg-require-headers: correlID
```

*Figura 39. Ejemplo de una solicitud HTTP **GET***

Figura 40 en la página 1222 es la respuesta devuelta al cliente. Se devuelve el ID de correlación al cliente, tal como se ha solicitado en `x-msg-require-headers` de la solicitud.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 50
Content-Type: text/plain; charset=utf-8
x-msg-correlId: 1234567890

Here is my message body that appears on the queue.
```

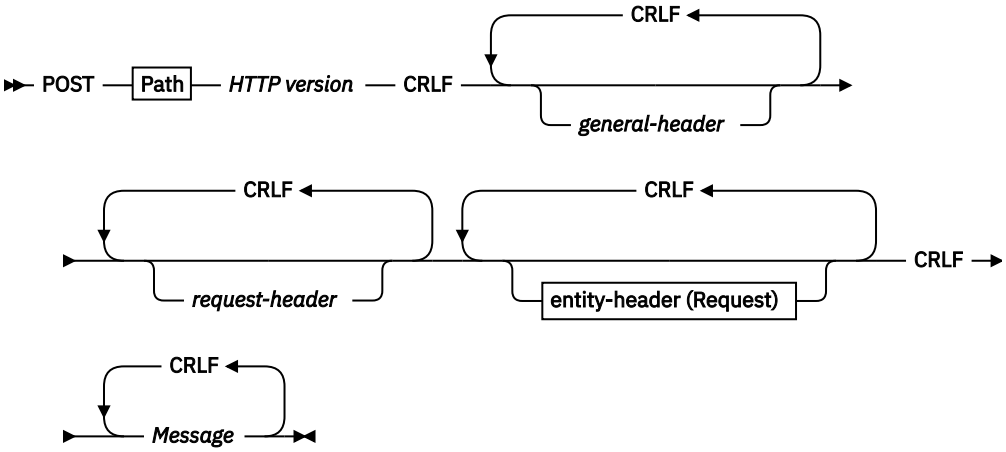
*Figura 40. Ejemplo de una respuesta de HTTP **GET***

HTTP POST: WebSphere MQ Bridge for HTTP, mandato

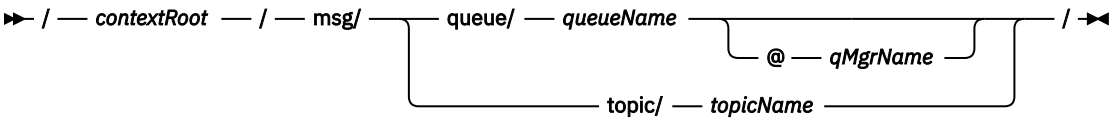
La operación HTTP **POST** coloca un mensaje en una cola de WebSphere MQ o publica un mensaje en un tema.

Syntax

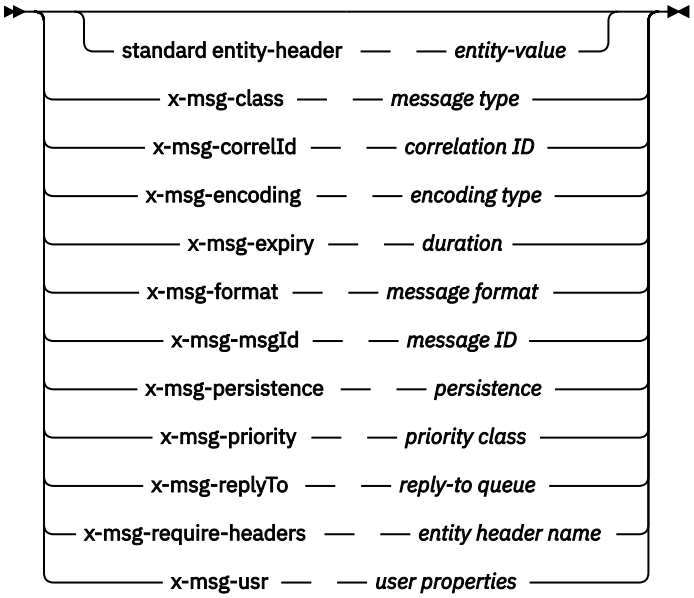
Request



Path



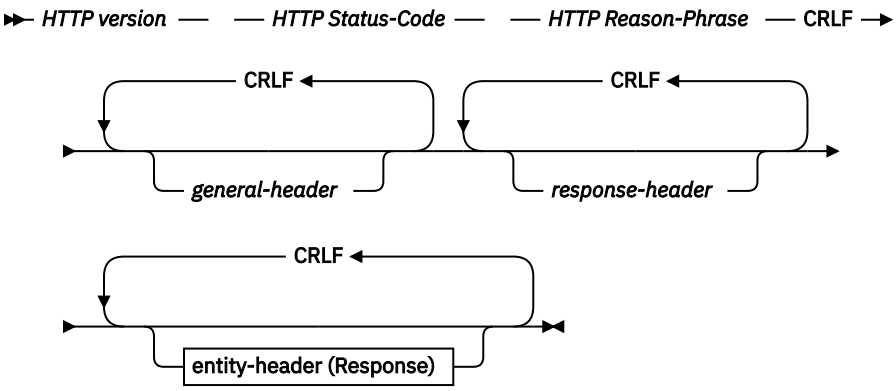
entity-header (Request)



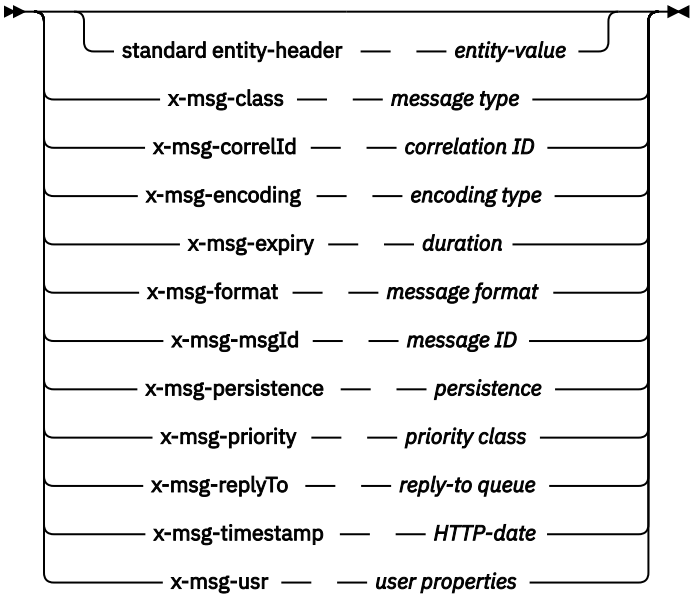
Nota:

1. If a question mark (?) is used it must be substituted with `%3f`. For example, `orange?topic` should be specified as `orange%3ftopic`.
2. @*qMgrName* is only valid on an HTTP **POST**

Response



entity-header (Response)



Parámetros de solicitud

path
Consulte “Formato de URI” en la página 1251.

Versión de HTTP
Versión HTTP; por ejemplo, HTTP/1.1

cabecera-general
Consulte HTTP/1.1 - 4.5 Campos de cabecera general.

cabecera-solicitud
Consulte HTTP/1.1 - 5.3 Campos de cabecera de solicitud. El campo Host es obligatorio en una solicitud HTTP/1.1 . A menudo se inserta automáticamente mediante la herramienta que utiliza para crear una solicitud de cliente.

cabecera-entidad (Solicitud)
Consulte HTTP/1.1 - 7.1 Campos de cabecera de entidad. Una de las cabeceras de entidad listadas en el diagrama de sintaxis de solicitud. La herramienta que utilice para crear una solicitud de cliente debe insertar `Content-Length` y `Content-Type` en una solicitud, y a menudo se insertan automáticamente. El `Tipo de contenido` debe coincidir con el tipo definido en la cabecera de entidad personalizada `x-msg-class` , si se especifica.

message
Mensaje para colocar en la cola o publicación para publicar en un tema.

Parámetros de respuesta

path
Consulte “Formato de URI” en la página 1251.

Versión de HTTP
Versión HTTP; por ejemplo, HTTP/1.1

cabecera-general
Consulte HTTP/1.1 - 4.5 Campos de cabecera general.

cabecera-respuesta
Consulte HTTP/1.1 - 6.2 Campos de cabecera de respuesta.

cabecera de entidad (Respuesta)
Consulte HTTP/1.1 - 7.1 Campos de cabecera de entidad. Una de las cabeceras de entidad o respuesta listadas en el diagrama de sintaxis de respuesta. La `Longitud de contenido` siempre está presente en una respuesta. Se establece en cero si no hay ningún cuerpo de mensaje.

Descripción

Si no se incluye ninguna cabecera `x-msg-usr` y la clase de mensaje es BYTES o TEXT, el mensaje colocado en la cola no tiene un `MQRFH2`.

Utilice la entidad HTTP y las cabeceras de solicitud en la solicitud HTTP **POST** para establecer las propiedades del mensaje que se coloca en la cola. También puede utilizar `x-msg-require-headers` para solicitar qué cabeceras se devuelven en el mensaje de respuesta.

Si la solicitud HTTP **POST** es satisfactoria, la entidad del mensaje de respuesta está vacía y su longitud de contenido es cero. El código de estado HTTP es `200 OK`.

Si la solicitud HTTP **POST** no es satisfactoria, la respuesta incluye un mensaje de error WebSphere MQ puente para HTTP y un código de estado HTTP. El mensaje WebSphere MQ no se coloca en la cola o tema.

Ejemplo de HTTP POST

HTTP **POST** coloca un mensaje en una cola o una publicación en un tema. El ejemplo Java **HTTPPOST** es un ejemplo de petición HTTP **POST** de un mensaje a una cola. En lugar de utilizar Java, puede crear una solicitud HTTP **POST** utilizando un formulario de navegador o un kit de herramientas AJAX en su lugar.

La Figura 41 en la página 1225 muestra una solicitud HTTP para colocar un mensaje en una cola denominada `myQueue`. Esta solicitud contiene la cabecera HTTP `x-msg-correlId` para establecer el ID de correlación del mensaje WebSphere MQ .

```
POST /msg/queue/myQueue/ HTTP/1.1
Host: www.example.org
Content-Type: text/plain
x-msg-correlID: 1234567890
Content-Length: 50

Here is my message body that is posted on the queue.
```

*Figura 41. Ejemplo de una solicitud HTTP **POST** para una cola*

Figura 42 en la página 1225 muestra la respuesta devuelta al cliente. No hay contenido de respuesta.

```
HTTP/1.1 200 OK
Date: Wed, 2 Jan 2007 22:38:34 GMT
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
Content-Length: 0
```

Figura 42. Ejemplo de una respuesta de HTTP POST

Cabeceras HTTP

El puente WebSphere MQ para HTTP da soporte a cabeceras HTTP de solicitud personalizadas, cabeceras HTTP de entidad personalizadas y un subconjunto de cabeceras HTTP estándar.

La práctica HTTP es añadir el prefijo `x-` a todas las cabeceras personalizadas, WebSphere MQ Bridge for HTTP headers tienen el prefijo `x-msg-`. Por ejemplo, para establecer la cabecera de prioridad, utilice `x-msg-priority`.

Nota:

- La mayoría de los valores de cabecera distinguen entre mayúsculas y minúsculas. Por ejemplo, cuando se utiliza la cabecera `msgId` , `NONE` es una palabra clave, mientras que `none` es un `msgID`.
- Las cabeceras con errores de ortografía se ignoran.

Cabeceras HTTP de entidad personalizadas

Las cabeceras HTTP de entidad personalizadas contienen información sobre los mensajes de WebSphere MQ . Utilizando cabeceras de entidad, puede establecer valores en el descriptor de mensaje (`MQMD`) o valores de consulta en `MQMD`. Una cabecera de entidad adicional, `x-msg-usr`, establece y devuelve cualquier información de propiedad de usuario que desee asociar a una solicitud.

Puede utilizar cabeceras de entidad en distintos contextos de solicitud HTTP:

DELETE
Sólo puede utilizar las cabeceras de entidad `x-msg-correlId`o `x-msg-msgId`, o ambas, con una solicitud HTTP de **DELETE** . El efecto de las cabeceras es seleccionar un mensaje concreto mediante `MsgId` y `CorrelId` en un `MQGET`, y suprimir el mensaje de su cola.

GET
Sólo puede utilizar las cabeceras de entidad `x-msg-correlId`o `x-msg-msgId`, o ambas, con una solicitud HTTP de **GET** . El efecto de las cabeceras es seleccionar un mensaje determinado mediante `MsgId` y `CorrelId` en un `MQGET` para examinar.

POST
Puede utilizar cualquier cabecera de entidad en una solicitud HTTP de **POST** , excepto `x-msg-timestamp`.

x-msg-require-headers
En cualquier solicitud **GET**, **POST** o **DELETE** HTTP, puede añadir varias cabeceras de entidad dentro de la cabecera de solicitud `x-msg-require-headers` , separadas por comas. El efecto es devolver las cabeceras de entidad especificadas en el mensaje de respuesta HTTP, que contiene el valor de la propiedad de mensaje asociada.

La descripción de cada cabecera lista en qué contextos procesa la cabecera el puente de WebSphere MQ para HTTP. Por ejemplo, en la cabecera **POST**, `x-msg-require-headers`, la cabecera es procesada por el puente WebSphere MQ para HTTP en una solicitud HTTP **POST** o en la cabecera de solicitud `x-msg-require-headers` en una solicitud HTTP **POST**, **GET**o **DELETE** . Si la cabecera se incluye en un contexto en el que no está permitida, se ignora la cabecera. No se informa de ningún error.

Puede colocar cualquier cabecera HTTP estándar en las solicitudes que procesará el servidor web u otros manejadores de solicitudes. De forma similar, la respuesta puede contener otras cabeceras HTTP estándar insertadas por el servidor web u otros manejadores de respuestas.

Cabeceras HTTP de solicitud personalizadas

Las tres cabeceras HTTP de solicitud personalizadas, `x-msg-range`, `x-msg-require-headers`y `x-msg-wait`, pasan información adicional sobre la solicitud HTTP al servidor. Actúan como modificadores de solicitud. Con `x-msg-range`, puede restringir la cantidad de datos de mensaje devueltos en una respuesta. Con `x-msg-require-headers`, puede solicitar la respuesta para contener información sobre el resultado de la solicitud. Con `x-msg-wait`, puede modificar el tiempo que el cliente espera una respuesta HTTP.

Cabeceras HTTP estándar

La cabecera de solicitud HTTP estándar `Host` debe especificarse en una solicitud HTTP/1.1 .

Las cabeceras de entidad HTTP estándar `Content-Length` y `Content-Type` se pueden especificar en una solicitud.

Las cabeceras de entidad HTTP estándar `Content-Length`, `Content-Location`, `Content-Range`, `Content-Type`y `Server` se pueden devolver en respuesta a una solicitud. Especifique una o más de las cabeceras HTTP estándar en la cabecera `x-msg-request-header` en el mensaje de solicitud.

class: cabecera de entidad x-msg-class HTTP

Establezca o devuelva el tipo de mensaje.

Tipo	Descripción
Nombre de cabecera HTTP	clase-mensaje-x
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	POST, x-msg-require-headers
Valores permitidos	**BYTES** **MAP** **STREAM** **TEXT**
Valor predeterminado	BYTES

Descripción

- En una solicitud HTTP **POST** , establece el tipo del mensaje creado.
- Si se especifica la cabecera de clase en un **GET** o **DELETE** , se devuelve un 400 Bad Request con el cuerpo de entidad MQHTTP40007.
- Especificado en x-msg-require-headers, establece x-msg-class en el mensaje de respuesta HTTP en el tipo de mensaje.
- Si se especifica un valor no válido para esta cabecera, se devuelve un mensaje MQHTTP40005 .
- Si no se especifica la cabecera x-msg-class y el tipo de contenido del mensaje es application/x-www-form-urlencoded, se supone que los datos son un objeto de correlación JMS.

Longitud de contenido: cabecera de entidad HTTP

Establezca o devuelva la longitud, en bytes, del cuerpo del mensaje.

Tipo	Descripción
Nombre de cabecera HTTP	Content-Length
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	x-msg-require-headers
Valor permitido y devuelto	***Integer value*** Longitud en bytes del cuerpo del mensaje.

Descripción

- Content-Length es opcional en una solicitud HTTP. Para un **GET** o **DELETE** , la longitud debe ser cero. Para **POST**, si se especifica Content-Length y no coincide con la longitud de la línea de mensaje, el mensaje se trunca o se rellena con nulos a la longitud especificada.
- La Longitud de contenido siempre se devuelve en la respuesta HTTP incluso cuando no hay contenido, en cuyo caso el valor es cero.

Content-Location: cabecera de entidad HTTP

Devuelve la cola o tema al que se hace referencia en la solicitud, en la cabecera Content-Location estándar en el mensaje de respuesta HTTP.

Tipo	Descripción
Nombre de cabecera HTTP	Ubicación de contenido
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	x-msg-require-headers
Valor devuelto	URI en el formato, */msg/queue/queuename* o */msg/topic/topicname*

Descripción

- Cuando se solicita en x-msg-require-headers, la cabecera de entidad Content-Location devuelve la cola o tema al que se hace referencia en la solicitud HTTP.

Content-Range: cabecera de entidad HTTP

Devuelve el rango de bytes seleccionados de un mensaje de WebSphere MQ en la cabecera Content-Range en una respuesta HTTP.

Tipo	Descripción
Nombre de cabecera HTTP	Rango de contenido
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	x-msg-require-headers
Valor devuelto	***String*** Devuelve el límite inferior, *m* y el límite superior, *n* de la subserie devuelta y *length* de todo el mensaje. Por ejemplo: *m-n/length*

Descripción

-
- El rango de contenido sólo se devuelve en la respuesta HTTP cuando se especifica rango de contenido en una solicitud **GET** o **DELETE** que contiene una cabecera de solicitud x-msg-range .
- Si se especifica x-msg-range en una solicitud **GET** o **DELETE** , el rango de bytes especificado en la cabecera Content-Range se devuelve en la respuesta. Por ejemplo, si se utiliza x-msg-range: 0-60 en una solicitud para un mensaje que contiene 100 bytes, la cabecera de rango de contenido contiene la serie 0-60/100
- Una solicitud x-msg-range también devuelve el rango de contenido en la cabecera x-msg-range en la respuesta HTTP.

Content-Type: cabecera de entidad HTTP

Establezca o devuelva la clase del mensaje JMS en un mensaje WebSphere MQ de acuerdo con el tipo de contenido HTTP.

Tipo	Descripción
Nombre de cabecera HTTP	`Content-Type`
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	**POST**, `x-msg-require-headers`
Valor permitido o devuelto	***media-type*** Para los tipos de soporte soportados, consulte Tabla 601 en la página 1229.

Tabla 601. Correlación entre `x-msg-class` y HTTP `Content-Type`

clase-mensaje-x	Tipo de contenido HTTP
BYTES	`application/octet-stream` `application/xml`
TEXT	`text/*`
MAP	`application/x-www-form-urlencoded` `application/xml` (opcional)
STREAM	`application/xml` (opcional)

Descripción

- En una solicitud HTTP **POST** , especifique `Content-Type` o `x-msg-class`. Si especifica ambos, deben ser coherentes o se devuelve una excepción `HTTP Bad Request` , `Status code 400` . Si omite ambos, `Content-Type` y `x-msg-class`, se presupone un `Content-Type` de `text/*` .
- El `Content-Type` siempre se establece en la respuesta a un HTTP **GET** o **DELETE** que tiene un cuerpo de mensaje. El `Content-Type` se establece de acuerdo con las reglas de Tabla 602 en la página 1229.

Tabla 602. Correlación de tipos de mensaje con `x-msg-class` y `Content-Type`

Formato de mensaje	Tipo de mensajes JMS	clase-mensaje-x	Content-Type
Cualquier cosa excepto `MQFMT_STRING`	Ninguna	BYTES	`application/octet-stream`
`MQFMT_STRING`	Ninguna	TEXT	`text/plain`
`MQFMT_NONE`	`jms_bytes`	BYTES	`application/octet-stream`
`MQFMT_NONE`	`jms_text`	TEXT	`text/plain`
`MQFMT_NONE`	`jms_map`	MAP	`application/xml`
`MQFMT_NONE`	`jms_stream`	STREAM	`application/xml`

correlId: HTTP x-msg-correlId cabecera-entidad

Establezca o devuelva el identificador de correlación.

Tipo	Descripción
Nombre de cabecera HTTP	x-msg-correlId
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	**DELETE**, **GET**, **POST**, x-msg-require-headers

Valores permitidos	***String value***
	Por ejemplo:
	`x-msg-correlId: mycorrelationid`
	Se permiten las series entre comillas; por ejemplo:
	`x-msg-correlId: "my id"`
	Hex value
	Un valor hexadecimal con el prefijo 0x:; por ejemplo:
	`x-msg-correlId: 0x:43c1d23a`
	El valor hexadecimal que sigue a 0x: está limitado a 48 caracteres que representan 24 bytes. Los datos adicionales se ignoran.

Valor predeterminado	No aplicable

Descripción

- En una solicitud HTTP **POST** , establece el ID de correlación del mensaje creado.
- En una solicitud HTTP **GET** o **DELETE** , selecciona el mensaje de la cola o tema. Si no existe ningún mensaje con el ID de correlación especificado, se devuelve una respuesta HTTP 504 Gateway Timeout . x-msg-correlId se puede utilizar con x-msg-msgID para seleccionar un mensaje de una cola o tema que coincida con ambos selectores.
- Especificado en x-msg-require-headers, establece x-msg-corelId en el mensaje de respuesta HTTP en el ID de correlación de un mensaje.
- Se permite un espacio en blanco horizontal después del prefijo 0x: .

Nota:

- Si especifica x-msg-correlId sin un valor en una solicitud HTTP **GET** o **DELETE** ; por ejemplo, "x-msg-correlId:", devuelve el siguiente mensaje en la cola o tema independientemente de su ID de correlación.
- Si especifica un selector de 24 caracteres o menos, o 0x: seguido de 48 caracteres o menos, el puente WebSphere MQ para HTTP utiliza un selector optimizado para mejorar el rendimiento.
- Se utiliza un selector de mensajes JMS que contiene JMSCorrelationID al seleccionar mensajes de la cola. Este selector se comporta como se describe en Comportamiento de selección.

encoding: cabecera de entidad HTTP x-msg-encoding

Establezca o devuelva la codificación del mensaje.

Tipo	Descripción
Nombre de cabecera HTTP	codificación-x-msg

Tipo	Descripción
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	**POST**, `x-msg-require-headers`
Valores permitidos	Una lista separada por comas de los valores siguientes: **DECIMAL_NORMAL** **DECIMAL_REVERSED** **FLOAT_IEEE_NORMAL** **FLOAT_IEEE_REVERSED** **FLOAT_S390** **INTEGER_NORMAL** **INTEGER_REVERSED** Por ejemplo: `x-msg-encoding: INTEGER_NORMAL,DECIMAL_NORMAL,FLOAT_IEEE_NORMAL` **Nota:** El valor distingue entre mayúsculas y minúsculas
Valor predeterminado	`DECIMAL_NORMAL, FLOAT_IEEE_NORMAL, INTEGER_NORMAL`

Descripción

- En una solicitud HTTP **POST** , especifica la codificación del mensaje creado.
- En una solicitud HTTP **GET** o **DELETE** , la cabecera `x-msg-encoding` se ignora.
- Especificado en `x-msg-require-headers`, establece `x-msg-encoding` en el mensaje de respuesta HTTP en la propiedad de codificación de un mensaje.

caducidad: HTTP `x-msg-expire` cabecera-entidad

Establezca o devuelva la duración de caducidad del mensaje.

Tipo	Descripción
Nombre de cabecera HTTP	`x-msg-caducidad`
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	**POST**, `x-msg-require-headers`

Valores permitidos	**UNLIMITED**
	Por ejemplo:
	`x-msg-expiry: UNLIMITED`
	Integer value
	Milisegundos antes de la caducidad.
	Por ejemplo:
	`x-msg-expiry: 10000`

Valor predeterminado	`UNLIMITED`

Descripción

- Cuando se establece en una solicitud HTTP **POST** , el mensaje de solicitud caduca en el tiempo especificado.
- En una solicitud HTTP **GET** o **DELETE** , la cabecera `x-msg-expire` se ignora.
- Especificado en `x-msg-require-headers`, establece `x-msg-expire` en el mensaje de respuesta HTTP en la hora de caducidad de un mensaje.
- `UNLIMITED` especifica que el mensaje nunca caduca.
- La caducidad de un mensaje se inicia desde el momento en que el mensaje llega a la cola, ya que se ignora la latencia de red.
- El valor máximo está limitado por WebSphere MQ a 214748364700 milisegundos. Si se especifica un valor mayor que este, se presupone el tiempo de caducidad máximo posible.

`format`: HTTP `x-msg-format` cabecera-entidad

Establezca o devuelva el formato de mensaje WebSphere MQ .

Tipo	Descripción
Nombre de cabecera HTTP	`formato-mensaje-x`
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	**POST**, `x-msg-require-headers`

Valores permitidos	**NONE**
	Por ejemplo:
	`x-msg-format: NONE`
	String value
	Cualquier valor definido por el usuario de hasta ocho caracteres. Por ejemplo:
	`x-msg-format: myformat`

Valor predeterminado	None

Descripción

- Cuando se establece en una solicitud HTTP **POST** , establezca el formato del mensaje de solicitud.
- En una solicitud HTTP **GET** o **DELETE** , la cabecera `x-msg-format` se ignora.
- Especificado en `x-msg-require-headers`, establece `x-msg-format` en el mensaje de respuesta HTTP en el formato de un mensaje.
- `NONE` distingue entre mayúsculas y minúsculas e indica que el formato del mensaje está en blanco.
- Se utiliza el valor de `x-msg-format` , incluso si contradice el tipo de soporte de la solicitud HTTP. Consulte Tabla 603 en la página 1232.

Tabla 603. Correlación de content-type y `x-msg-class` con el formato de mensaje

clase-mensaje-x	**Content-type**	**Formato de mensaje en cola/tema**
BYTES	• `application/octet-stream` • aplicación/xml	Mensaje WebSphere MQ : `MQFMT` establézcalo en `MQC.MQFMT_NONE`

Tabla 603. Correlación de content-type y `x-msg-class` con el formato de mensaje (continuación)		
clase-mensaje-x	**Content-type**	**Formato de mensaje en cola/tema**
TEXT	• `text/*`	Mensaje WebSphere MQ : `MQFMT` establézcalo en `MQC.MQFMT_STRING`
MAP	• `application/x-www-form-urlencoded` • `application/xml` (opcional)	`JMSMap`
STREAM	• `application/xml` (opcional)	`JMSStream`

msgId: HTTP x-msg-msgId cabecera-entidad

Establezca o devuelva el identificador de mensaje.

Tipo	Descripción
Nombre de cabecera HTTP	`x-msg-msgId`
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	**DELETE**, **GET**, **POST**, `x-msg-require-headers`

Valores permitidos	***String value***
	Por ejemplo:
	`x-msg-msgId: mymsgid`
	Series entre comillas, por ejemplo, `x-msg-msgId: "my id"`
	Hex value
	Un valor hexadecimal con el prefijo `0x:`; por ejemplo,
	`x-msg-msgId: 0x:43c1d23a`

Valor predeterminado	No aplicable

Descripción

- En una solicitud HTTP **POST** , establece el ID de mensaje del mensaje creado.
- En una solicitud HTTP **GET** o **DELETE** , selecciona el mensaje de la cola o tema. Si no existe ningún mensaje con el ID de mensaje especificado, se devuelve una respuesta `HTTP 504 Gateway Timeout` . `x-msg-msgId` se puede utilizar con `x-msg-correlID` para seleccionar un mensaje de una cola o tema que coincida con ambos selectores.
- Especificado en `x-msg-require-headers`, devuelve `x-msg-msgId` en la respuesta HTTP al ID de mensaje de un mensaje.
- Se permite un espacio en blanco horizontal después del prefijo `0x:`.

Nota: Si especifica `x-msg-msgId` sin un valor en una solicitud HTTP **GET** o **DELETE** ; por ejemplo, `"x-msg-msgId:"`, devuelve el siguiente mensaje en la cola o tema independientemente de su ID de mensaje.

Se utiliza un selector de mensajes JMS que contiene JMSMessageID al seleccionar mensajes de la cola. Este selector se comporta como se describe en Comportamiento de selección .

persistence: HTTP x-msg-persistence cabecera-entidad

Establezca o devuelva la persistencia del mensaje.

Tipo	Descripción
Nombre de cabecera HTTP	persistencia-mensaje-x
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	**POST**, x-msg-require-headers

Valores permitidos	**NON_PERSISTENT**
	El mensaje no sobrevive a las anomalías del sistema o a los reinicios del gestor de colas.
	Por ejemplo:
	`x-msg-persistence: NON_PERSISTENT`
	PERSISTENT
	El mensaje sobrevive a las anomalías del sistema y a los reinicios del gestor de colas.
	Por ejemplo:
	`x-msg-persistence: PERSISTENT`
	AS_DESTINATION
	Sólo se aplica a **POST** .
	Utilice la persistencia predeterminada del destino según determine el proveedor de mensajes.
	Nota: Distingue entre mayúsculas y minúsculas

Valor predeterminado	NON_PERSISTENT

Descripción

- Cuando se establece en una solicitud HTTP **POST** , establezca la persistencia del mensaje de solicitud.
- En una solicitud HTTP **GET** o **DELETE** , la cabecera x-msg-persistence se ignora.
- Especificado en x-msg-require-headers, establece x-msg-persistence en el mensaje de respuesta HTTP en la persistencia de un mensaje.

priority: cabecera de entidad x-msg-priority HTTP

Establezca o devuelva la prioridad del mensaje.

Tipo	Descripción
Nombre de cabecera HTTP	prioridad-mensaje-x
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	**POST**, x-msg-require-headers
Valores permitidos	**LOW** Por ejemplo: `x-msg-priority: LOW`

Tipo	Descripción

	MEDIUM
	Esta prioridad es igual a un nivel de prioridad 4 de WebSphere MQ . Por ejemplo:
	`x-msg-priority: MEDIUM`
	HIGH
	Por ejemplo:
	`x-msg-priority: HIGH`
	Integer value
	Una representación de serie de un entero en el rango de 0 a 9; por ejemplo,
	`x-msg-priority: 3`
	AS_DESTINATION
	Sólo se aplica a **POST** .
	Utilice la prioridad predeterminada del destino según determine el proveedor de mensajes.
	Nota: Distingue entre mayúsculas y minúsculas

Valor predeterminado	MEDIUM

Descripción

- Cuando se establece en una solicitud HTTP **POST** , establezca la prioridad del mensaje de solicitud.
- En una solicitud HTTP **GET** o **DELETE** , la cabecera `x-msg-priority` se ignora.
- Especificado en `x-msg-require-headers`, establece `x-msg-priority` en el mensaje de respuesta HTTP en la prioridad de un mensaje.

`priority`: cabecera de entidad `x-msg-priority` HTTP

Establezca o devuelva la prioridad del mensaje.

Tipo	Descripción
Nombre de cabecera HTTP	`prioridad-mensaje-x`
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	**POST**, `x-msg-require-headers`

Valores permitidos	**LOW**
	Por ejemplo:
	`x-msg-priority: LOW`
	MEDIUM
	Esta prioridad es igual a un nivel de prioridad 4 de WebSphere MQ . Por ejemplo:
	`x-msg-priority: MEDIUM`

Tipo	Descripción

	HIGH
	Por ejemplo:
	`x-msg-priority: HIGH`
	Integer value
	Una representación de serie de un entero en el rango de 0 a 9; por ejemplo,
	`x-msg-priority: 3`
	AS_DESTINATION
	Sólo se aplica a **POST** .
	Utilice la prioridad predeterminada del destino según determine el proveedor de mensajes.
	Nota: Distingue entre mayúsculas y minúsculas

Valor predeterminado	MEDIUM

Descripción

- Cuando se establece en una solicitud HTTP **POST** , establezca la prioridad del mensaje de solicitud.
- En una solicitud HTTP **GET** o **DELETE** , la cabecera `x-msg-priority` se ignora.
- Especificado en `x-msg-require-headers`, establece `x-msg-priority` en el mensaje de respuesta HTTP en la prioridad de un mensaje.

replyTo: HTTP x-msg-replyTo cabecera-entidad

Establezca o devuelva la cola de respuesta de mensaje y el nombre del gestor de colas.

Tipo	Descripción
Nombre de cabecera HTTP	`x-msg-replyTo`
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	**POST**, `x-msg-require-headers`

Valores permitidos	***URI***
	Un URI punto a punto; por ejemplo,
	`x-msg-replyTo: /msg/queue/myReplyQueue`
	`x-msg-replyTo: /msg/queue/myReplyQueue@myReplyQueueManager`
	Nota: Distingue entre mayúsculas y minúsculas

Valor predeterminado	MEDIUM

Descripción

- Cuando se establece en una solicitud HTTP **POST** , establezca el destino replyTo del mensaje de solicitud.
- En una solicitud HTTP **GET** o **DELETE** , la cabecera `x-msg-replyTo` se ignora.
- Especificado en `x-msg-require-headers`, establece `x-msg-replyTo` en el mensaje de respuesta HTTP en la cola de respuesta y el nombre del gestor de colas de un mensaje .

Nota: El URI en la respuesta HTTP puede incluir el nombre del gestor de colas al que está conectado el puente WebSphere MQ para HTTP.

`Servidor`: cabecera de respuesta HTTP

Devuelve información sobre el servidor y el protocolo al que está conectado el cliente.

Tipo	Descripción
Nombre de cabecera HTTP	SERVER
Tipo de cabecera HTTP	Cabecera de respuesta
Válido en mensaje de solicitud HTTP	`x-msg-require-headers`

Valor devuelto	`WMQ-HTTP/1.1 JEE-Bridge/1.1`
	o
	`Server: Product-token WMQ-HTTP/1.1 JEE-Bridge/1.1`

Descripción

- Si WebSphere MQ Bridge for HTTP se despliega en un servidor de aplicaciones, los detalles del puente WebSphere MQ para HTTP se añaden a la cabecera de respuesta del servidor. Por ejemplo, el puente WebSphere MQ para HTTP desplegado en WebSphere Application Server Community Edition, denominado Apache-Coyote, proporciona la respuesta:

```
Server: Apache-Coyote/1.1 WMQ-HTTP/1.1 JEE-Bridge/1.1
```

`require-headers`: HTTP `x-msg-require-headers` cabecera-solicitud

Establezca qué cabeceras deben devolverse en el mensaje de respuesta HTTP.

Tipo	Descripción
Nombre de cabecera HTTP	`x-msg-require-headers`
Tipo de cabecera HTTP	Cabecera de solicitud
Válido en mensaje de solicitud HTTP	**POST**, **GET**, **DELETE**
Valores permitidos	Una lista separada por comas de los nombres de cabecera de entidad: **ALL** **ALL-USR** **class** **content-location** **correlId** **encoding** **expiry** **format** **msgId** **NO_require-headers** **persistence**

Tipo	Descripción

	priority
	replyTo
	server
	timestamp
	usr-*property name*
	Por ejemplo:
	`x-msg-require-headers: msgId`
	o,
	`x-msg-require-headers: expiry,correlId,timestamp`
	Para solicitar una propiedad específica:
	`x-msg-require-headers: usr-myCustomProperty`
	Para solicitar todas las propiedades:
	`x-msg-require-headers: ALL-USR, ALL`

Valor predeterminado	`NO_require-headers`

Descripción

- El valor de `x-msg-require-headers` no distingue entre mayúsculas y minúsculas, excepto en los casos de las constantes `ALL`, `NO_require-headers`y `ALL-USR` y la variable *`property-name`* .

`timestamp`: cabecera de entidad HTTP `x-msg-timestamp`

Devuelve la indicación de fecha y hora del mensaje.

Tipo	Descripción
Nombre de cabecera HTTP	`x-msg-indicación-hora`
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	`x-msg-require-headers`
Valor devuelto	***HTTP-date*** Una fecha con el formato; día, fecha mes año hora hora-zona; por ejemplo, `Sun, 06 Nov 1994 08:49:37 GMT` Definido por RFC 822y actualizado en RFC 1123.
Valor predeterminado	No aplicable

Descripción

- En una solicitud HTTP **POST**, **GET** o **DELETE** , la cabecera `x-msg-timestamp` se ignora.
- Especificado en `x-msg-require-headers`, establece `x-msg-timestamp` en el mensaje de respuesta HTTP en la indicación de fecha y hora de un mensaje.

usr: cabecera de entidad HTTP x-msg-usr

Establezca o devuelva las propiedades de usuario.

Tipo	Descripción
Nombre de cabecera HTTP	`x-msg-usr`
Tipo de cabecera HTTP	Cabecera de entidad
Válido en mensaje de solicitud HTTP	**POST**, `x-msg-require-headers`

Valores permitidos	Consulte “Sintaxis” en la página 1239; por ejemplo,
	`x-msg-usr: myProp1;5;i1,`
	`x-msg-usr: myProp2;"My String";string`

Valor predeterminado	No aplicable

Descripción

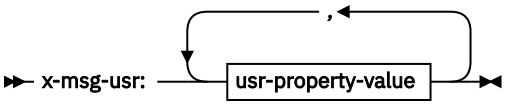
- Cuando se establece en una solicitud HTTP **POST** , establezca las propiedades de usuario del mensaje de solicitud.
- En una solicitud HTTP **GET** o **DELETE** , la cabecera `x-msg-usr` se ignora.
- Especificado en `x-msg-require-headers`, establece `x-msg-usr` en el mensaje de respuesta HTTP en las propiedades de usuario de un mensaje.
- Se pueden establecer varias propiedades en un mensaje. Especifique varias propiedades separadas por comas en una sola cabecera `x-msg-usr` o utilizando dos o más instancias separadas de la cabecera `x-msg-usr` .
- Puede solicitar que se devuelva una propiedad específica en la respuesta a una solicitud **GET** o **DELETE** . Especifique el nombre de la propiedad en la cabecera `x-msg-require-headers` de la solicitud, utilizando el prefijo `usr-`. Por ejemplo:

```
x-msg-require-headers: usr-myProp1
```

- Para solicitar que se devuelvan todas las propiedades de usuario en una respuesta, utilice la constante `ALL-USR` . Por ejemplo:

```
x-msg-require-headers: ALL-USR
```

Sintaxis

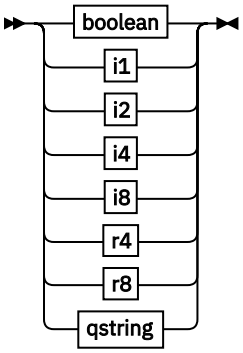


usr-property-value

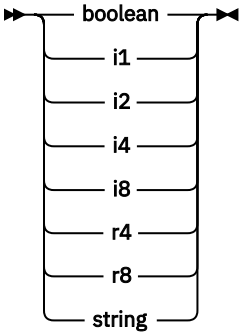
property-name

serie

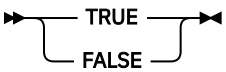
usr-value



usr-type



boolean

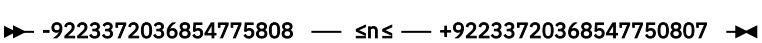


i1

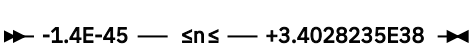
i2

i4

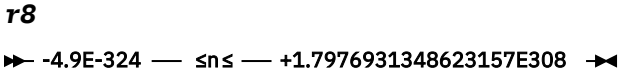
i8



r4



r8



qstring

wait: HTTP x-msg-wait cabecera-solicitud

Establezca el periodo de tiempo que se debe esperar a que llegue un mensaje, antes de devolver un mensaje de respuesta de `HTTP 504 Gateway Timeout`.

Tipo	Descripción
Nombre de cabecera HTTP	`espera-mensaje-x`

Tipo	Descripción
Tipo de cabecera HTTP	Cabecera de solicitud
Válido en mensaje de solicitud HTTP	**GET**, **DELETE**

Valor permitido	**NO_WAIT**
	Por ejemplo:
	`x-msg-wait: NO_WAIT`
	Integer value
	El número de milisegundos que el puente de WebSphere MQ para HTTP espera a que llegue un mensaje; por ejemplo,
	`x-msg-wait: 8`

Valor predeterminado	NO_WAIT

Descripción

- En una solicitud HTTP **POST** , la cabecera `x-msg-wait` se ignora.
- En una solicitud HTTP **GET** o **DELETE** , `x-msg-wait` especifica el tiempo que se debe esperar a que llegue un mensaje antes de devolver una respuesta `HTTP 504 Gateway Timeout` .
- `NO_WAIT` distingue entre mayúsculas y minúsculas.
- El tiempo de espera máximo predeterminado es 35000. Puede cambiar el valor predeterminado estableciendo el parámetro `maximum_wait_time` del servlet. Consulte la sección Instalación, configuración y verificación de WebSphere MQ Bridge for HTTP para obtener más información.
- Si establece un valor mayor que `maximum_wait_time`, se utiliza `maximum_wait_time` en su lugar.

Códigos de retorno HTTP

Lista de códigos de retorno del puente WebSphere MQ para HTTP

El puente WebSphere MQ para HTTP devuelve cuatro tipos de error:

Errores de servlet
MQHTTP0001 y MQHTTP0002 son errores de servlet. Se registran, pero no se devuelven al cliente HTTP.

Operaciones correctas
Un código de estado HTTP en el rango 200-299 indica una operación satisfactoria.

Errores de cliente
Un código de estado HTTP en el rango 400-499 indica un error de cliente. WebSphere MQ Bridge for HTTP en el rango MQHTTP40001 - MQHTTP49999 corresponden a errores de cliente.

Errores del servidor

Un código de estado HTTP en el rango 500-599 indica un error de cliente. WebSphere MQ Bridge for HTTP en el rango MQHTTP50001 - MQHTTP59999 corresponden a errores de servidor.

Si se produce un error de servidor, se genera un rastreo de pila completo en el registro de errores del servidor de aplicaciones. El rastreo de pila también se devuelve al cliente HTTP en la respuesta HTTP. Maneje el rastreo de pila en la aplicación cliente o consulte al administrador del servidor de aplicaciones para resolver el problema.

Si el rastreo de pila contiene errores de adaptador de recursos, consulte la documentación del adaptador de recursos.

Lista alfabética de códigos de retorno

HTTP 200: Correcto

Esta clase de código de estado indica que la solicitud se ha recibido, entendido y aceptado correctamente.

Código de estado HTTP
`200 OK`

HTTP 204: Sin contenido

Se ha enviado después de un HTTP **GET** o **DELETE** satisfactorio y `x-msg-range: 0` se ha enviado en la solicitud.

Código de estado HTTP
`204 No Content`

MQHTTP0001: No se ha especificado ninguna fábrica de conexiones en el contexto de servlet

Error de servlet

Explicación
Error de servlet

Código de estado HTTP
Ninguna

Respuesta del programador
El lugar donde se registran estos errores es específico del servidor de aplicaciones. Consulte la documentación del servidor de aplicaciones.

MQHTTP0002: No se ha podido obtener el gestor de conexiones para *`queueOrTopic`* utilizando el nombre JNDI de *`jndiNameTried`*

Error de servlet

Explicación
Error de servlet

Código de estado HTTP
Ninguna

Respuesta del programador
El lugar donde se registran estos errores es específico del servidor de aplicaciones. Consulte la documentación del servidor de aplicaciones.

MQHTTP40001: Reservado

Reserved

Código de estado HTTP
`400 Bad Request`

MQHTTP40002: El URI no es válido para el transporte WebSphere MQ para HTTP

El URI especificado en la solicitud HTTP no es válido.

Explicación
El URI especificado en la solicitud HTTP no es válido.

Código de estado HTTP
`400 Bad Request`

Respuesta del programador
Confirme que el formato y la sintaxis del URI especificado son correctos.

MQHTTP40003: URI no es válido. `@qmgr` sólo es válido en POST

La opción de URI `@qmgr` se ha especificado en un URI para una solicitud HTTP que no es una solicitud **POST** .

Explicación
La opción de URI `@qmgr` se ha especificado en un URI para una solicitud HTTP que no es una solicitud **POST** .

Código de estado HTTP
`400 Bad Request`

Respuesta del programador
Si está intentando colocar un mensaje utilizando el verbo **POST** , cambie la solicitud HTTP por una solicitud **POST** . Si está intentando obtener un mensaje utilizando los verbos **DELETE** o **GET** , elimine `@qmgr` del URI.

MQHTTP40004: Se ha especificado `Content-Type` no válido

El campo de cabecera `Content-Type` especificado en una solicitud **POST** no es compatible con el valor de cabecera `x-msg-class` .

Explicación
El campo de cabecera `Content-Type` especificado en una solicitud **POST** no es compatible con el valor de cabecera `x-msg-class` .

Código de estado HTTP
`400 Bad Request`

Respuesta del programador
Cambie el campo de cabecera `Content-Type` por uno que esté soportado. La cabecera `Content-Type` debe ser compatible con el campo de cabecera `x-msg-class` especificado.

MQHTTP40005: Valor de cabecera de mensaje incorrecto

Se ha especificado un campo de cabecera soportado con un valor que no es válido para la solicitud especificada.

Explicación
Se ha especificado un campo de cabecera soportado con un valor que no es válido para la solicitud especificada.

Código de estado HTTP
`400 Bad Request`

Respuesta del programador
Cambie el valor especificado para el campo de cabecera especificado por un valor que sea válido. Compruebe las mayúsculas y minúsculas del valor especificado, ya que algunos campos de cabecera tienen valores sensibles a las mayúsculas y minúsculas.

MQHTTP40006: *Header_name* no es una cabecera de solicitud válida

Una cabecera que sólo es válida en un mensaje de respuesta HTTP se ha especificado en un mensaje de solicitud HTTP.

Explicación
Una cabecera que sólo es válida en un mensaje de respuesta HTTP se ha especificado en un mensaje de solicitud HTTP.

Código de estado HTTP
`400 Bad Request`

Respuesta del programador
Elimine las cabeceras de la solicitud HTTP que sólo son válidas en una respuesta HTTP; por ejemplo, `x-msg-timestamp`.

MQHTTP40007: *Header_name* sólo es válido el ...

Se ha especificado una cabecera en una solicitud HTTP, pero el campo de cabecera no es válido para el verbo de solicitud especificado.

Explicación
Se ha especificado una cabecera en una solicitud HTTP, pero el campo de cabecera no es válido para el verbo de solicitud especificado.

Código de estado HTTP
`400 Bad Request`

Respuesta del programador
Elimine las cabeceras de la solicitud HTTP que no son válidas para el verbo de solicitud determinado. Por ejemplo, `x-msg-encoding` es válido para solicitudes HTTP **POST** , pero no para solicitudes HTTP **GET** o HTTP **DELETE** .

MQHTTP40008: *Header_name* es ...

Se ha sobrepasado la longitud máxima para el campo de cabecera especificado.

Explicación
Se ha sobrepasado la longitud máxima para el campo de cabecera especificado.

Código de estado HTTP
`400 Bad Request`

Respuesta del programador
Cambie el valor del campo de cabecera por un valor que esté dentro del rango permitido para el campo de cabecera.

MQHTTP40009: El campo de cabecera *header_field* no es válido para ...

Un campo de cabecera especificado en una solicitud HTTP no está soportado por el proveedor de mensajería al que está conectado el puente WebSphere MQ para HTTP.

Explicación
Un campo de cabecera especificado en una solicitud HTTP no está soportado por el proveedor de mensajería al que está conectado el puente WebSphere MQ para HTTP. El error se produce si se utiliza un proveedor de mensajería que no puede dar soporte a todas las características del puente WebSphere MQ para HTTP.

Código de estado HTTP
`400 Bad Request`

Respuesta del programador
Elimine la cabecera no soportada de la solicitud HTTP.

MQHTTP40010: No se ha podido analizar el mensaje con `Content-Type` *content_type*

El contenido de la solicitud HTTP no es compatible con el `Content-Type` de la solicitud.

Explicación
El contenido de la solicitud HTTP no es compatible con el `Content-Type` de la solicitud. Una causa común está mal formada `application/x-www-form-urlencoded` o `application/xml data`.

Código de estado HTTP
`400 Bad Request`

Respuesta del programador
Corrija el contenido de la solicitud HTTP para que esté en el formato correcto para el `Content-Type` de la solicitud.

MQHTTP40301: Tiene prohibido el acceso ...

El puente WebSphere MQ para HTTP no ha podido autenticarse a sí mismo para el destino especificado.

Explicación
El puente WebSphere MQ para HTTP no ha podido autenticarse a sí mismo para el destino especificado.

Código de estado HTTP
`403 Forbidden`

Respuesta del programador
Cambie las propiedades de autenticación del destino para que WebSphere MQ Bridge for HTTP esté autorizado para conectarse a él. De forma alternativa, especifique un destino al que el puente WebSphere MQ para HTTP esté autorizado para conectarse.

MQHTTP40302: Está prohibido ...

El puente WebSphere MQ para HTTP no ha podido conectarse al gestor de colas.

Explicación
El puente WebSphere MQ para HTTP no ha podido conectarse al gestor de colas. El puente WebSphere MQ para la configuración de seguridad HTTP es incorrecto.

Código de estado HTTP
`403 Forbidden`

Respuesta del programador
Cambie la configuración de autenticación del gestor de colas para que WebSphere MQ Bridge for HTTP esté autorizado para conectarse a él. De forma alternativa, configure el puente WebSphere MQ para HTTP para conectarse a un gestor de colas al que esté autorizado para conectarse.

MQHTTP40401: No se ha podido encontrar el destino *destination_name*

El destino especificado en el URI de solicitud HTTP no puede ser encontrado por el puente WebSphere MQ para HTTP.

Explicación
El destino especificado en el URI de solicitud HTTP no puede ser encontrado por el puente WebSphere MQ para HTTP.

Código de estado HTTP
`404 Not found`

Respuesta del programador
Compruebe que el destino especificado en el URI de solicitud HTTP existe o especifique un destino alternativo.

MQHTTP40501: Método *method_name* no permitido

El método especificado en la solicitud HTTP no está soportado por el puente WebSphere MQ para HTTP.

Explicación
El método especificado en la solicitud HTTP no está soportado por el puente WebSphere MQ para HTTP.

Código de estado HTTP
`405 Method not allowed`

Respuesta del programador
Cambie el método especificado en la solicitud HTTP por uno que esté soportado por el puente WebSphere MQ para HTTP.

MQHTTP41301: El mensaje que se estaba enviando era demasiado grande para el destino

El destino especificado en el URI de solicitud HTTP POST no puede aceptar mensajes que sean tan largos como el mensaje especificado en la solicitud HTTP.

Explicación
El destino especificado en el URI de solicitud HTTP POST no puede aceptar mensajes que sean tan largos como el mensaje especificado en la solicitud HTTP.

Código de estado HTTP
`413 Request entity too large`

Respuesta del programador
Reduzca el tamaño del mensaje especificado en la solicitud HTTP. De forma alternativa, especifique un destino que pueda soportar mensajes de la longitud deseada.

MQHTTP41501: El juego de caracteres de tipo de soporte no está soportado

El juego de caracteres especificado en el campo de cabecera `Content-Type` no está soportado por el puente WebSphere MQ para HTTP.

Explicación
El juego de caracteres especificado en el campo de cabecera `Content-Type` no está soportado por el puente WebSphere MQ para HTTP.

Código de estado HTTP
`415 Unsupported media type`

Respuesta del programador
Cambie el juego de caracteres del campo de cabecera `Content-Type` por uno que esté soportado por el puente WebSphere MQ para HTTP.

MQHTTP41502: Media-type *media-type* no está soportado ...

El tipo de soporte especificado en la solicitud HTTP no está soportado por el puente de WebSphere MQ para HTTP para el verbo HTTP especificado.

Explicación
El tipo de soporte especificado en la solicitud HTTP no está soportado por el puente de WebSphere MQ para HTTP para el verbo HTTP especificado.

Código de estado HTTP
`415 Unsupported media type`

Respuesta del programador
Cambie el tipo de soporte especificado en la solicitud HTTP por uno que esté soportado por WebSphere MQ Bridge for HTTP para el verbo HTTP especificado.

MQHTTP41503: Media-type *media-type* no está soportado ...

El tipo de soporte especificado en la solicitud HTTP no está soportado por el puente WebSphere MQ para HTTP para el campo de cabecera `x-msg-class` especificado.

Explicación
El tipo de soporte especificado en la solicitud HTTP no está soportado por el puente WebSphere MQ para HTTP para el campo de cabecera `x-msg-class` especificado.

Código de estado HTTP
`415 Unsupported media type`

Respuesta del programador
Cambie el tipo de soporte especificado en la solicitud HTTP por uno que esté soportado por WebSphere MQ Bridge for HTTP para el campo de cabecera `x-msg-class` especificado.

MQHTTP41701: La cabecera HTTP `Expect` no está soportada

El puente WebSphere MQ para HTTP no da soporte al campo de cabecera `Expect` .

Explicación
La cabecera `Expect` se ha especificado en una solicitud HTTP. El puente WebSphere MQ para HTTP no da soporte al campo de cabecera `Expect` .

Código de estado HTTP
`417 Expectation failed`

Respuesta del programador
Elimine la cabecera `Expect` de la solicitud HTTP.

MQHTTP50001: Se ha producido un problema inesperado ...

Se ha producido un error en el puente WebSphere MQ para HTTP.

Explicación
Se ha producido un error en el puente WebSphere MQ para HTTP.

Código de estado HTTP
`500 Internal server error`

Respuesta del programador
Póngase en contacto con el administrador del sistema de WebSphere MQ Bridge for HTTP.

MQHTTP50201: Se ha producido un error entre el puente de WebSphere MQ para HTTP y el gestor de colas

Se ha producido un error entre el puente WebSphere MQ para HTTP y el gestor de colas

Explicación
Se ha producido un error entre el puente WebSphere MQ para HTTP y el gestor de colas

Código de estado HTTP
`502 Bad Gateway`

Respuesta del programador
Póngase en contacto con el administrador del sistema de WebSphere MQ Bridge for HTTP.

MQHTTP50401: Se ha agotado el tiempo de espera de recuperación de mensajes

No se ha devuelto ningún mensaje que coincida con los parámetros de solicitud especificados en un HTTP **GET** o HTTP **DELETE** en el periodo de tiempo de espera.

Explicación
No se ha devuelto ningún mensaje que coincida con los parámetros de solicitud especificados en un HTTP **GET** o HTTP **DELETE** en el periodo de tiempo de espera. El código de retorno indica que no había ningún mensaje adecuado disponible en ningún momento durante la vida de la solicitud HTTP.

Código de estado HTTP
`504 Gateway timeout`

Respuesta del programador
Si se esperaba un mensaje, compruebe los campos de cabecera de la solicitud HTTP como `x-msg-correlId` y `x-msg-msgid`. Compruebe que el destino especificado en el URI de solicitud HTTP sea correcto. Intente ampliar el tiempo de espera de la solicitud HTTP utilizando el campo de cabecera `x-msg-wait` .

MQHTTP50501: HTTP 1.1 y superior ...

El protocolo HTTP utilizado en la solicitud HTTP no está soportado por el puente WebSphere MQ para HTTP.

Explicación
El protocolo HTTP utilizado en la solicitud HTTP no está soportado por el puente WebSphere MQ para HTTP.

Código de estado HTTP
`505 HTTP version not supported`

Respuesta del programador
Cambie la solicitud HTTP para utilizar el protocolo HTTP V1.1 o superior.

Tipos de mensajes y correlaciones de mensajes para WebSphere Bridge for HTTP

El puente WebSphere MQ para HTTP da soporte a cuatro clases de mensajes, TEXT, BYTES, STREAM y MAP. Las clases de mensajes se correlacionan con tipos de mensajes JMS y HTTP `Content-Type`.

HTTP POST

El tipo de mensaje que llega al destino depende del valor de la cabecera `x-msg-class` o del `Content-Type` de la solicitud HTTP. Tabla 604 en la página 1249 muestra el tipo HTTP `Content-Type` que corresponde a cada `x-msg-class`. Se puede utilizar cualquiera de los campos para establecer el tipo de mensaje y el formato de mensaje. Si se establecen ambos campos y se establecen de forma incoherente, se devuelve un `Bad Request exception` (`HTTP 400, MQHTTP20004`).

Tabla 604. Correlación entre `x-msg-class` y HTTP `Content-Type`

clase-mensaje-x	**Tipo de contenido HTTP**
BYTES	`application/octet-stream` `application/xml`
TEXT	`text/*`
MAP	`application/x-www-form-urlencoded` `application/xml` (opcional)

Tabla 604. Correlación entre x-msg-class y HTTP Content-Type (continuación)	
clase-mensaje-x	**Tipo de contenido HTTP**
STREAM	application/xml (opcional)

Si el tipo de mensaje JMS se establece en la cabecera MQRFH2 , se correlaciona de acuerdo con Tabla 605 en la página 1250.

Tabla 605. Correlación entre los tipos de mensaje x-msg-class y JMS.	
clase-mensaje-x	**Tipo de mensajes JMS**
BYTES	jms_bytes
TEXT	jms_text
MAP	jms_map
STREAM	jms_stream

El tipo de mensaje JMS siempre se establece para una clase de mensaje MAP o STREAM. No siempre se establece para una clase de mensaje de BYTES o TEXT. Si se va a crear un MQRFH2 para la solicitud, el tipo de mensaje JMS siempre se establece. De lo contrario, si no se crea ningún MQRFH2 , no se establece ningún tipo de mensaje JMS. Se crea un MQRFH2 si las propiedades de usuario se establecen en la solicitud, utilizando la cabecera x-msg-usr .

Si se establece el tipo de mensaje JMS, el formato de mensaje se establece en MQFMT_NONE, consulte Tabla 607 en la página 1250:

Tabla 606. Correlación entre el formato de mensaje x-msg-class y WebSphere MQ		
clase-mensaje-x	**Formato de mensaje con MQRFH2 presente en el mensaje**	**Formato de mensaje con *no* MQRFH2 presente en el mensaje**
BYTES	MQFMT_NONE	MQFMT_NONE
TEXT	MQFMT_NONE	MQFMT_STRING
MAP	MQFMT_NONE	No es posible
STREAM	MQFMT_NONE	No es posible

HTTP GET o DELETE

El tipo de mensaje o formato recuperado determina el valor de la cabecera x-msg-class y el Content-Type de la respuesta HTTP. La cabecera x-msg-class sólo se devuelve si se solicita en una solicitud x-msg-headers .

Tabla 607 en la página 1250 describe las correlaciones entre x-msg-class y Content-Type, y el tipo de mensaje recuperado de la cola o tema.

Tabla 607. Correlación de tipos de mensaje con x-msg-class y Content-Type			
Formato de mensaje	**Tipo de mensajes JMS**	**clase-mensaje-x**	**Content-Type**
Cualquier cosa excepto MQFMT_STRING	Ninguna	BYTES	application/octet-stream
MQFMT_STRING	Ninguna	TEXT	text/plain
MQFMT_NONE	jms_bytes	BYTES	application/octet-stream

Tabla 607. Correlación de tipos de mensaje con `x-msg-class` y `Content-Type` (continuación)			
Formato de mensaje	**Tipo de mensajes JMS**	**`clase-mensaje-x`**	**`Content-Type`**
`MQFMT_NONE`	`jms_text`	`TEXT`	`text/plain`
`MQFMT_NONE`	`jms_map`	`MAP`	`application/xml`
`MQFMT_NONE`	`jms_stream`	`STREAM`	`application/xml`

Serialización de clases de mensajes MAP y STREAM

Las clases de mensajes `MAP` y `STREAM` se serializan de nuevo en el cliente en la respuesta HTTP de la misma forma que un mensaje se serializa en una cola.

Para `MAP`, el nombre XML, el tipo y los tripletes de valores se codifican como:

```
<map>
    <elt name="elementname1" dt="datatype1">value1</elt>
    <elt name="elementname2" dt="datatype2">value2</elt>
    ...
</map>
```

`STREAM` es como `MAP`, pero no tiene nombres de elemento:

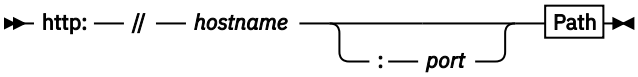
```
<stream>
    <elt dt="datatype1">value1</elt>
    <elt dt="datatype2">value2</elt>
    ...
</stream>
```

Nota: `datatype` es uno de los tipos de datos definidos para definir propiedades definidas por el usuario y listados en “usr: cabecera de entidad HTTP x-msg-usr” en la página 1239. El atributo `dt="string"` se omite para los elementos de serie porque el tipo de datos predeterminado es `string`.

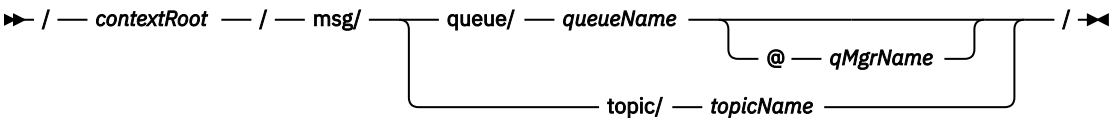
Formato de URI

URI interceptados por el puente WebSphere MQ para HTTP.

Syntax



Path



Nota:

1. If a question mark (?) is used it must be substituted with `%3f`. For example, `orange?topic` should be specified as `orange%3ftopic`.
2. @*qMgrName* is only valid on an HTTP **POST**

Descripción

Despliegue el puente WebSphere MQ para el servlet HTTP en el servidor de aplicaciones JEE con una raíz de contexto de *contextRoot*. Solicitudes para

```
http://hostname:port/context_root/msg/queue/queueName@qMgrName
```

y

```
http://hostname:port/context_root/msg/topic/topicString
```

se interceptan mediante el puente WebSphere MQ para HTTP.

Las clases e interfaces .NET deIBM WebSphere MQ

Las clases e interfaces .NET de IBM WebSphere MQ se listan alfabéticamente. Se describen las propiedades, métodos y constructores.

Clase MQAsyncStatus .NET

Utilice MQAsyncStatus para consultar sobre el estado de la actividad MQI anterior; por ejemplo, consultar sobre el éxito de las operaciones de colocación asíncrona anteriores. MQAsyncStatus encapsula las características de la estructura de datos de MQSTS .

Clase

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.MQAsyncStatus
```

```
public class IBM.WMQ.MQAsyncStatus extends IBM.WMQ.MQBaseObject;
```

- “Propiedades” en la página 1252
- “Constructores” en la página 1253

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

public static int CompCode {get;}

El código de terminación del primer error o aviso.

public static int Reason {get;}

El código de razón del primer error o aviso.

public static int PutSuccessCount {get;}

Número de llamadas put de MQI asíncronas satisfactorias.

public static int PutWarningCount {get;}

Número de llamadas put de MQI asíncronas que se han realizado correctamente con un aviso.

public static int PutFailureCount {get;}

Número de llamadas put de MQI asíncronas fallidas.

public static int ObjectType {get;}

El tipo de objeto para el primer error. Son posibles los siguientes valores:

- MQC.MQOT_ALIAS_Q
- MQC.MQOT_LOCAL_Q
- MQC.MQOT_MODEL_Q
- MQC.MQOT_Q

- MQC.MQOT_REMOTE_Q
- MQC.MQOT_TOPIC
- 0, lo que significa que no se devuelve ningún objeto

public static string ObjectName {get;}

El nombre de objeto.

public static string ObjectQMgrName {get;}

El nombre del gestor de colas de objetos.

public static string ResolvedObjectName {get;}

El nombre de objeto resuelto.

public static string ResolvedObjectQMgrName {get;}

El nombre del gestor de colas de objeto resuelto.

Constructores

public MQAsyncStatus() throws MQException;

Método constructor, construye un objeto con campos inicializados en cero o en blanco según corresponda.

Clase MQAuthenticationInformationRecord .NET

Utilice MQAuthenticationInformationRecord para especificar información sobre un autenticador que se va a utilizar en una conexión de cliente SSL de WebSphere MQ . MQAuthenticationInformationRecord encapsula un registro de información de autenticación, MQAIR.

Clase

```
System.Object
    │
    └─ IBM.WMQ.MQAuthenticationInformationRecord
```

```
public class IBM.WMQ.MQAuthenticationInformationRecord extends System.Object;
```


- "Propiedades" en la página 1253
- "Constructores" en la página 1254

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

public long Version {get; set;}

Número de versión de la estructura.

public long AuthInfoType {get; set;}

El tipo de información de autenticación. Este atributo debe establecerse en uno de los valores siguientes:

- OCSP -La comprobación de estado de revocación de certificados se realiza utilizando OCSP.
- CRLLDAP -La comprobación de estado de revocación de certificados se realiza utilizando listas de revocación de certificados en servidores LDAP.

public string AuthInfoConnName {get; set;}

El nombre DNS o la dirección IP del host en el que se ejecuta el servidor LDAP, con un número de puerto opcional. Esta palabra clave es necesaria.

public string LDAPPassword {get; set;}

La contraseña asociada con el nombre distinguido del usuario que está accediendo al servidor LDAP. Esta propiedad sólo se aplica cuando **AuthInfoType** se establece en CRLLDAP.

public string LDAPUserName {get; set;}

Nombre distinguido del usuario que está accediendo al servidor LDAP. Cuando se establece esta propiedad, LDAPUserNameLength y LDAPUserNamePtr se establecen automáticamente correctamente. Esta propiedad sólo se aplica cuando AuthInfoTipo está establecido en CRLLDAP.

public string OCSPResponderURL {get; set;}

El URL en el que se puede establecer contacto con el programa de respuesta OCSP. Esta propiedad sólo se aplica cuando AuthInfoTipo está establecido en OCSP

Este campo distingue entre mayúsculas y minúsculas. Debe empezar con la serie http:// en minúsculas. El resto del URL puede ser sensible a las mayúsculas y minúsculas, en función de la implementación del servidor OCSP.

Constructores

MQAuthenticationInformationRecord();

Clase MQDestination .NET

Utilice MQDestination para acceder a métodos comunes a MQQueue y MQTopic. MQDestination es una clase base abstracta y no se puede instanciar.

Clase

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.MQManagedObject
                └ IBM.WMQ.MQDestination
                    └ IBM.WMQ.MQQueue
                    └ IBM.WMQ.MQTopic
```

public class IBM.WMQ.MQDestination extends IBM.WMQ.MQManagedObject;

- "Propiedades" en la página 1254
- "Métodos" en la página 1255
- "Constructores" en la página 1256

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

public DateTime CreationDateTime {get;}

Fecha y hora en que se ha creado la cola o el tema. Originalmente contenida en MQQueue, esta propiedad se ha movido a la clase MQDestination base.

No hay ningún valor predeterminado.

public int DestinationType {get;}

Valor entero que describe el tipo de destino que se está utilizando. Inicializado desde el constructor de subclases, MQQueue o MQTopic, este valor puede tomar uno de estos valores:

- MQOT_Q
- MQOT_TOPIC

No hay ningún valor predeterminado.

Métodos

public void Get(MQMessage *message*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*, int *MaxMsgSize*);

Genera MQException.

Obtiene un mensaje de una cola si el destino es un objeto MQQueue o de un tema si el destino es un objeto MQTopic , utilizando una instancia predeterminada de MQGetMessageOptions para realizar la obtención.

Si la obtención falla, el objeto MQMessage no se modifica. Si se ejecuta correctamente, el descriptor de mensaje y las partes de datos de mensaje del MQMessage se sustituyen por el descriptor de mensaje y los datos de mensaje del mensaje de entrada.

Todas las llamadas a WebSphere MQ desde un MQQueueManager determinado son síncronas. Por lo tanto, si realiza una operación get con espera, todas las demás hebras que utilizan el mismo MQQueueManager no pueden realizar más llamadas WebSphere MQ hasta que se realice la llamada Get. Si necesita varias hebras para acceder a WebSphere MQ simultáneamente, cada hebra debe crear su propio objeto MQQueueManager .

message
Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada MessageId y CorrelationId estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón MQRC_BACKED_OUT después de una reconexión satisfactoria, para los mensajes recibidos en MQGM_SYNCPOINT.

getMessageOptions
Opciones que controlan la acción de la obtención.

El uso de la opción MQC.MQGMO_CONVERT puede dar como resultado una excepción con el código de razón MQC.MQRC_CONVERTED_STRING_TOO_BIG al convertir de códigos de caracteres de un solo byte a códigos de doble byte. En este caso, el mensaje se copia en el almacenamiento intermedio sin conversión.

Si no se especifica *getMessageOptions* , la opción de mensaje utilizada es MQGMO_NOWAIT.

Si utiliza la opción MQGMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

MaxMsgSize
El mensaje más grande que este objeto de mensaje va a recibir. Si el mensaje en la cola es mayor que este tamaño, se produce una de las dos cosas siguientes:

- Si el distintivo MQGMO_ACCEPT_TRUNCATED_MSG se establece en el objeto MQGetMessageOptions , el mensaje se rellena con la mayor cantidad posible de datos de mensaje. Se genera una excepción con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_ACCEPTED .
- Si el distintivo MQGMO_ACCEPT_TRUNCATED_MSG no está establecido, el mensaje se deja en la cola. Se genera una excepción con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_FAILED .

Si no se especifica *MaxMsgSize* , se recupera todo el mensaje.

public void Put(MQMessage *message*);
public void Put(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);

Genera MQException.

Coloca un mensaje en una cola si el destino es un objeto MQQueue o publica un mensaje en un tema si el destino es un objeto MQTopic .

Las modificaciones en el objeto MQMessage después de que se haya realizado la llamada Put no afectan al mensaje real en el tema de publicación o cola WebSphere MQ .

Put actualiza las propiedades MessageId y CorrelationId del objeto MQMessage y no borra los datos del mensaje. Las llamadas Put o Get adicionales hacen referencia a la información actualizada en el objeto MQMessage . Por ejemplo, en el siguiente fragmento de código, el primer mensaje contiene a y el segundo ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

message
Un objeto MQMessage que contiene los datos del descriptor de mensaje y el mensaje que se va a enviar. El descriptor de mensaje se puede modificar como consecuencia de este método. Los valores del descriptor de mensaje inmediatamente después de la finalización de este método son los valores que se colocaron en la cola o se publicaron en el tema.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- MQRC_CALL_INTERRUPTED si la conexión se interrumpe al ejecutar una llamada Put en un mensaje persistente y la reconexión se realiza correctamente.
- MQRC_NONE si la conexión es satisfactoria al ejecutar una llamada Put en un mensaje no persistente (consulte Recuperación de aplicaciones).

putMessageOptions
Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptons .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

Nota: Por motivos de simplicidad y rendimiento, si desea transferir un único mensaje a una cola, utilice el objeto MQQueueManager.Put . Debe tener un objeto MQQueue para esto.

Constructores
MQDestination es una clase base abstracta y no se puede instanciar. Acceda a destinos utilizando constructores MQQueue y MQTopic , o utilizando MQQueueManager.AccessQueue y MQQueueManager.AccessTopic methods.

Clase MQEnvironment .NET

Utilice MQEnvironment para controlar cómo se llama al constructor MQQueueManager y para seleccionar una conexión de cliente MQI de WebSphere MQ . La clase MQEnvironment contiene propiedades que controlan el comportamiento de WebSphere MQ.

Clase

```
System.Object
    │
    └─ IBM.WMQ.MQEnvironment
```

```
public class IBM.WMQ.MQEnvironment extends System.Object;
```


Propiedades-sólo cliente

Prueba de `MQException` que se genera al obtener las propiedades.

`public static int CertificateValPolicy {get; set;}`

Establezca qué política de validación de certificados SSL/TLS se utiliza para validar los certificados digitales recibidos de los sistemas asociados remotos. Los valores válidos son:

- MQC.CERTIFICATE_VALIDATION_POLICY_ANY
- MQC.CERTIFICATE_VALIDATION_POLICY_RFC5280

`public static ArrayList EncryptionPolicySuiteB {get; set;}`

Establezca el nivel de criptografía compatible con Suite B. Los valores válidos son:

- MQC.MQ_SUITE_B_NONE -Este es el valor predeterminado.
- MQC.MQ_SUITE_B_128_BIT
- MQC.MQ_SUITE_B_192_BIT

`public static string Channel {get; set;}`
El nombre del canal para conectarse al gestor de colas de destino. *Debe* establecer la propiedad de canal antes de crear una instancia de `MQQueueManager` en modalidad de cliente.

`public static int FipsRequired {get; set;}`

Especifique `MQC.MQSSL_FIPS_YES` para utilizar sólo algoritmos certificados por FIPS si el cifrado se lleva a cabo en WebSphere MQ. El valor predeterminado es `MQC.MQSSL_FIPS_NO`.

Si el hardware criptográfico está configurado, los módulos criptográficos utilizados son los proporcionados por el producto de hardware. En función del hardware en uso, es posible que no estén certificados por FIPS a un nivel determinado.

`public static string Hostname {get; set;}`
El nombre de host TCP/IP del sistema en el que reside el servidor WebSphere MQ . Si no se establece el nombre de host y no se establecen propiedades de alteración temporal, se utiliza la modalidad de enlaces de servidor para conectarse al gestor de colas local.

`public static int Port {get; set;}`
El puerto al que conectarse. Este es el puerto en el que el servidor de WebSphere MQ está a la escucha de solicitudes de conexión entrantes. El valor predeterminado es 1414.

`public static string SSLCipherSpec {get; set;}`
Establezca `SSLCipherSpec` en el valor de CipherSpec establecido en el canal SVRCONN para habilitar SSL para la conexión. El valor predeterminado es Null y SSL no está habilitado para la conexión.

`public static string sslPeerName {get; set;}`

Un patrón de nombre distinguido. Si se establece `sslCipherSpec` , esta variable se puede utilizar para asegurarse de que se utiliza el gestor de colas correcto. Si se establece en nulo (valor predeterminado), el DN del gestor de colas no se realiza. `sslPeerNombre` se ignora si `sslCipherSpec` es nulo.

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

public static ArrayList HdrCompList {get; set;}
Lista de compresión de datos de cabecera

public static int KeyResetCount {get; set;}
Indica el número de bytes no cifrados enviados y recibidos dentro de una conversación SSL antes de que se renegocie la clave secreta.

public static ArrayList MQAIRArray {get; set;}
Una matriz de objetos `MQAuthenticationInformationRecord` .

public static ArrayList MsgCompList {get; set;}
Lista de compresión de datos de mensaje

public static string Password {get; set;}
La contraseña que se va a autenticar. La contraseña a la que se hace referencia desde la estructura `MQCSP` se rellena estableciendo esta propiedad de contraseña.

public static string ReceiveExit {get; set;}
Una salida de recepción le permite examinar y modificar los datos recibidos de un gestor de colas. Normalmente se utiliza con una salida de emisión correspondiente en el gestor de colas. Si ReceiveExit se establece en nulo, no se llama a ninguna salida de recepción.

public static string ReceiveUserData {get; set;}
Los datos de usuario asociados a una salida de recepción. Limitado a 32 caracteres.

public static string SecurityExit {get; set;}
Una salida de seguridad le permite personalizar los flujos de seguridad que se producen cuando se intenta conectarse a un gestor de colas. Si `SecurityExit` se establece en nulo, no se llama a ninguna salida de seguridad.

public static string SecurityUserData {get; set;}
Los datos de usuario asociados a una salida de seguridad. Limitado a 32 caracteres.

public static string SendExit {get; set;}
Una salida de emisión le permite examinar o modificar los datos enviados a un gestor de colas. Normalmente se utiliza con una salida de recepción correspondiente en el gestor de colas. Si `SendExit` se establece en nulo, no se llama a ninguna salida de envío.

public static string SendUserData {get; set;}
Los datos de usuario asociados a una salida de emisión. Limitado a 32 caracteres.

public static string SharingConversations {get; set;}

El campo `SharingConversations` se utiliza en conexiones de aplicaciones .NET, cuando estas aplicaciones no utilizan una tabla de definición de canal de cliente (CCDT).

`SharingConversations` determina el número máximo de conversaciones que se pueden compartir en un socket asociado a esta conexión.

Un valor de 0 significa que el canal funciona como lo hacía antes de WebSphere MQ Versión 7.0, con respecto a la compartición de conversación, la lectura anticipada y la pulsación.

El campo se pasa en la tabla hash de propiedades como un `SHARING_CONVERSATIONS_PROPERTY`, al crear una instancia de un gestor de colas de WebSphere MQ .

Si no especifica `SharingConversations`, se utiliza un valor predeterminado de 10.

public static string SSLCryptoHardware {get; set;}

Establece el nombre de la serie de parámetro necesaria para configurar el hardware criptográfico presente en el sistema. `SSLCryptoHardware` se ignora si `sslCipherSpec` es nulo.

public static string SSLKeyRepository {get; set;}

Establezca el nombre de archivo completo del repositorio de claves.

Si `SSLKeyRepository` se establece en nulo (valor predeterminado), se utiliza la variable de entorno `MQSSLKEYR` del certificado para localizar el repositorio de claves. `SSLCryptoHardware` se ignora si `sslCipherSpec` es nulo.

Nota: La extensión `.kdb` es una parte obligatoria del nombre de archivo, pero no se incluye como parte del valor del parámetro. El directorio que especifique debe existir. WebSphere MQ crea el archivo la primera vez que accede al nuevo repositorio de claves, a menos que el archivo ya exista.

`public static string UserId {get; set;}`

El ID de usuario que se va a autenticar. El ID de usuario al que se hace referencia desde la estructura `MQCSP` se rellena estableciendo `UserId`. Autentique `UserId` utilizando una salida de API o de seguridad.

Constructores

`public MQEnvironment()`

Clase `MQException` .NET

Utilice `MQException` para averiguar el código de terminación y de razón de una función de WebSphere MQ anómala. Se emite un `MQException` siempre que se produce un error de WebSphere MQ .

Clase

```
System.Object
    └ System.Exception
        └ System.ApplicationException
            └ IBM.WMQ.MQException
```

`public class IBM.WMQ.MQException extends System.ApplicationException;`

- “Propiedades” en la página 1259
- “Constructores” en la página 1259

Propiedades

`public int CompletionCode {get; set;}`
El código de terminación de WebSphere MQ asociado con el error. Los valores posibles son:

- `MQException.MQCC_OK`
- `MQException.MQCC_WARNING`
- `MQException.MQCC_FAILED`

`public int ReasonCode {get; set;}`
WebSphere MQ código de razón que describe el error.

Constructores

`public MQException(int completionCode, int reasonCode)`

`completionCode`
El código de terminación de WebSphere MQ .

`reasonCode`
El código de terminación de WebSphere MQ .

Clase `MQGetMessageOptions` .NET

Utilice `MQGetMessageOptions` para especificar cómo se recuperan los mensajes. Modifica el comportamiento de `MQDestination.Get`.

Clase

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.MQGetMessageOptions
```

```
public class IBM.WMQ.MQGetMessageOptions extends IBM.WMQ.MQBaseObject;
```


- "Propiedades" en la página 1260
- "Constructores" en la página 1262

Propiedades

Nota: El comportamiento de algunas de las opciones disponibles en esta clase depende del entorno en el que se utilizan. Estos elementos se marcan con un asterisco *.

Prueba de `MQException` que se genera al obtener las propiedades.

`public int GroupStatus {get;}*`
`GroupStatus` indica si el mensaje recuperado está en un grupo y si es el último del grupo. Los valores posibles son:

MQC.MQGS_LAST_MSG_IN_GROUP
El mensaje es el último o único mensaje del grupo.

MQC.MQGS_MSG_IN_GROUP
El mensaje está en un grupo, pero no es el último del grupo.

MQC.MQGS_NOT_IN_GROUP
El mensaje no está en un grupo.

`public int MatchOptions {get; set;}*`
`MatchOptions` determina cómo se selecciona un mensaje. Se pueden establecer las siguientes opciones de coincidencia:

MQC.MQMO_MATCH_CORREL_ID
ID de correlación que debe coincidir.

MQC.MQMO_MATCH_GROUP_ID
ID de grupo que debe coincidir.

MQC.MQMO_MATCH_MSG_ID
ID de mensaje que debe coincidir.

MQC.MQMO_MATCH_MSG_SEQ_NUMBER
Coincide con el número de secuencia de mensaje.

MQC.MQMO_NONE
No se requiere ninguna coincidencia.

`public int Options {get; set;}`
Las `Opciones` controlan la acción de `MQQueue.get`. Se puede especificar cualquiera de los valores siguientes. Si se necesita más de una opción, los valores se pueden añadir o combinar utilizando el operador OR a nivel de bit.

MQC.MQGMO_ACCEPT_TRUNCATED_MSG
Permitir truncamiento de datos de mensaje.

MQC.MQGMO_ALL_MSGS_AVAILABLE*
Recuperar mensajes de un grupo sólo cuando todos los mensajes del grupo están disponibles.

MQC.MQGMO_ALL_SEGMENTS_AVAILABLE*
Recuperar los segmentos de un mensaje lógico sólo cuando todos los segmentos del grupo están disponibles.

MQC.MQGMO_BROWSE_FIRST
Examinar desde el inicio de la cola.

MQC.MQGMO_BROWSE_MSG_UNDER_CURSOR*
Examine el mensaje bajo el cursor para examinar.

MQC.MQGMO_BROWSE_NEXT
Examinar desde la posición actual en la cola.

MQC.MQGMO_COMPLETE_MSG*
Recuperar sólo mensajes lógicos completos.

MQC.MQGMO_CONVERT
Solicite que los datos de aplicación se conviertan, para que se ajusten a los atributos `CharacterSet` y `Codificación` del `MQMessage`, antes de que los datos se copien en el almacenamiento intermedio de mensajes. Puesto que la conversión de datos también se aplica cuando los datos se recuperan del almacenamiento intermedio de mensajes, las aplicaciones no establecen esta opción.

El uso de esta opción puede causar problemas al convertir de juegos de caracteres de un solo byte a juegos de caracteres de doble byte. En su lugar, realice la conversión utilizando los métodos `readString`, `readLine`y `writeString` después de que se haya entregado el mensaje.

MQC.MQGMO_FAIL_IF_QUIESCING
Fallar si el gestor de colas se está desactivando temporalmente.

MQC.MQGMO_LOCK*
Bloquear el mensaje que se examina.

MQC.MQGMO_LOGICAL_ORDER*
Devuelve mensajes en grupos y segmentos de mensajes lógicos, en orden lógico.

Si utiliza la opción `MQGMO_LOGICAL_ORDER` en un cliente reconectable, el código de razón `MQRC_RECONNECT_INCOMPATIBLE` se devuelve a la aplicación.

MQC.MQGMO_MARK_SKIP_BACKOUT*
Permitir que se restituya una unidad de trabajo sin restablecer el mensaje en la cola.

MQC.MQGMO_MSG_UNDER_CURSOR
Obtener mensaje bajo el cursor para examinar.

MQC.MQGMO_NONE
No se ha especificado ninguna otra opción; todas las opciones asumen sus valores predeterminados.

MQC.MQGMO_NO_PROPERTIES
No se recupera ninguna propiedad del mensaje, excepto las propiedades contenidas en el descriptor de mensaje (o extensión).

MQC.MQGMO_NO_SYNCPOINT
Obtener mensaje sin control de punto de sincronización.

MQC.MQGMO_NO_WAIT
Vuelva inmediatamente si no hay ningún mensaje adecuado.

MQC.MQGMO_PROPERTIES_AS_Q_DEF
Recupere las propiedades de mensaje tal como las define el atributo `PropertyControl` de `MQQueue`. El acceso a las propiedades del mensaje en el descriptor de mensaje, o extensión, no se ve afectado por el atributo `PropertyControl` .

MQC.MQGMO_PROPERTIES_COMPATIBILITY
Recupere las propiedades de mensaje con el prefijo `mcd`, `jms`, `usro` `mqext`, en las cabeceras `MQRFH2` . Otras propiedades del mensaje, excepto las propiedades contenidas en el descriptor de mensaje, o extensión, se descartan.

MQC.MQGMO_PROPERTIES_FORCE_MQRFH2
Recuperar propiedades de mensaje, excepto las propiedades contenidas en el descriptor de mensaje, o extensión, en las cabeceras `MQRFH2` . Utilice

MQC.MQGMO_PROPERTIES_FORCE_MQRFH2 en aplicaciones que esperan recuperar propiedades pero que no se pueden cambiar para utilizar manejadores de mensajes.

MQC.MQGMO_PROPERTIES_IN_HANDLE
Recuperar propiedades de mensaje utilizando un MsgHandle.

MQC.MQGMO_SYNCPOINT
Obtenga el mensaje bajo control de punto de sincronización. El mensaje se marca como no disponible para otras aplicaciones, pero sólo se suprime de la cola cuando se confirma la unidad de trabajo. El mensaje vuelve a estar disponible si se restituye la unidad de trabajo.

MQC.MQGMO_SYNCPOINT_IF_PERSISTENT*
Obtener mensaje con control de punto de sincronización si el mensaje es persistente.

MQC.MQGMO_UNLOCK*
Desbloquear un mensaje bloqueado anteriormente.

MQC.MQGMO_WAIT
Espere a que llegue un mensaje.

public string ResolvedQueueName {get;}
El gestor de colas establece ResolvedQueueNombre en el nombre local de la cola de la que se ha recuperado el mensaje. ResolvedQueueNombre es diferente del nombre utilizado para abrir la cola si se ha abierto una cola alias o una cola modelo.

public char Segmentation {get;}*
Segmentación indica si puede permitir la segmentación para el mensaje recuperado. Los valores posibles son:

MQC.MQSEG_INHIBITED
No permitir segmentación.

MQC.MQSEG_ALLOWED
Permitir segmentación

public byte SegmentStatus {get;}*
SegmentStatus es un campo de salida que indica si el mensaje recuperado es un segmento de un mensaje lógico. Si el mensaje es un segmento, el distintivo indica si es el último segmento. Los valores posibles son:

MQC.MQSS_LAST_SEGMENT
El mensaje es el último o único segmento del mensaje lógico.

MQC.MQSS_NOT_A_SEGMENT
El mensaje no es un segmento.

MQC.MQSS_SEGMENT
El mensaje es un segmento, pero no es el último segmento del mensaje lógico.

public int WaitInterval {get; set;}
WaitInterval es el tiempo máximo en milisegundos que una llamada MQQueue.get espera a que llegue un mensaje adecuado. Utilice WaitInterval con MQC.MQGMO_WAIT. Establezca un valor de MQC.MQWI_UNLIMITED para esperar un tiempo ilimitado para un mensaje.

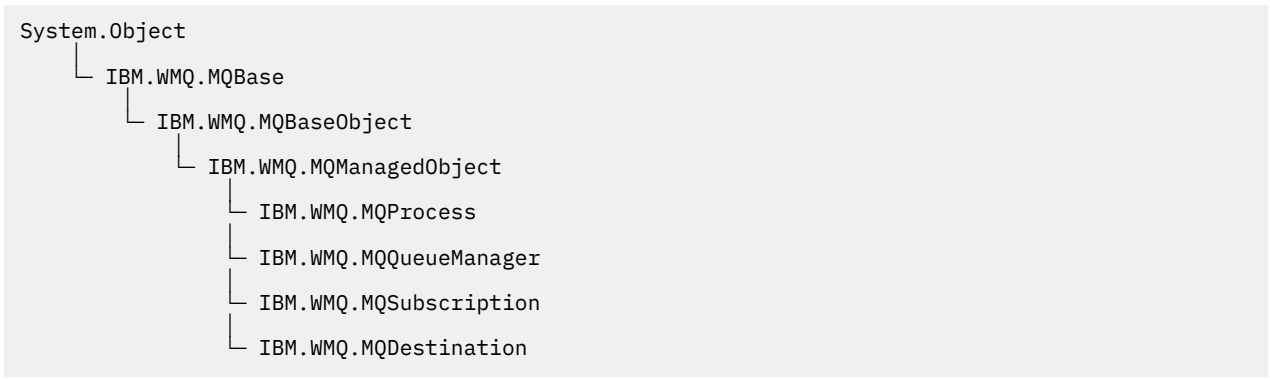
Constructores

public MQGetMessageOptions()
Construya un nuevo objeto MQGetMessageOptions con Options establecido en MQC.MQGMO_NO_WAIT, WaitInterval establecido en cero y ResolvedQueueName establecido en blanco.

Clase MQManagedObject .NET

Utilice MQManagedObject para consultar y establecer atributos de MQDestination, MQProcess, MQQueueManagery MQSubscription. MQManagedObject es una superclase de estas clases.

Clases



`public class IBM.WMQ.MQManagedObject extends IBM.WMQ.MQBaseObject;`

- "Propiedades" en la página 1263
- "Métodos" en la página 1264
- "Constructores" en la página 1265

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

`public string AlternateUserId {get; set;}`
El ID de usuario alternativo, si lo hay, establecido cuando se abrió el recurso. `AlternateUserID.set` se ignora cuando se emite para un objeto que está abierto. `AlternateUserId` no es válido para suscripciones.

`public int CloseOptions {get; set;}`

Establezca este atributo para controlar la forma en que se cierra el recurso. El valor predeterminado es `MQC.MQCO_NONE`. `MQC.MQCO_NONE` es el único valor permitido para todos los recursos que no sean colas dinámicas permanentes, colas dinámicas temporales, suscripciones y temas a los que acceden los objetos que las han creado.

Para colas y temas, se permiten los siguientes valores adicionales:

`MQC.MQCO_DELETE`
Suprima la cola si no hay mensajes.

`MQC.MQCO_DELETE_PURGE`
Suprima la cola, depurando los mensajes que haya en ella.

`MQC.MQCO_QUIESCE`
Solicite que se cierre la cola, recibiendo un aviso si queda algún mensaje (permitiendo que se recuperen antes del cierre final).

Para suscripciones, se permiten los siguientes valores adicionales:

`MQC.MQCO_KEEP_SUB`
La suscripción no se suprime. Esta opción sólo es válida si la suscripción original es duradera. `MQC.MQCO_KEEP_SUB` es el valor predeterminado para un tema duradero.

`MQC.MQCO_REMOVE_SUB`
La suscripción se suprime. `MQC.MQCO_REMOVE_SUB` es el valor predeterminado para un tema no duradero y no gestionado.

`MQC.MQCO_PURGE_SUB`
La suscripción se suprime. `MQC.MQCO_PURGE_SUB` es el valor predeterminado para un tema gestionado no duradero.

`public MQQueueManager ConnectionReference {get;}`
El gestor de colas al que pertenece este recurso.

public string MQDescription {get;}
La descripción del recurso tal como la retiene el gestor de colas. `MQDescription` devuelve una serie vacía para suscripciones y temas.

public boolean IsOpen {get;}
Indica si el recurso está abierto actualmente.

public string Name {get;}
Nombre del recurso. El nombre es el proporcionado en el método de acceso o el asignado por el gestor de colas para una cola dinámica.

public int OpenOptions {get; set;}
OpenOptions se establecen cuando se abre un objeto WebSphere MQ . El método `OpenOptions.set` se ignora y no provoca un error. Las suscripciones no tienen `OpenOptions`.

Métodos

public virtual void Close();
Genera `MQException`.

Cierra el objeto. No se permiten más operaciones en este recurso después de llamar a `Close`. Para cambiar el comportamiento del método `Close` , establezca el atributo `closeOptions` .

public string GetAttributeString(int *selector*, int *length*);
Genera `MQException`.

Obtiene una serie de atributo.

selector
Entero que indica qué atributo se está consultando.

length
Entero que indica la longitud de la serie necesaria.

public void Inquire(int[] *selectors*, int[] *intAttrs*, byte[] *charAttrs*);
Genera `MQException`.

Devuelve una matriz de enteros y un conjunto de series de caracteres que contienen los atributos de una cola, un proceso o un gestor de colas. Los atributos que se van a consultar se especifican en la matriz de selectores.

Nota: Muchos de los atributos más comunes se pueden consultar utilizando los métodos `Get` definidos en `MQManagedObject`, `MQQueue` y `MQQueueManager`.

selectors
Matriz de enteros que identifica los atributos con los valores que se van a consultar.

intAttrs
La matriz en la que se devuelven los valores de atributo de entero. Los valores de atributos enteros se devuelven en el mismo orden que los selectores de atributos enteros de la matriz de selectores.

charAttrs
El almacenamiento intermedio en el que se devuelven los atributos de carácter, concatenado. Los atributos de caracteres se devuelven en el mismo orden que los selectores de atributos de caracteres de la matriz de selectores. La longitud de cada serie de atributo es fija para cada atributo.

public void Set(int[] *selectors*, int[] *intAttrs*, byte[] *charAttrs*);
Genera `MQException`.

Establece los atributos definidos en el vector de selectores. Los atributos que se van a establecer se especifican en la matriz de selectores.

selectors
Matriz de enteros que identifica los atributos con valores que se van a establecer.

intAttrs
La matriz de valores de atributo de entero que se van a establecer. Estos valores deben estar en el mismo orden que los selectores de atributos enteros de la matriz de selectores.

charAttrs
El almacenamiento intermedio en el que se concatenan los atributos de carácter que se van a establecer. Estos valores deben estar en el mismo orden que los selectores de atributos de caracteres de la matriz de selectores. La longitud de cada atributo de carácter es fija.

public void SetAttributeString(int *selector*, string *value*, int *length*);
Genera `MQException`.

Establece una serie de atributo.

selector
Entero que indica qué atributo se está definiendo.

value
La serie que se va a establecer como valor de atributo.

length
Entero que indica la longitud de la serie necesaria.

Constructores

protected MQManagedObject()
Método constructor. Este objeto es una clase base abstracta que no se puede instanciar por sí misma.

Clase MQMessage .NET

Utilice `MQMessage` para acceder al descriptor de mensaje y a los datos de un mensaje de WebSphere MQ . `MQMessage` encapsula un mensaje de WebSphere MQ .

Clase

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.MQMessage
```

```
public class IBM.WMQ.MQMessage extends IBM.WMQ.MQBaseObject;
```

Cree un objeto `MQMessage` y, a continuación, utilice los métodos Read y `Write` para transferir datos entre el mensaje y otros objetos de la aplicación. Enviar y recibir objetos `MQMessage` utilizando los métodos `Put` y `Get` de las clases `MQDestination`, `MQQueue` y `MQTopic` .

Obtenga y establezca las propiedades del descriptor de mensaje utilizando las propiedades de `MQMessage`. Establezca y obtenga propiedades de mensajes ampliadas utilizando los métodos `SetProperty` y `GetProperty` .

- "Propiedades" en la página 1265
- "Métodos de mensaje Read y Write" en la página 1271
- "Métodos de almacenamiento intermedio" en la página 1274
- "Métodos de propiedad" en la página 1274
- "Constructores" en la página 1276

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

public string AccountingToken {get; set;}
Parte del contexto de identidad del mensaje; ayuda a una aplicación a cobrar por el trabajo realizado como resultado del mensaje. El valor predeterminado es MQC.MQACT_NONE.

public string ApplicationIdData {get; set;}
Parte del contexto de identidad del mensaje. ApplicationIdDatos es información definida por la suite de aplicaciones y se puede utilizar para proporcionar información adicional sobre el mensaje o su originador. El valor predeterminado es "".

public string ApplicationOriginData {get; set;}
Información definida por la aplicación que se puede utilizar para proporcionar información adicional sobre el origen del mensaje. El valor predeterminado es "".

public int BackoutCount {get;}
Recuento del número de veces que una llamada MQQueue.Get ha devuelto y restituido previamente el mensaje como parte de una unidad de trabajo. El valor predeterminado es cero.

public int CharacterSet {get; set;}

El identificador de juego de caracteres codificado de datos de caracteres en el mensaje.

Establezca CharacterSet para identificar el juego de caracteres de los datos de caracteres en el mensaje. Obtenga CharacterSet para averiguar en qué juego de caracteres se ha utilizado para codificar datos de caracteres en el mensaje.

Las aplicaciones .NET siempre se ejecutan en Unicode, mientras que en otros entornos las aplicaciones se ejecutan en el mismo juego de caracteres bajo el que se ejecuta el gestor de colas.

Los métodos ReadString y ReadLine convierten los datos de caracteres del mensaje a Unicode automáticamente.

El método WriteString se convierte de Unicode al juego de caracteres codificado en CharacterSet. Si CharacterSet se establece en su valor predeterminado, MQC.MQCCSI_Q_MGR, que es 0, no tiene lugar ninguna conversión y CharacterSet se establece en 1200. Si establece CharacterSet en algún otro valor, WriteString se convierte de Unicode al valor alternativo.

Nota: Otros métodos de lectura y escritura no utilizan CharacterSet.

- ReadChar y WriteChar leen y escriben un carácter Unicode en y desde el almacenamiento intermedio de mensajes sin conversión.
- ReadUTF y WriteUTF se convierten entre una serie Unicode en la aplicación y una serie UTF-8 , con el prefijo de un campo de longitud de 2 bytes, en el almacenamiento intermedio de mensajes.
- Los métodos de bytes transfieren bytes entre la aplicación y el almacenamiento intermedio de mensajes sin modificación.

public byte[] CorrelationId {get; set;}

- Para una llamada MQQueue.Get , el identificador de correlación del mensaje que se va a recuperar. El gestor de colas devuelve el primer mensaje con un identificador de mensaje y un identificador de correlación que coinciden con los campos del descriptor de mensaje. El valor predeterminado, MQC.MQCI_NONE, ayuda a que coincida cualquier identificador de correlación.
- Para una llamada MQQueue.Put , el identificador de correlación a establecer.

public int DataLength {get;}
Número de bytes de datos de mensaje que quedan por leer.

public int DataOffset {get; set;}
La posición actual del cursor dentro de los datos del mensaje. Las lecturas y grabaciones entran en vigor en la posición actual.

public int Encoding {get; set;}
Representación utilizada para valores numéricos en los datos de mensaje de aplicación. Codificación se aplica a datos binarios, decimales empaquetados y de coma flotante. El comportamiento de los métodos de lectura y escritura para estos formatos numéricos se modifica en consecuencia. Construya un valor para el campo de codificación añadiendo un valor de cada una

de estas tres secciones. De forma alternativa, construya el valor que combina los valores de cada una de las tres secciones utilizando el operador OR a nivel de bit.

1. Entero binario

   MQC.MQENC_INTEGER_NORMAL
   Enteros big-endian.

   MQC.MQENC_INTEGER_REVERSED
   Enteros little-endian, como se utiliza en la arquitectura Intel.

2. Decimal empaquetado

   MQC.MQENC_DECIMAL_NORMAL
   Big-endian packed-decimal, tal como lo utiliza z/OS.

   MQC.MQENC_DECIMAL_REVERSED
   Decimal empaquetado little-endian.

3. Coma flotante

   MQC.MQENC_FLOAT_IEEE_NORMAL
   Flotadores IEEE de big-endian.

   MQC.MQENC_FLOAT_IEEE_REVERSED
   Little-endian IEEE flotantes, como se utiliza en la arquitectura Intel.

   MQC.MQENC_FLOAT_S390
   z/OS formatear puntos flotantes.

El valor predeterminado es:

```
MQC.MQENC_INTEGER_REVERSED |
MQC.MQENC_DECIMAL_REVERSED |
MQC.MQENC_FLOAT_IEEE_REVERSED
```

El valor predeterminado hace que `WriteInt` escriba un entero little-endian y que `ReadInt` lea un entero little-endian. Si establece el distintivo `MQC.MQENC_INTEGER_NORMAL` en su lugar, `WriteInt` escribe un entero big-endian y `ReadInt` lee un entero big-endian.

Nota: Se puede producir una pérdida de precisión al convertir de las coma flotante de formato IEEE a las coma flotante de formato zSeries .

public int Expiry {get; set;}
Tiempo de caducidad expresado en décimas de segundo, establecido por la aplicación que coloca el mensaje. Una vez transcurrido el tiempo de caducidad de un mensaje, el gestor de colas lo puede descartar. Si el mensaje ha especificado uno de los distintivos `MQC.MQRO_EXPIRATION` , se genera un informe cuando se descarta el mensaje. El valor predeterminado es `MQC.MQEI_UNLIMITED`, lo que significa que el mensaje nunca caduca.

public int Feedback {get; set;}
Utilice `Comentarios` con un mensaje de tipo `MQC.MQMT_REPORT` para indicar la naturaleza del informe. El sistema define los siguientes códigos de retorno:

- `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

También se pueden utilizar los valores de comentarios definidos por la aplicación en el rango de MQC.MQFB_APPL_FIRST a MQC.MQFB_APPL_LAST . El valor predeterminado de este campo es MQC.MQFB_NONE, lo que indica que no se proporciona ningún comentario.

public string Format {get; set;}
Nombre de formato utilizado por el remitente del mensaje para indicar al destinatario la naturaleza de los datos del mensaje. Puede utilizar sus propios nombres de formato, pero los nombres que empiezan por las letras MQ tienen significados definidos por el gestor de colas. Los formatos incorporados del gestor de colas son:

MQC.MQFMT_ADMIN
Mensaje de solicitud/respuesta del servidor de mandatos.

MQC.MQFMT_COMMAND_1
Mensaje de respuesta de mandato de tipo 1.

MQC.MQFMT_COMMAND_2
Mensaje de respuesta de mandato de tipo 2.

MQC.MQFMT_DEAD_LETTER_HEADER
Cabecera de mensaje no entregado.

MQC.MQFMT_EVENT
Mensaje de suceso.

MQC.MQFMT_NONE
No hay nombre de formato.

MQC.MQFMT_PCF
Mensaje definido por el usuario en formato de mandato programable.

MQC.MQFMT_STRING
Mensaje que consta totalmente de caracteres.

MQC.MQFMT_TRIGGER
Mensaje de desencadenante

MQC.MQFMT_XMIT_Q_HEADER
Cabecera de cola de transmisión.

El valor predeterminado es MQC.MQFMT_NONE.

public byte[] GroupId {get; set;}
Serie de bytes que identifica el grupo de mensajes al que pertenece el mensaje físico. El valor predeterminado es MQC.MQGI_NONE.

public int MessageFlags {get; set;}
Distintivos que controlan la segmentación y el estado de un mensaje.

public byte[] MessageId {get; set;}

Para una llamada MQQueue.Get , este campo especifica el identificador de mensaje del mensaje que se va a recuperar. Normalmente, el gestor de colas devuelve el primer mensaje con un identificador de mensaje y un identificador de correlación que coinciden con los campos del descriptor de mensaje. Permitir que cualquier identificador de mensaje coincida utilizando el valor especial MQC.MQMI_NONE.

Para una llamada MQQueue.Put , este campo especifica el identificador de mensaje que se va a utilizar. Si se especifica MQC.MQMI_NONE , el gestor de colas genera un identificador de mensaje exclusivo cuando se coloca el mensaje. El valor de esta variable de miembro se actualiza después de la colocación, para indicar el identificador de mensaje que se ha utilizado. El valor predeterminado es MQC.MQMI_NONE.

public int MessageLength {get;}
Número de bytes de datos de mensaje en el objeto MQMessage .

public int MessageSequenceNumber {get; set;}
Número de secuencia de un mensaje lógico dentro de un grupo.

public int MessageType {get; set;}
Indica el tipo del mensaje. El sistema define actualmente los valores siguientes:

- MQC.MQMT_DATAGRAM
- MQC.MQMT_REPLY
- MQC.MQMT_REPORT
- MQC.MQMT_REQUEST

También se pueden utilizar valores definidos por la aplicación, en el rango de MQC.MQMT_APPL_FIRST a MQC.MQMT_APPL_LAST. El valor predeterminado de este campo es MQC.MQMT_DATAGRAM.

public int Offset {get; set;}
En un mensaje segmentado, el desplazamiento de datos en un mensaje físico desde el inicio de un mensaje lógico.

public int OriginalLength {get; set;}
La longitud original de un mensaje segmentado.

public int Persistence {get; set;}

Persistencia de mensajes. Los valores siguientes están definidos:

- MQC.MQPER_NOT_PERSISTENT

  Si establece esta opción en un cliente reconectable, el código de razónMQRC_NONE se devuelve a la aplicación cuando la conexión es satisfactoria.

- MQC.MQPER_PERSISTENT

  Si establece esta opción en un cliente reconectable, el código de razón MQRC_CALL_INTERRUPTED se devuelve a la aplicación después de que la conexión sea satisfactoria.

- MQC.MQPER_PERSISTENCE_AS_Q_DEF

El valor predeterminado es MQC.MQPER_PERSISTENCE_AS_Q_DEF, que toma la persistencia del mensaje del atributo de persistencia predeterminado de la cola de destino.

public int Priority {get; set;}
Prioridad del mensaje. El valor especial MQC.MQPRI_PRIORITY_AS_Q_DEF también se puede establecer en el mensaje de salida. A continuación, la prioridad del mensaje se toma del atributo de prioridad predeterminado de la cola de destino. El valor predeterminado es MQC.MQPRI_PRIORITY_AS_Q_DEF.

public int PropertyValidation {get; set;}
Especifica si la validación de propiedades tiene lugar cuando se establece una propiedad del mensaje. Los valores posibles son:

- MQCMHO_DEFAULT_VALIDATION
- MQCMHO_VALIDATE
- MQCMHO_NO_VALIDATION

El valor predeterminado es MQCMHO_DEFAULT_VALIDATION.

public string PutApplicationName {get; set;}
El nombre de la aplicación que ha transferido el mensaje. El valor predeterminado es "".

public int PutApplicationType {get; set;}
El tipo de aplicación que transfiere el mensaje. PutApplicationTipo puede ser un valor definido por el sistema o definido por el usuario. El sistema define los valores siguientes:

- 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

El valor predeterminado es MQC.MQAT_NO_CONTEXT, que indica que no hay información de contexto en el mensaje.

public DateTime PutDateTime {get; set;}
La fecha y hora en que se ha colocado el mensaje.

public string ReplyToQueueManagerName {get; set;}
El nombre del gestor de colas para enviar mensajes de respuesta o de informe. El valor predeterminado es ""y el gestor de colas proporciona el ReplyToQueueManagerName.

public string ReplyToQueueName {get; set;}
El nombre de la cola de mensajes a la que la aplicación que ha emitido la solicitud get para el mensaje envía mensajes MQC.MQMT_REPLY y MQC.MQMT_REPORT . El valor predeterminado de ReplyToQueueName es "".

public int Report {get; set;}
Utilice Informe para especificar opciones sobre los mensajes de informe y respuesta:

- Indica si los informes son necesarios.
- Indica si los datos de mensaje de aplicación se van a incluir en los informes.
- Cómo establecer los identificadores de mensaje y correlación en el informe o respuesta.

Se puede solicitar cualquier combinación de los cuatro tipos de informe:

- Especifique cualquier combinación de los cuatro tipos de informe. Seleccionar cualquiera de las tres opciones para cada tipo de informe, en función de si los datos del mensaje de aplicación se van a incluir en el mensaje de informe.

  1. Confirmar al recibir
     - MQC.MQRO_COA
     - MQC.MQRO_COA_WITH_DATA
     - MQC.MQRO_COA_WITH_FULL_DATA**
  2. Confirmar al entregar
     - MQC.MQRO_COD
     - MQC.MQRO_COD_WITH_DATA
     - MQC.MQRO_COD_WITH_FULL_DATA**
  3. Excepción
     - MQC.MQRO_EXCEPTION
     - MQC.MQRO_EXCEPTION_WITH_DATA
     - MQC.MQRO_EXCEPTION_WITH_FULL_DATA**
  4. Caducidad
     - MQC.MQRO_EXPIRATION
     - MQC.MQRO_EXPIRATION_WITH_DATA
     - MQC.MQRO_EXPIRATION_WITH_FULL_DATA**

Nota: Los valores marcados con ** en la lista no están soportados por los gestores de colas z/OS . No los utilice si es probable que la aplicación acceda a un gestor de colas z/OS , independientemente de la plataforma en la que se ejecute la aplicación.

- Especifique una de las opciones siguientes para controlar cómo se genera el ID de mensaje para el informe o el mensaje de respuesta:
  - `MQC.MQRO_NEW_MSG_ID`
  - `MQC.MQRO_PASS_MSG_ID`
- Especifique una de las opciones siguientes para controlar cómo se va a establecer el ID de correlación del informe o mensaje de respuesta:
  - `MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID`
  - `MQC.MQRO_PASS_CORREL_ID`
- Especifique uno de los siguientes para controlar la disposición del mensaje original cuando no se pueda entregar a la cola de destino:
  - `MQC.MQRO_DEAD_LETTER_Q`
  - `MQC.MQRO_DISCARD_MSG**`
- Si no se especifica ninguna opción de informe, el valor predeterminado es:

```
MQC.MQRO_NEW_MSG_ID |
MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID |
MQC.MQRO_DEAD_LETTER_Q
```

- Puede especificar uno o ambos de los siguientes para solicitar que la aplicación receptora envíe un mensaje de informe de acción positiva o de acción negativa.
  - `MQC.MQRO_PAN`
  - `MQC.MQRO_NAN`

`public int TotalMessageLength {get;}`
Número total de bytes en el mensaje tal como se ha almacenado en la cola de mensajes desde la que se ha recibido este mensaje.

`public string UserId {get; set;}`
`UserId` forma parte del contexto de identidad del mensaje. El gestor de colas generalmente proporciona el valor. Puede alterar temporalmente el valor si tiene autorización para establecer el contexto de identidad.

`public int Version {get; set;}`
La versión de la estructura MQMD en uso.

Métodos de mensaje Read y `Write`

Los métodos `Read` y `Write` realizan las mismas funciones que los miembros de las clases `BinaryReader` y `BinaryWriter` en el espacio de nombres .NET `System.IO` . Consulte MSDN para ver los ejemplos de uso y sintaxis de lenguaje completo. Los métodos leen o escriben desde la posición actual en el almacenamiento intermedio de mensajes. Mueven la posición actual hacia adelante por el número de bytes leídos o grabados.

Nota: Si los datos del mensaje contienen una cabecera MQRFH o MQRFH2 , debe utilizar el método `ReadBytes` para leer los datos.

- Todos los métodos emiten `IOException`.
- Los métodos `ReadFully` redimensionan automáticamente la matriz `byte` o `sbyte` de destino para que se ajuste exactamente al mensaje. También se cambia el tamaño de una matriz nula.
- `Read` métodos throw `EndOfStreamException`.
- `WriteDecimal` métodos throw `MQException`.
- Los métodos `ReadString`, `ReadLine` y `WriteString` se convierten entre Unicode y el juego de caracteres del mensaje; consulte `CharacterSet`.

- Los métodos `Decimal` leen y escriben números decimales empaquetados codificados en formato big-endian, `MQC.MQENC_DECIMAL_NORMAL`o little-endian `MQC.MQENC_DECIMAL_REVERSE` , según el valor de `Codificación`. Los rangos decimales y los tipos .NET correspondientes son los siguientes:

  Decimal2/short
  -999 a 999

  Decimal4/int
  -9999999 a 9999999

  Decimal8/long
  -999999999999999 a 9999999999999999

- Los métodos `Double` y `Float` leen y escriben valores flotantes codificados en formatos big-endian y little-endian de IEE, `MQC.MQENC_FLOAT_IEEE_NORMAL` y `MQC.MQENC_FLOAT_IEEE_REVERSED`, o en formato S/390 , `MQC.MQENC_FLOAT_S390`, según el valor de `Codificación`.
- Los métodos `Int` leen y escriben valores enteros codificados en formato big-endian, `MQC.MQENC_INTEGER_NORMAL`o little-endian, `MQC.MQENC_INTEGER_REVERSED`, según el valor de `Codificación`. Los enteros están todos con signo, excepto la adición de un tipo entero de 2 bytes sin signo. Los tamaños enteros y los tipos .NET y WebSphere MQ son los siguientes:

  2 bytes
  `short`, `Int2`, `ushort`, `UInt2`

  4 bytes
  `int`, `Int4`

  8 bytes
  `long`, `Int8`

- `WriteObject` transfiere la clase de un objeto, los valores de sus campos no transitorios y no estáticos, y los campos de sus supertipos, al almacenamiento intermedio de mensajes.
- `ReadObject` crea un objeto a partir de la clase del objeto, la firma de la clase y los valores de sus campos no transitorios y no estáticos, y los campos de sus supertipos.

Tabla 608. Métodos de lectura y escritura de mensajes

Tipo de destino	Signaturas de método

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)`

Tabla 608. Métodos de lectura y escritura de mensajes (continuación)

Tipo de destino	Signaturas de método
Decimal2	`public void WriteDecimal2(short value)`
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)`

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)`

Tabla 608. Métodos de lectura y escritura de mensajes (continuación)	
Tipo de destino	**Signaturas de método**
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)`

Métodos de almacenamiento intermedio

public void ClearMessage();

Genera IOException.

Descarta los datos del almacenamiento intermedio de mensajes y vuelve a establecer el desplazamiento de datos en cero.

public void ResizeBuffer(int *size*)

Genera IOException.

Una sugerencia para el objeto MQMessage sobre el tamaño del almacenamiento intermedio que puede ser necesario para las operaciones get posteriores. Si el mensaje contiene actualmente datos de mensaje y el nuevo tamaño es menor que el tamaño actual, los datos del mensaje se truncan.

public void Seek(int *pos*)

Genera IOException, ArgumentOutOfRangeException, ArgumentException.

Mueve el cursor a la posición absoluta en el almacenamiento intermedio de mensajes proporcionado por *pos*. Las lecturas y grabaciones posteriores actúan en esta posición del almacenamiento intermedio.

public int SkipBytes(int *i*)

Genera IOException, EndOfStreamException.

Avanza n bytes en el almacenamiento intermedio de mensajes y devuelve n, el número de bytes omitidos.

El método SkipBytes se bloquea hasta que se produce uno de los sucesos siguientes:

- Se omiten todos los bytes
- Se ha detectado el final del almacenamiento intermedio de mensajes
- Se genera una excepción

Métodos de propiedad

public void DeleteProperty(string *name*);

Genera MQException.

Suprime una propiedad con el nombre especificado del mensaje.

name
El nombre de la propiedad que se va a suprimir.

public System.Collections.IEnumerator GetPropertyNames(string *name*)

Genera `MQException`.

Devuelve un `IEnumerator` de todos los nombres de propiedad que coinciden con el nombre especificado. El signo de porcentaje `'%'` se puede utilizar al final del nombre como carácter comodín para filtrar las propiedades del mensaje, coincidiendo con cero o más caracteres, incluido el punto.

name
El nombre de la propiedad en la que debe coincidir.

- Todos los métodos `SetProperty` y `GetProperty` generan `MQException`

Tabla 609. Métodos `SetProperty` y `GetProperty`

Tipo	Signaturas de método

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);`

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);`

Tabla 609. Métodos SetProperty y GetProperty (continuación)

Tipo	Signaturas de método

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);`

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);`

Constructores

public MQMessage();

Crea un objeto MQMessage con información de descriptor de mensaje predeterminado y un almacenamiento intermedio de mensajes vacío.

Clase MQProcess .NET

Utilice MQProcess para consultar los atributos de un proceso de WebSphere MQ . Cree un objeto MQProcess utilizando un constructor o un método MQQueueManager AccessProcess .

Clase

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.MQManagedObject
                └ IBM.WMQ.MQProcess
```

```
public class IBM.WMQ.MQProcess extends IBM.WMQ.MQManagedObject;
```


- "Propiedades" en la página 1277
- "Constructores" en la página 1278

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

`public string ApplicationId {get;}`

Obtiene la serie de caracteres que identifica la aplicación que se va a iniciar. `ApplicationId` lo utiliza una aplicación de supervisor desencadenante. `ApplicationId` se envía a la cola de inicio como parte del mensaje desencadenante.

El valor por omisión es nulo.

`public int ApplicationType {get;}`

Identifica el tipo de proceso que debe iniciar una aplicación de supervisor desencadenante. Los tipos estándar están definidos, pero se pueden utilizar otros:

- `MQAT_AIX`
- `MQAT_CICS`
- `MQAT_IMS`
- `MQAT_MVS`
- `MQAT_NATIVE`
- `MQAT_OS400`
- `MQAT_UNIX`
- `MQAT_WINDOWS`
- `MQAT_JAVA`
- `MQAT_USER_FIRST`
- `MQAT_USER_LAST`

El valor predeterminado es `MQAT_NATIVE`.

`public string EnvironmentData {get;}`

Obtiene información sobre el entorno de la aplicación que se va a iniciar.

El valor por omisión es nulo.

`public string UserData {get;}`

Obtiene información que el usuario ha proporcionado sobre la aplicación que se va a iniciar.

El valor por omisión es nulo.

Constructores

public MQProcess(MQQueueManager *queueManager*, string *processName*, int *openOptions*);
public MQProcess(MQQueueManager *qMgr*, string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Genera MQException.

Acceda a un proceso de WebSphere MQ en el gestor de colas *qMgr* para consultar los atributos de proceso.

qMgr
Gestor de colas al que acceder.

processName
El nombre del proceso que se va a abrir.

openOptions
Opciones que controlan la apertura del proceso. Las opciones válidas que se pueden añadir o combinar utilizando un OR a nivel de bit son:

- MQC.MQOO_FAIL_IF_QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName
El nombre del gestor de colas en el que se define el proceso. Puede dejar un nombre de gestor de colas en blanco o nulo si el gestor de colas es el mismo que al que está accediendo el proceso.

alternateUserId

Si se especifica MQC.MQOO_ALTERNATE_USER_AUTHORITY en el parámetro *openOptions* , *alternateUserId* especifica el ID de usuario alternativo utilizado para comprobar la autorización de la acción. Si no se especifica MQOO_ALTERNATE_USER_AUTHORITY , *alternateUserId* puede estar en blanco o ser nulo.

Se utiliza la autorización de usuario predeterminada para la conexión con el gestor de colas si no se especifica MQC.MQOO_ALTERNATE_USER_AUTHORITY .

public MQProcess MQQueueManager.AccessProcess(string *processName*, int *openOptions*);
public MQProcess MQQueueManager.AccessProcess(string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Genera MQException.

Acceda a un proceso de WebSphere MQ en este gestor de colas para consultar los atributos de proceso.

processName
El nombre del proceso que se va a abrir.

openOptions
Opciones que controlan la apertura del proceso. Las opciones válidas que se pueden añadir o combinar utilizando un OR a nivel de bit son:

- MQC.MQOO_FAIL_IF_QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

queueManagerName
El nombre del gestor de colas en el que se define el proceso. Puede dejar un nombre de gestor de colas en blanco o nulo si el gestor de colas es el mismo que al que está accediendo el proceso.

alternateUserId

Si se especifica MQC.MQOO_ALTERNATE_USER_AUTHORITY en el parámetro *openOptions* , *alternateUserId* especifica el ID de usuario alternativo utilizado para comprobar la autorización de la acción. Si no se especifica MQOO_ALTERNATE_USER_AUTHORITY , *alternateUserId* puede estar en blanco o ser nulo.

Se utiliza la autorización de usuario predeterminada para la conexión con el gestor de colas si no se especifica MQC.MQOO_ALTERNATE_USER_AUTHORITY .

Clase MQPropertyDescriptor .NET

Utilice MQPropertyDescriptor como parámetro para los métodos MQMessage GetProperty y SetProperty . MQPropertyDescriptor describe una propiedad MQMessage .

Clase

```
System.Object
        │
        └─ IBM.WMQ.MQPropertyDescriptor
```

```
public class IBM.WMQ.MQPropertyDescriptor extends System.Object;
```


- "Propiedades" en la página 1279
- "Constructores" en la página 1280

Propiedades

Prueba de MQException que se genera al obtener las propiedades.

public int Context {get; set;}

El contexto de mensaje al que pertenece la propiedad. Los valores posibles son:

MQC.MQPD_NO_CONTEXT
La propiedad no está asociada a un contexto de mensaje.

MQC.MQPD_USER_CONTEXT
La propiedad está asociada al contexto de usuario.

Si el usuario está autorizado, se guarda una propiedad asociada con el contexto de usuario cuando se recupera un mensaje. Un método Put posterior que haga referencia al contexto guardado, puede pasar la propiedad al nuevo mensaje.

public int CopyOptions {get; set;}

CopyOptions describe en qué tipo de mensaje se puede copiar la propiedad.

Cuando un gestor de colas recibe un mensaje que contiene una propiedad definida de WebSphere MQ que el gestor de colas reconoce como incorrecta, el gestor de colas corrige el valor del campo CopyOptions .

Se puede especificar cualquier combinación de las opciones siguientes. Combine las opciones añadiendo los valores o utilizando ORa nivel de bit.

MQC.MQCOPY_ALL
La propiedad se copia en todos los tipos de mensajes posteriores.

MQC.MQCOPY_FORWARD
La propiedad se copia en un mensaje que se está reenviando.

MQC.MQCOPY_PUBLISH
La propiedad se copia en el mensaje recibido por un suscriptor cuando se publica un mensaje.

MQC.MQCOPY_REPLY
La propiedad se copia en un mensaje de respuesta.

MQC.MQCOPY_REPORT
La propiedad se copia en un mensaje de informe.

MQC.MQCOPY_DEFAULT
El valor ha indicado que no se ha especificado ninguna otra opción de copia. No existe ninguna relación entre la propiedad y los mensajes posteriores. `MQC.MQCOPY_DEFAULT` siempre se devuelve para las propiedades del descriptor de mensaje.

MQC.MQCOPY_NONE
Lo mismo que `MQC.MQCOPY_DEFAULT`

public int Options { set; }
`Opciones` toma como valor predeterminado `CMQC.MQPD_NONE`. No puede establecer ningún otro valor.

public int Support { get; set; }

Establezca `Soporte` para especificar el nivel de soporte necesario para las propiedades de mensaje definidas por WebSphere MQ. El soporte para todas las demás propiedades es opcional. Se puede especificar cualquiera o ninguno de los valores siguientes

MQC.MQPD_SUPPORT_OPTIONAL
Un gestor de colas acepta la propiedad aunque no esté soportada. La propiedad se puede descartar para que el mensaje fluya a un gestor de colas que no da soporte a las propiedades de mensaje. Este valor también se asigna a propiedades que no están definidas en WebSphere MQ .

MQC.MQPD_SUPPORT_REQUIRED
Se necesita soporte para la propiedad. Si coloca el mensaje en un gestor de colas que no da soporte a la propiedad definida por WebSphere MQ, el método falla. Devuelve el código de terminación `MQC.MQCC_FAILED` y el código de razón `MQC.MQRC_UNSUPPORTED_PROPERTY`.

MQC.MQPD_SUPPORT_REQUIRED_IF_LOCAL

Se necesita soporte para la propiedad, si el mensaje está destinado a una cola local. Si coloca el mensaje en una cola local de un gestor de colas que no da soporte a la propiedad definida por WebSphere MQ, el método falla. Devuelve el código de terminación `MQC.MQCC_FAILED` y el código de razón `MQC.MQRC_UNSUPPORTED_PROPERTY`.

No se realiza ninguna comprobación si el mensaje se coloca en un gestor de colas remoto.

Constructores

PropertyDescriptor();
Cree un descriptor de propiedad.

Clase MQPutMessageOptions .NET

Utilice `MQPutMessageOptions` para especificar cómo se envían los mensajes. Modifica el comportamiento de `MQDestination.Put`.

Clase

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.MQPutMessageOptions
```

```
public class IBM.WMQ.MQPutMessageOptions extends IBM.WMQ.MQBaseObject;
```

- “Propiedades” en la página 1281“Constructores” en la página 1283

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

Nota: El comportamiento de algunas de las opciones disponibles en esta clase depende del entorno en el que se utilizan. Estos elementos se marcan con un asterisco, *.

`public MQQueue ContextReference {get; set;}`

Si el campo `options` incluye `MQC.MQPMO_PASS_IDENTITY_CONTEXT` o `MQC.MQPMO_PASS_ALL_CONTEXT`, establezca este campo para que haga referencia al `MQQueue` del que se tomará la información de contexto.

El valor inicial de este campo es nulo.

`public int InvalidDestCount {get;}*`

Generalmente, se utiliza para las listas de distribución, `InvalidDest` indica el número de mensajes que no se han podido enviar a las colas de una lista de distribución. El recuento incluye las colas que no se han podido abrir y también las colas que se han abierto correctamente, pero para las que la operación de colocación ha fallado.

.NET no da soporte a listas de distribución, pero `InvalidDest` se establece al abrir una sola cola.

`public int KnownDestCount {get;} *`

Generalmente se utiliza para listas de distribución, `KnownDestKnownDest` indica el número de mensajes que la llamada actual ha enviado correctamente a las colas que se resuelven en las colas locales.

.NET no da soporte a listas de distribución, pero `InvalidDest` se establece al abrir una sola cola.

`public int Options {get; set;}`

Opciones que controlan la acción de `MQDestination.put` y `MQQueueManager.put`. Se puede especificar cualquiera de los valores siguientes o ninguno de ellos. Si se necesita más de una opción, los valores se pueden añadir o combinar utilizando el operador OR a nivel de bit.

`MQC.MQPMO_ASYNC_RESPONSE`
Esta opción hace que la llamada `MQDestination.put` se realice de forma asíncrona, con algunos datos de respuesta.

`MQC.MQPMO_DEFAULT_CONTEXT`
Asocie el contexto predeterminado con el mensaje.

`MQC.MQPMO_FAIL_IF_QUIESCING`
Fallar si el gestor de colas se está desactivando temporalmente.

`MQC.MQPMO_LOGICAL_ORDER*`
Coloque los mensajes lógicos y los segmentos de los grupos de mensajes en su orden lógico.

Si utiliza la opción `MQPMO_LOGICAL_ORDER` en un cliente reconectable, el código de razón `MQRC_RECONNECT_INCOMPATIBLE` se devuelve a la aplicación.

`MQC.MQPMO_NEW_CORREL_ID*`
Genere un nuevo ID de correlación para cada mensaje enviado.

MQC.MQPMO_NEW_MSG_ID*
Genere un nuevo ID de mensaje para cada mensaje enviado.

MQC.MQPMO_NONE
No se ha especificado ninguna opción. No lo utilice con otras opciones.

MQC.MQPMO_NO_CONTEXT
No se debe asociar ningún contexto con el mensaje.

MQC.MQPMO_NO_SYNCPOINT
Colocar un mensaje sin control de punto de sincronización. Si no se especifica la opción de control de punto de sincronización, se presupone un valor predeterminado de ningún punto de sincronización.

MQC.MQPMO_PASS_ALL_CONTEXT
Pasar todo el contexto desde un descriptor de contexto de cola de entrada.

MQC.MQPMO_PASS_IDENTITY_CONTEXT
Pasar contexto de identidad desde un descriptor de contexto de cola de entrada.

MQC.MQPMO_RESPONSE_AS_Q_DEF

Para una llamada `MQDestination.put` , esta opción toma el tipo de respuesta put del atributo `DEFPRESP` de la cola.

Para una llamada `MQQueueManager.put` , esta opción hace que la llamada se realice de forma síncrona.

MQC.MQPMO_RESPONSE_AS_TOPIC_DEF
`MQC.MQPMO_RESPONSE_AS_TOPIC_DEF` es un sinónimo de `MQC.MQPMO_RESPONSE_AS_Q_DEF` para su uso con objetos de tema.

MQC.MQPMO_RETAIN

El gestor de colas debe retener la publicación que se está enviando. Si se utiliza esta opción y la publicación no se puede retener, el mensaje no se publica y la llamada falla con `MQC.MQRC_PUT_NOT_RETAINED`.

Solicite una copia de esta publicación después de la hora en que se publicó, llamando al método `MQSubscription.RequestPublicationUpdate` . La publicación guardada se envía a las aplicaciones que crean una suscripción sin establecer la opción `MQC.MQSO_NEW_PUBLICATIONS_ONLY` . Compruebe la propiedad de mensaje `MQIsRetained` de una publicación, cuando se reciba, para averiguar si era la publicación retenida.

Cuando un suscriptor solicita las publicaciones retenidas, la suscripción utilizada puede contener un comodín en la serie de tema. Si hay varias publicaciones retenidas en el árbol de temas que coinciden con la suscripción, se envían todas.

MQC.MQPMO_SET_ALL_CONTEXT
Establezca todo el contexto de la aplicación.

MQC.MQPMO_SET_IDENTITY_CONTEXT
Establezca el contexto de identidad de la aplicación.

MQC.MQPMO_SYNC_RESPONSE
Esta opción hace que la llamada `MQDestination.put` o `MQQueueManager.put` se realice de forma síncrona, con datos de respuesta completos.

MQC.MQPMO_SUPPRESS_REPLYTO
Cualquier información rellenada en los campos `ReplyToQueueName` y `ReplyToQueueManagerName` de la publicación no se pasa a los suscriptores. Si esta opción se utiliza en combinación con una opción de informe que requiere un `ReplyToQueueName`, la llamada falla con `MQC.MQRC_MISSING_REPLY_TO_Q`.

MQC.MQPMO_SYNCPOINT
Colocar un mensaje con control de punto de sincronización. El mensaje no está visible fuera de la unidad de trabajo hasta que se confirme la unidad de trabajo. Si la unidad de trabajo se restituye, se suprime el mensaje.

public int RecordFields {get; set;}*

Información sobre listas de distribución. Las listas de distribución no son compatibles con .NET.

public string ResolvedQueueManagerName {get;}

Campo de salida establecido por el gestor de colas en el nombre del gestor de colas que es propietario de la cola especificada por el nombre de cola remota. `ResolvedQueueManagerName` puede ser diferente del nombre del gestor de colas desde el que se ha accedido a la cola si la cola es una cola remota.

Sólo se devuelve un valor no en blanco si el objeto es una sola cola. Si el objeto es una lista de distribución o un tema, el valor devuelto no está definido.

public string ResolvedQueueName {get;}

Campo de salida establecido por el gestor de colas en el nombre de la cola en la que se coloca el mensaje. `ResolvedQueueNombre` puede ser diferente del nombre utilizado para abrir la cola si la cola abierta era un alias o una cola modelo.

Sólo se devuelve un valor que no esté en blanco si el objeto es una sola cola. Si el objeto es una lista de distribución o un tema, el valor devuelto no está definido.

public int UnknownDestCount {get;}*

Generalmente se utiliza para listas de distribución, `UnknownDestdesconocido` es un campo de salida establecido por el gestor de colas. Informa del número de mensajes que la llamada actual ha enviado satisfactoriamente a las colas que se resuelven en colas remotas.

.NET no da soporte a listas de distribución, pero `InvalidDest` se establece al abrir una sola cola.

Constructores

public MQPutMessageOptions();

Construya un nuevo objeto `MQPutMessageOptions` sin opciones establecidas y un `ResolvedQueueNombre` y `ResolvedQueueManagerNameen` blanco.

Clase MQQueue .NET

Utilice `MQQueue` para enviar y recibir mensajes y atributos de consulta de una cola de WebSphere MQ . Cree un objeto `MQQueue` utilizando un constructor o un método `MQQueueManager.AccessProcess` .

Clase

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.MQManagedObject
                └ IBM.WMQ.MQDestination
                    └ IBM.WMQ.MQQueue
```

`public class IBM.WMQ.MQQueue extends IBM.WMQ.MQDestination;`

- “Propiedades” en la página 1284
- “Métodos” en la página 1286
- “Constructores” en la página 1288

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

public int ClusterWorkLoadPriority {get;}
Especifica la prioridad de la cola. Este parámetro sólo es válido para colas locales, remotas y alias.

public int ClusterWorkLoadRank {get;}
Especifica el rango de la cola. Este parámetro sólo es válido para colas locales, remotas y alias.

public int ClusterWorkLoadUseQ {get;}
Especifica el comportamiento de una operación MQPUT cuando la cola de destino tiene una instancia local y al menos una instancia de clúster remoto. Este parámetro no se aplica si MQPUT se origina en un canal de clúster. Este parámetro sólo es válido para colas locales.

public DateTime CreationDateTime {get;}
Fecha y hora en que se ha creado esta cola.

public int CurrentDepth {get;}
Obtiene el número de mensajes que hay actualmente en la cola. Este valor se incrementa durante una llamada put y durante la restitución de una llamada get. Se decrementa durante una operación get de no examinar y durante la restitución de una llamada put.

public int DefinitionType {get;}
Cómo se ha definido la cola. Los valores posibles son:

- `MQC.MQQDT_PREDEFINED`
- `MQC.MQQDT_PERMANENT_DYNAMIC`
- `MQC.MQQDT_TEMPORARY_DYNAMIC`

public int InhibitGet {get; set;}
Controla si puede obtener mensajes en esta cola o para este tema. Los valores posibles son:

- `MQC.MQQA_GET_INHIBITED`
- `MQC.MQQA_GET_ALLOWED`

public int InhibitPut {get; set;}
Controla si puede colocar mensajes en esta cola o para este tema. Los valores posibles son:

- `MQQA_PUT_INHIBITED`
- `MQQA_PUT_ALLOWED`

public int MaximumDepth {get;}
Número máximo de mensajes que pueden existir en la cola en cualquier momento. Un intento de colocar un mensaje en una cola que ya contiene este número de mensajes falla con el código de razón `MQC.MQRC_Q_FULL`.

public int MaximumMessageLength {get;}
Longitud máxima de los datos de aplicación que pueden existir en cada mensaje de esta cola. Un intento de colocar un mensaje mayor que este valor falla con el código de razón `MQC.MQRC_MSG_TOO_BIG_FOR_Q`.

public int NonPersistentMessageClass {get;}
El nivel de fiabilidad para los mensajes no permanentes colocados en esta cola.

public int OpenInputCount {get;}
Número de descriptores de contexto que son válidos actualmente para eliminar mensajes de la cola. `OpenInput` es el número total de descriptores de contexto de entrada válidos conocidos por el gestor de colas local, no sólo los descriptores de contexto creados por la aplicación.

public int OpenOutputCount {get;}
Número de descriptores de contexto válidos actualmente para añadir mensajes a la cola. `OpenOutput` es el número total de descriptores de contexto de salida válidos conocidos por el gestor de colas local, no sólo los descriptores de contexto creados por la aplicación.

public int QueueAccounting {get;}
Especifica si puede habilitar la recopilación de información de contabilidad para la cola.

public int QueueMonitoring {get;}
Especifica si puede habilitar la supervisión para la cola.

public int QueueStatistics {get;}
Especifica si puede habilitar la recopilación de estadísticas para la cola.

public int QueueType {get;}
El tipo de esta cola con uno de los valores siguientes:

- MQC.MQQT_ALIAS
- MQC.MQQT_LOCAL
- MQC.MQQT_REMOTE
- MQC.MQQT_CLUSTER

public int Shareability {get;}
Indica si la cola se puede abrir para entrada varias veces. Los valores posibles son:

- MQC.MQQA_SHAREABLE
- MQC.MQQA_NOT_SHAREABLE

public string TPIPE {get;}
El nombre de TPIPE utilizado para la comunicación con OTMA utilizando el puente WebSphere MQ IMS .

public int TriggerControl {get; set;}
Indica si los mensajes desencadenantes se graban en una cola de inicio, para iniciar una aplicación para dar servicio a la cola. Los valores posibles son:

- MQC.MQTC_OFF
- MQC.MQTC_ON

public string TriggerData {get; set;}
Los datos de formato libre que el gestor de colas inserta en el mensaje desencadenante. Inserta TriggerData cuando un mensaje que llega a esta cola hace que se grabe un mensaje desencadenante en la cola de inicio. La longitud máxima permitida de la serie la proporciona MQC.MQ_TRIGGER_DATA_LENGTH.

public int TriggerDepth {get; set;}
El número de mensajes que deben estar en la cola antes de que se escriba un mensaje desencadenante cuando el tipo de desencadenante se establece en MQC.MQTT_DEPTH.

public int TriggerMessagePriority {get; set;}
Prioridad de mensaje bajo la que los mensajes no contribuyen a la generación de mensajes desencadenantes. Es decir, el gestor de colas ignora estos mensajes al decidir si se genera un desencadenante. Un valor de cero hace que todos los mensajes contribuyan a la generación de mensajes desencadenantes.

public int TriggerType {get; set;}
Las condiciones en las que se graban los mensajes desencadenantes como resultado de los mensajes que llegan a esta cola. Los valores posibles son:

- MQC.MQTT_NONE
- MQC.MQTT_FIRST
- MQC.MQTT_EVERY
- MQC.MQTT_DEPTH

Métodos

public void Get(MQMessage *message*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*, int *MaxMsgSize*);

Genera MQException.

Obtiene un mensaje de una cola.

Si la obtención falla, el objeto MQMessage no se modifica. Si se ejecuta correctamente, el descriptor de mensaje y las partes de datos de mensaje del MQMessage se sustituyen por el descriptor de mensaje y los datos de mensaje del mensaje de entrada.

Todas las llamadas a WebSphere MQ desde un MQQueueManager determinado son síncronas. Por lo tanto, si realiza una operación get con espera, todas las demás hebras que utilizan el mismo MQQueueManager no pueden realizar más llamadas WebSphere MQ hasta que se realice la llamada Get. Si necesita varias hebras para acceder a WebSphere MQ simultáneamente, cada hebra debe crear su propio objeto MQQueueManager .

message
Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada MessageId y CorrelationId estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón MQRC_BACKED_OUT después de una reconexión satisfactoria, para los mensajes recibidos en MQGM_SYNCPOINT.

getMessageOptions
Opciones que controlan la acción de la obtención.

El uso de la opción MQC.MQGMO_CONVERT puede dar como resultado una excepción con el código de razón MQC.MQRC_CONVERTED_STRING_TOO_BIG al convertir de códigos de caracteres de un solo byte a códigos de doble byte. En este caso, el mensaje se copia en el almacenamiento intermedio sin conversión.

Si no se especifica *getMessageOptions* , la opción de mensaje utilizada es MQGMO_NOWAIT.

Si utiliza la opción MQGMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

MaxMsgSize
El mensaje más grande que este objeto de mensaje va a recibir. Si el mensaje en la cola es mayor que este tamaño, se produce una de las dos cosas siguientes:

- Si el distintivo MQGMO_ACCEPT_TRUNCATED_MSG se establece en el objeto MQGetMessageOptions , el mensaje se rellena con la mayor cantidad posible de datos de mensaje. Se genera una excepción con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_ACCEPTED .
- Si el distintivo MQGMO_ACCEPT_TRUNCATED_MSG no está establecido, el mensaje se deja en la cola. Se genera una excepción con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_FAILED .

Si no se especifica *MaxMsgSize* , se recupera todo el mensaje.

public void Put(MQMessage *message*);
public void Put(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);

Genera MQException.

Coloca un mensaje en una cola.

Las modificaciones en el objeto MQMessage después de que se haya realizado la llamada Put no afectan al mensaje real en el tema de publicación o cola WebSphere MQ .

Put actualiza las propiedades MessageId y CorrelationId del objeto MQMessage y no borra los datos del mensaje. Las llamadas Put o Get adicionales hacen referencia a la información actualizada en el objeto MQMessage . Por ejemplo, en el siguiente fragmento de código, el primer mensaje contiene a y el segundo ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

message
Un objeto MQMessage que contiene los datos del descriptor de mensaje y el mensaje que se va a enviar. El descriptor de mensaje se puede modificar como consecuencia de este método. Los valores del descriptor de mensaje inmediatamente después de la finalización de este método son los valores que se colocaron en la cola o se publicaron en el tema.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- MQRC_CALL_INTERRUPTED si la conexión se interrumpe al ejecutar una llamada Put en un mensaje persistente y la reconexión se realiza correctamente.
- MQRC_NONE si la conexión es satisfactoria al ejecutar una llamada Put en un mensaje no persistente (consulte Recuperación de aplicaciones).

putMessageOptions
Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptons .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

Nota: Por motivos de simplicidad y rendimiento, si desea transferir un único mensaje a una cola, utilice el objeto MQQueueManager.Put . Debe tener un objeto MQQueue para esto.

public void PutForwardMessage(MQMessage *message*);
public void PutForwardMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);

Genera MQException

Coloque un mensaje que se está reenviando en la cola, donde *message* es el mensaje original.

message
Un objeto MQMessage que contiene los datos del descriptor de mensaje y el mensaje que se va a enviar. El descriptor de mensaje se puede modificar como consecuencia de este método. Los valores del descriptor de mensaje inmediatamente después de la finalización de este método son los valores que se colocaron en la cola o se publicaron en el tema.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- MQRC_CALL_INTERRUPTED si la conexión se interrumpe al ejecutar una llamada Put en un mensaje persistente y la reconexión se realiza correctamente.
- MQRC_NONE si la conexión es satisfactoria al ejecutar una llamada Put en un mensaje no persistente (consulte Recuperación de aplicaciones).

putMessageOptions
Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptons .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

public void PutReplyMessage(MQMessage *message*)
public void PutReplyMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*)

Genera MQException.

Coloque un mensaje de respuesta en la cola, donde *message* es el mensaje original.

message
Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada MessageId y CorrelationId estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón MQRC_BACKED_OUT después de una reconexión satisfactoria, para los mensajes recibidos en MQGM_SYNCPOINT.

putMessageOptions
Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptons .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

public void PutReportMessage(MQMessage *message*)
public void PutReportMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*)

Genera MQException.

Coloque un mensaje de informe en la cola, donde *message* es el mensaje original.

message
Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada MessageId y CorrelationId estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón MQRC_BACKED_OUT después de una reconexión satisfactoria, para los mensajes recibidos en MQGM_SYNCPOINT.

putMessageOptions
Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptons .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

Constructores

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 cola en este gestor de colas.

Puede obtener o examinar mensajes, colocar mensajes, consultar los atributos de la cola o establecer los atributos de la cola. Si la cola especificada es una cola modelo, se crea una cola local dinámica. Consulte el atributo name del objeto MQQueue resultante para averiguar el nombre de la cola dinámica.

queueName
Nombre de la cola que se va a abrir.

openOptions
Opciones que controlan la apertura de la cola.

MQC.MQOO_ALTERNATE_USER_AUTHORITY
Validar con el identificador de usuario especificado.

MQC.MQOO_BIND_AS_QDEF
Utilice el enlace predeterminado para la cola.

MQC.MQOO_BIND_NOT_FIXED
No enlazar a un destino específico.

MQC.MQOO_BIND_ON_OPEN
Manejador de enlace a destino cuando se abre la cola.

MQC.MQOO_BROWSE
Abra para examinar el mensaje.

MQC.MQOO_FAIL_IF_QUIESCING
Fallar si el gestor de colas se está desactivando temporalmente.

MQC.MQOO_INPUT_AS_Q_DEF
Abrir para obtener mensajes utilizando el valor predeterminado definido por la cola.

MQC.MQOO_INPUT_SHARED
Abierto para obtener mensajes con acceso compartido.

MQC.MQOO_INPUT_EXCLUSIVE
Abierto para obtener mensajes con acceso exclusivo.

MQC.MQOO_INQUIRE
Abrir para consulta necesaria si desea consultar propiedades.

MQC.MQOO_OUTPUT
Abierto para colocar mensajes.

MQC.MQOO_PASS_ALL_CONTEXT
Permitir que se pase todo el contexto.

MQC.MQOO_PASS_IDENTITY_CONTEXT
Permitir que se pase el contexto de identidad.

MQC.MQOO_SAVE_ALL_CONTEXT
Guardar contexto cuando se recupere el mensaje.

MQC.MQOO_SET
Abra para establecer los atributos necesarios si desea establecer las propiedades.

MQC.MQOO_SET_ALL_CONTEXT
Permite establecer todo el contexto.

MQC.MQOO_SET_IDENTITY_CONTEXT
Permite establecer el contexto de identidad.

queueManagerName
Nombre del gestor de colas en el que está definida la cola. Un nombre que está totalmente en blanco o es nulo indica el gestor de colas al que está conectado el objeto `MQQueueManager` .

dynamicQueueName
dynamicQueueName se ignora a menos que `queueName` especifique el nombre de una cola modelo. Si lo hace, *dynamicQueueName* especifica el nombre de la cola dinámica que se va a crear. Un nombre en blanco o nulo no es válido si `queueName` especifica el nombre de una cola modelo. Si el último carácter no en blanco del nombre es un asterisco, `*`, el gestor de colas sustituye el asterisco por una serie de caracteres. Los caracteres garantizan que el nombre generado para la cola es exclusivo en este gestor de colas.

alternateUserId
Si se especifica `MQC.MQOO_ALTERNATE_USER_AUTHORITY` en el parámetro `openOptions` , *alternateUserId* especifica el identificador de usuario alternativo que se utiliza para comprobar la autorización para la apertura. Si no se especifica

MQC.MQOO_ALTERNATE_USER_AUTHORITY , *alternateUserId* puede dejarse en blanco o ser nulo.

public MQQueue(MQQueueManager *queueManager*, string *queueName*, int *openOptions*, string *queueManagerName*, string *dynamicQueueName*, string *alternateUserId*);

Genera MQException.

Accede a una cola en queueManager.

Puede obtener o examinar mensajes, colocar mensajes, consultar los atributos de la cola o establecer los atributos de la cola. Si la cola especificada es una cola modelo, se crea una cola local dinámica. Consulte el atributo name del objeto MQQueue resultante para averiguar el nombre de la cola dinámica.

queueManager
Gestor de colas en el que acceder a la cola.

queueName
Nombre de la cola que se va a abrir.

openOptions
Opciones que controlan la apertura de la cola.

MQC.MQOO_ALTERNATE_USER_AUTHORITY
Validar con el identificador de usuario especificado.

MQC.MQOO_BIND_AS_QDEF
Utilice el enlace predeterminado para la cola.

MQC.MQOO_BIND_NOT_FIXED
No enlazar a un destino específico.

MQC.MQOO_BIND_ON_OPEN
Manejador de enlace a destino cuando se abre la cola.

MQC.MQOO_BROWSE
Abra para examinar el mensaje.

MQC.MQOO_FAIL_IF_QUIESCING
Fallar si el gestor de colas se está desactivando temporalmente.

MQC.MQOO_INPUT_AS_Q_DEF
Abrir para obtener mensajes utilizando el valor predeterminado definido por la cola.

MQC.MQOO_INPUT_SHARED
Abierto para obtener mensajes con acceso compartido.

MQC.MQOO_INPUT_EXCLUSIVE
Abierto para obtener mensajes con acceso exclusivo.

MQC.MQOO_INQUIRE
Abrir para consulta necesaria si desea consultar propiedades.

MQC.MQOO_OUTPUT
Abierto para colocar mensajes.

MQC.MQOO_PASS_ALL_CONTEXT
Permitir que se pase todo el contexto.

MQC.MQOO_PASS_IDENTITY_CONTEXT
Permitir que se pase el contexto de identidad.

MQC.MQOO_SAVE_ALL_CONTEXT
Guardar contexto cuando se recupere el mensaje.

MQC.MQOO_SET
Abra para establecer los atributos necesarios si desea establecer las propiedades.

MQC.MQOO_SET_ALL_CONTEXT
Permite establecer todo el contexto.

MQC.MQOO_SET_IDENTITY_CONTEXT
Permite establecer el contexto de identidad.

queueManagerName
Nombre del gestor de colas en el que está definida la cola. Un nombre que está totalmente en blanco o es nulo indica el gestor de colas al que está conectado el objeto `MQQueueManager` .

dynamicQueueName
dynamicQueueName se ignora a menos que queueName especifique el nombre de una cola modelo. Si lo hace, *dynamicQueueName* especifica el nombre de la cola dinámica que se va a crear. Un nombre en blanco o nulo no es válido si queueName especifica el nombre de una cola modelo. Si el último carácter no en blanco del nombre es un asterisco, `*`, el gestor de colas sustituye el asterisco por una serie de caracteres. Los caracteres garantizan que el nombre generado para la cola es exclusivo en este gestor de colas.

alternateUserId
Si se especifica `MQC.MQOO_ALTERNATE_USER_AUTHORITY` en el parámetro `openOptions` , *alternateUserId* especifica el identificador de usuario alternativo que se utiliza para comprobar la autorización para la apertura. Si no se especifica `MQC.MQOO_ALTERNATE_USER_AUTHORITY` , *alternateUserId* puede dejarse en blanco o ser nulo.

Clase MQQueueManager .NET

Utilice `MQQueueManager` para conectarse a un gestor de colas y acceder a los objetos del gestor de colas. También controla las transacciones. El constructor `MQQueueManager` crea una conexión de cliente o servidor.

Clase

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.ManagedObject
                └ IBM.WMQ.MQQueueManager
```

```
public class IBM.WMQ.MQQueueManager extends IBM.WMQ.MQManagedObject;
```


- “Propiedades” en la página 1291
- “Métodos” en la página 1295
- “Constructores” en la página 1300

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

public int AccountingConnOverride {get;}
Indica si las aplicaciones pueden alterar temporalmente el valor de los valores de contabilidad y de contabilidad de cola de `MQI` .

public int AccountingInterval {get;}
Tiempo antes de que se escriban los registros de contabilidad intermedios (en segundos).

public int ActivityRecording {get;}
Controla la generación de los informes de actividades.

public int AdoptNewMCACheck {get;}
Especifica qué elementos se comprueban para determinar si el MCA se adopta cuando se detecta un nuevo canal de entrada. Para que se adopte, el nombre de MCA debe coincidir con el nombre de un MCA activo.

public int AdoptNewMCAInterval {get;}
Cantidad de tiempo, en segundos, que el nuevo canal espera a que finalice el canal huérfano.

public int AdoptNewMCAType {get;}
Indica si una instancia de MCA huérfana debe adoptarse (reiniciarse) cuando se detecta una nueva solicitud de canal de entrada que coincide con el valor de MCACheck AdoptNew.

public int BridgeEvent {get;}
Indica si se generan sucesos de puente IMS .

public int ChannelEvent {get;}
Indica si se generan sucesos de canal.

public int ChannelInitiatorControl {get;}
Si el iniciador de canal se inicia automáticamente cuando se inicia el gestor de colas.

public int ChannelInitiatorAdapters {get;}
Número de subtareas de adaptador para procesar llamadas de WebSphere MQ .

public int ChannelInitiatorDispatchers {get;}
Número de asignadores a utilizar para el iniciador de canal.

public int ChannelInitiatorTraceAutoStart {get;}
Especifica si el rastreo del iniciador de canal se inicia automáticamente.

public int ChannelInitiatorTraceTableSize {get;}
Tamaño, en megabytes, del espacio de datos de rastreo de un iniciador de canal .

public int ChannelMonitoring {get;}
Si se utiliza la supervisión de canal.

public int ChannelStatistics {get;}
Controla la recopilación de los datos estadísticos para los canales.

public int CharacterSet {get;}
Devuelve el identificador de juego de caracteres codificado (CCSID) del gestor de colas. El gestor de colas utiliza `CharacterSet` para todos los campos de serie de caracteres de la interfaz de programación de aplicaciones.

public int ClusterSenderMonitoring {get;}
Controla la recopilación de datos de supervisión en línea para canales emisores de clúster definidos automáticamente.

public int ClusterSenderStatistics {get;}
Controla la recopilación de datos estadísticos para los canales emisores de clúster definidos automáticamente.

public int ClusterWorkLoadMRU {get;}
Número máximo de canales de clúster de salida.

public int ClusterWorkLoadUseQ {get;}
El valor predeterminado de la propiedad `MQQueue` , `ClusterWorkLoadUseQ`, si especifica un valor de `QMGR`.

public int CommandEvent {get;}
Especifica si se generan sucesos de mandato.

public string CommandInputQueueName {get;}
Devuelve el nombre de la cola de entrada de mandatos definida en el gestor de colas. Las aplicaciones pueden enviar mandatos a esta cola, si están autorizadas a hacerlo.

public int CommandLevel {get;}
Indica el nivel de función del gestor de colas. El conjunto de funciones que corresponden a un nivel de función determinado depende de la plataforma. En una plataforma determinada, puede confiar en

que cada gestor de colas dé soporte a las funciones en el nivel funcional más bajo común a todos los gestores de colas.

public int CommandLevel {get;}
Indica si el servidor de mandatos se inicia automáticamente cuando se inicia el gestor de colas.

public string DNSGroup {get;}
El nombre del grupo al que debe unirse el escucha TCP que maneja las transmisiones de entrada para el grupo de compartición de colas. Se une a este grupo cuando se utiliza el soporte de Workload Manager for Dynamic Domain Name Services (DDNS).

public int DNSWLM {get;}
Indica si el escucha TCP que maneja las transmisiones de entrada para el grupo de compartición de colas debe registrarse con el Gestor de carga de trabajo para DDNS.

public int IPAddressVersion {get;}
Qué protocolo IP (IPv4 o IPv6) utilizar para una conexión de canal.

public boolean IsConnected {get;}
Devuelve el valor de `isConnected`.

Si es true, se ha realizado una conexión con el gestor de colas y no se sabe que se ha interrumpido. Las llamadas a IsConnected no intentan activamente acceder al gestor de colas, por lo que es posible que se interrumpa la conectividad física, pero IsConnected todavía puede devolver true. El estado IsConnected sólo se actualiza cuando la actividad, por ejemplo, colocar un mensaje, obtener un mensaje, se realiza en el gestor de colas.

Si es false, no se ha realizado una conexión con el gestor de colas, se ha interrumpido o se ha desconectado.

public int KeepAlive {get;}
Especifica si se va a utilizar el recurso TCP KEEPALIVE para comprobar que el otro extremo de la conexión sigue estando disponible. Si no está disponible, el canal se cierra.

public int ListenerTimer {get;}
El intervalo de tiempo, en segundos, entre los intentos de WebSphere MQ de reiniciar el escucha después de una anomalía de APPC o TCP/IP.

public int LoggerEvent {get;}
Indica si se generan sucesos de registrador.

public string LU62ARMSuffix {get;}
El sufijo del miembro APPCPM de SYS1.PARMLIB. Este sufijo designa el LUADD de este iniciador de canal. Cuando el gestor de reinicio automático (ARM) reinicia el iniciador de canal, se emite el mandato z/OS SET APPC=xx.

public string LUGroupName {get; z/os}
El nombre de LU genérico que utilizará el escucha de LU 6.2 que maneja las transmisiones de entrada para el grupo de compartición de colas.

public string LUName {get;}
El nombre de la LU que se va a utilizar para las transmisiones de LU de salida 6.2 .

public int MaximumActiveChannels {get;}
Número máximo de canales que pueden estar activos en cualquier momento.

public int MaximumCurrentChannels {get;}
El número máximo de canales que pueden ser actuales en cualquier momento (incluidos los canales de conexión de servidor con clientes conectados).

public int MaximumLU62Channels {get;}
El número máximo de canales que pueden ser actuales, o clientes que se pueden conectar, que utilizan el protocolo de transmisión LU 6.2 .

public int MaximumMessageLength {get;}
Devuelve la longitud máxima de un mensaje (en bytes) que puede manejar el gestor de colas. No se puede definir ninguna cola con una longitud máxima de mensaje mayor que `MaximumMessageLength`.

public int MaximumPriority {get;}
Devuelve la prioridad máxima de mensajes soportada por el gestor de colas. Las prioridades van de cero (el más bajo) a este valor. Emite `MQException` si llama a este método después de desconectarse del gestor de colas.

public int MaximumTCPChannels {get;}
Número máximo de canales que pueden ser actuales, o clientes que se pueden conectar, que utilizan el protocolo de transmisión TCP/IP.

public int MQIAccounting {get;}
Controla la recopilación de la información contable para los datos de la Interfaz de Colas de Mensajes (MQI).

public int MQIStatistics {get;}
Controla la recopilación de la información sobre la supervisión de estadísticas para el gestor de colas.

public int OutboundPortMax {get;}
El valor máximo en el rango de números de puerto que se van a utilizar al enlazar canales de salida.

public int OutboundPortMin {get;}
El valor mínimo en el rango de números de puerto que se van a utilizar al enlazar canales de salida.

public int QueueAccounting {get;}
Indica si los datos de contabilidad de clase 3 (de nivel de hebra y de nivel de cola) se van a utilizar para todas las colas.

public int QueueMonitoring {get;}
Controla la recopilación de los datos de supervisión para las colas.

public int QueueStatistics {get;}
Controla la recopilación de los datos estadísticos para las colas.

public int ReceiveTimeout {get;}
El periodo de tiempo que un canal TCP/IP espera para recibir datos, incluidas las pulsaciones, de su asociado antes de volver al estado inactivo.

public int ReceiveTimeoutMin {get;}
El periodo mínimo de tiempo que un canal TCP/IP espera para recibir datos, incluidas las pulsaciones, de su asociado antes de volver a un estado inactivo.

public int ReceiveTimeoutType {get;}
El calificador que se aplica al valor en `ReceiveTimeout`.

public int SharedQueueQueueManagerName {get;}
Especifica cómo entregar mensajes a una cola compartida. Si la transferencia especifica un gestor de colas diferente del mismo grupo de compartición de colas que el gestor de colas de destino, el mensaje se entrega de dos maneras:

MQC.MQSQQM_USE
Los mensajes se entregan al gestor de colas de objetos antes de colocarlos en la cola compartida.

MQCMQSQQM_IGNORE
Los mensajes se colocan directamente en la cola compartida.

public int SSLEvent {get;}
Indica si se generan sucesos SSL.

public int SSLFips {get;}
Si sólo se van a utilizar algoritmos certificados por FIPS si se realiza la criptografía en WebSphere MQ, en lugar del hardware criptográfico.

public int SSLKeyResetCount {get;}
Indica el número de bytes no cifrados enviados y recibidos dentro de una conversación SSL antes de que se renegocie la clave secreta.

public int ClusterSenderStatistics {get;}
Especifica el intervalo, en minutos, entre recopilaciones consecutivas de estadísticas.

public int SyncpointAvailability {get;}
Indica si el gestor de colas da soporte a unidades de trabajo y puntos de sincronización con los métodos `MQQueue.get` y `MQQueue.put` .

public string TCPName {get;}
El nombre del único sistema TCP/IP, o el predeterminado, que se va a utilizar, en función del valor de TCPStackType.

public int TCPStackType {get;}
Especifica si el iniciador de canal sólo utiliza el espacio de direcciones TCP/IP especificado en TCPName. De forma alternativa, el iniciador de canal puede enlazar con cualquier dirección TCP/IP.

public int TraceRouteRecording {get;}
Controla el registro de la información de rastreo de ruta.

Métodos

public MQProcess AccessProcess(string *processName*, int *openOptions*);
public MQProcess AccessProcess(string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Genera `MQException`.

Acceda a un proceso de WebSphere MQ en este gestor de colas para consultar los atributos de proceso.

processName
El nombre del proceso que se va a abrir.

openOptions
Opciones que controlan la apertura del proceso. Las opciones válidas que se pueden añadir o combinar utilizando un OR a nivel de bit son:

- `MQC.MQOO_FAIL_IF_QUIESCING`
- `MQC.MQOO_INQUIRE`
- `MQC.MQOO_SET`
- `MQC.MQOO_ALTERNATE_USER_AUTHORITY`

queueManagerName
El nombre del gestor de colas en el que se define el proceso. Puede dejar un nombre de gestor de colas en blanco o nulo si el gestor de colas es el mismo que al que está accediendo el proceso.

alternateUserId

Si se especifica `MQC.MQOO_ALTERNATE_USER_AUTHORITY` en el parámetro *openOptions* , *alternateUserId* especifica el ID de usuario alternativo utilizado para comprobar la autorización de la acción. Si no se especifica `MQOO_ALTERNATE_USER_AUTHORITY` , *alternateUserId* puede estar en blanco o ser nulo.

Se utiliza la autorización de usuario predeterminada para la conexión con el gestor de colas si no se especifica `MQC.MQOO_ALTERNATE_USER_AUTHORITY` .

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 cola en este gestor de colas.

Puede obtener o examinar mensajes, colocar mensajes, consultar los atributos de la cola o establecer los atributos de la cola. Si la cola especificada es una cola modelo, se crea una cola local dinámica. Consulte el atributo name del objeto `MQQueue` resultante para averiguar el nombre de la cola dinámica.

queueName
Nombre de la cola que se va a abrir.

openOptions
Opciones que controlan la apertura de la cola.

MQC.MQOO_ALTERNATE_USER_AUTHORITY
Validar con el identificador de usuario especificado.

MQC.MQOO_BIND_AS_QDEF
Utilice el enlace predeterminado para la cola.

MQC.MQOO_BIND_NOT_FIXED
No enlazar a un destino específico.

MQC.MQOO_BIND_ON_OPEN
Manejador de enlace a destino cuando se abre la cola.

MQC.MQOO_BROWSE
Abra para examinar el mensaje.

MQC.MQOO_FAIL_IF_QUIESCING
Fallar si el gestor de colas se está desactivando temporalmente.

MQC.MQOO_INPUT_AS_Q_DEF
Abrir para obtener mensajes utilizando el valor predeterminado definido por la cola.

MQC.MQOO_INPUT_SHARED
Abierto para obtener mensajes con acceso compartido.

MQC.MQOO_INPUT_EXCLUSIVE
Abierto para obtener mensajes con acceso exclusivo.

MQC.MQOO_INQUIRE
Abrir para consulta necesaria si desea consultar propiedades.

MQC.MQOO_OUTPUT
Abierto para colocar mensajes.

MQC.MQOO_PASS_ALL_CONTEXT
Permitir que se pase todo el contexto.

MQC.MQOO_PASS_IDENTITY_CONTEXT
Permitir que se pase el contexto de identidad.

MQC.MQOO_SAVE_ALL_CONTEXT
Guardar contexto cuando se recupere el mensaje.

MQC.MQOO_SET
Abra para establecer los atributos necesarios si desea establecer las propiedades.

MQC.MQOO_SET_ALL_CONTEXT
Permite establecer todo el contexto.

MQC.MQOO_SET_IDENTITY_CONTEXT
Permite establecer el contexto de identidad.

queueManagerName
Nombre del gestor de colas en el que está definida la cola. Un nombre que está totalmente en blanco o es nulo indica el gestor de colas al que está conectado el objeto `MQQueueManager` .

dynamicQueueName
dynamicQueueName se ignora a menos que `queueName` especifique el nombre de una cola modelo. Si lo hace, *dynamicQueueName* especifica el nombre de la cola dinámica que se va a crear. Un nombre en blanco o nulo no es válido si `queueName` especifica el nombre de una cola modelo. Si el último carácter no en blanco del nombre es un asterisco, `*`, el gestor de colas sustituye el asterisco por una serie de caracteres. Los caracteres garantizan que el nombre generado para la cola es exclusivo en este gestor de colas.

alternateUserId
Si se especifica MQC.MQOO_ALTERNATE_USER_AUTHORITY en el parámetro openOptions , *alternateUserId* especifica el identificador de usuario alternativo que se utiliza para comprobar la autorización para la apertura. Si no se especifica MQC.MQOO_ALTERNATE_USER_AUTHORITY , *alternateUserId* puede dejarse en blanco o ser nulo.

```
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);
```

Acceda a un tema en este gestor de colas.

Los objetos de MQTopic están estrechamente relacionados con los objetos de tema administrativo, que a veces se denominan objetos de tema. En la entrada, topicObject apunta a un objeto de tema administrativo. El constructor MQTopic obtiene una serie de tema del objeto de tema y la combina con topicName para crear un nombre de tema. Uno o ambostopicObject o topicName pueden ser nulos. El nombre de tema se compara con el árbol de temas y el nombre del objeto de tema administrativo más coincidente se devuelve en topicObject.

Los temas que están asociados con el objeto MQTopic son el resultado de combinar dos series de tema. La primera serie de tema se define mediante el objeto de tema administrativo identificado por *topicObject*. La segunda serie de tema es *topicString*. La serie de tema resultante asociada con el objeto MQTopic puede identificar varios temas incluyendo comodines.

En función de si el tema está abierto para su publicación o suscripción, puede utilizar los métodos MQTopic.Put para publicar en temas, o los métodos MQTopic.Get para recibir publicaciones en temas. Si desea publicar y suscribirse al mismo tema, debe acceder al tema dos veces, una para publicar y otra para suscribirse.

Si crea un objeto MQTopic para la suscripción, sin proporcionar un objeto MQDestination , se presupone una suscripción gestionada. Si pasa una cola como un objeto MQDestination , se presupone una suscripción no gestionada. Debe asegurarse de que las opciones de suscripción que establezca sean coherentes con la suscripción que se está gestionando o que no está gestionada.

destination
destination es una instancia de MQQueue . Al proporcionar *destination*, MQTopic se abre como una suscripción no gestionada. Las publicaciones sobre el tema se entregan en la cola a la que se accede como *destination*.

topicName
Serie de tema que es la segunda parte del nombre de tema. *topicName* se concatena con la serie de tema definida en el objeto de tema administrativo *topicObject* . Puede establecer *topicName* en nulo, en cuyo caso el nombre de tema se define mediante la serie de tema en *topicObject*.

topicObject
En la entrada, *topicObject* es el nombre del objeto de tema que contiene la serie de tema que forma la primera parte del nombre de tema. La serie de tema en *topicObject* se concatena con *topicName*. Las reglas para construir nombres de tema se definen en Combinación de series de tema.

En la salida, *topicObject* contiene el nombre del objeto de tema administrativo que es la coincidencia más cercana en el árbol de temas con el tema identificado por el nombre de tema.

openAs
Acceda al tema para publicar o suscribirse. El parámetro sólo puede contener una de estas opciones:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

options

Combine las opciones que controlan la apertura del tema para publicación o suscripción. Utilice las constantes de MQC.MQSO_* para acceder a un tema para la suscripción y las constantes de MQC.MQOO_* para acceder a un tema para su publicación.

Si se necesita más de una opción, añada los valores juntos o combine los valores de opción utilizando el operador OR a nivel de bit.

alternateUserId
Especifique el ID de usuario alternativo que se utiliza para comprobar la autorización necesaria para finalizar la operación. Debe especificar *alternateUserId*, si MQC.MQOO_ALTERNATE_USER_AUTHORITY o MQC.MQSO_ALTERNATE_USER_AUTHORITY se ha establecido en el parámetro de opciones.

subscriptionName
subscriptionName es necesario si se proporcionan las opciones MQC.MQSO_DURABLE o MQC.MQSO_ALTER . En ambos casos, MQTopic se abre implícitamente para la suscripción. Se genera una excepción si se ha establecido MQC.MQSO_DURABLE y la suscripción existe, o si se ha establecido MQC.MQSO_ALTER y la suscripción no existe.

properties
Establezca cualquiera de las propiedades de suscripción especiales listadas utilizando una tabla hash. Las entradas especificadas en la tabla hash se actualizan con valores de salida. Las entradas no se añaden a la tabla hash para informar de los valores de salida.

- 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();

Genera MQException

Devuelve un objeto MQAsyncStatus , que representa la actividad asíncrona para la conexión del gestor de colas.

public void Backout();

Genera MQException.

Retroceder cualquier mensaje que se haya leído o escrito en el punto de sincronización desde el último punto de sincronización.

Los mensajes que se han escrito con el conjunto de distintivos MQC.MQPMO_SYNCPOINT se eliminan de las colas. Los mensajes leídos con el distintivo MQC.MQGMO_SYNCPOINT se restablecen en las colas de las que proceden. Si los mensajes son persistentes, se registran los cambios.

Para los clientes reconectables, el código de razón MQRC_NONE se devuelve a un cliente después de que la reconexión sea satisfactoria.

public void Begin();

Genera MQException.

Begin sólo está soportado en la modalidad de enlaces de servidor. Inicia una unidad de trabajo global.

public void Commit();

Genera MQException.

Confirme los mensajes que se hayan leído o grabado en el punto de sincronización desde el último punto de sincronización.

Los mensajes escritos con el conjunto de distintivos MQC.MQPMO_SYNCPOINT se ponen a disposición de otras aplicaciones. Los mensajes recuperados con el conjunto de distintivos MQC.MQGMO_SYNCPOINT se suprimen. Si los mensajes son persistentes, se registran los cambios.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- MQRC_CALL_INTERRUPTED si se pierde la conexión al realizar la llamada de confirmación.
- MQRC_BACKED_OUT si la llamada de confirmación se emite después de la reconexión.

Disconnect();

Genera MQException.

Cierre la conexión con el gestor de colas. Todos los objetos a los que se accede en este gestor de colas ya no son accesibles para esta aplicación. Para volver a acceder a los objetos, cree un objeto MQQueueManager .

Generalmente, se confirma cualquier trabajo realizado como parte de una unidad de trabajo. Sin embargo, si la unidad de trabajo está gestionada por .NET, es posible que la unidad de trabajo se retrotraiga.

```
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.

Coloca un solo mensaje en una cola o tema sin crear primero un objeto MQQueue o MQTopic .

queueName
El nombre de la cola en la que colocar el mensaje.

destinationName
El nombre de un objeto de destino. Es una cola o un tema en función del valor de *type*.

type
El tipo de objeto de destino. No debe combinar las opciones.

MQC.MQOT_Q
Cola

MQC.MQOT_TOPIC
Tema

queueManagerName

El nombre del gestor de colas o alias de gestor de colas, en el que se define la cola. Si se especifica el tipo `MQC.MQOT_TOPIC` , este parámetro se ignora.

Si la cola es una cola modelo y el nombre del gestor de colas resuelto no es este gestor de colas, se emite un `MQException` .

topicString

topicString se combina con el nombre de tema en el objeto de tema *destinationName* .

topicString se ignora si *destinationName* es una cola.

message
El mensaje a enviar. El mensaje es un objeto de entrada/salida.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- `MQRC_CALL_INTERRUPTED` si la conexión se interrumpe al realizar una llamada Put en un mensaje persistente.
- `MQRC_NONE` si la conexión es satisfactoria al realizar una llamada de colocación en un mensaje no persistente (consulte Recuperación de aplicaciones).

putMessageOptions

Opciones que controlan las acciones de la colocación.

Si omite *putMessageOptions*, se crea una instancia predeterminada de *putMessageOptions* . *putMessageOptions* es un objeto de entrada/salida.

Si utiliza la opción `MQPMO_LOGICAL_ORDER` en un cliente reconectable, se devuelve el código de razón `MQRC_RECONNECT_INCOMPATIBLE` .

alternateUserId

Especifica un identificador de usuario alternativo utilizado para comprobar la autorización al colocar el mensaje en una cola.

Puede omitir *alternateUserId* si no establece `MQC.MQOO_ALTERNATE_USER_AUTHORITY` en *putMessageOptions*. Si establece `MQC.MQOO_ALTERNATE_USER_AUTHORITY`, también debe establecer *alternateUserId*. *alternateUserId* no tiene efecto a menos que también establezca `MQC.MQOO_ALTERNATE_USER_AUTHORITY`.

Constructores

```
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 conexión con un gestor de colas. Seleccione entre crear una conexión de cliente o una conexión de servidor.

Debe tener autorización de consulta (inq) sobre el gestor de colas al intentar conectarse al gestor de colas. Sin autorización para inquire, el intento de conexión falla.

Se crea una conexión de cliente si se cumple una de las condiciones siguientes:

1. *channel* o *connName* se especifican en el constructor.
2. *HostName*, *Port*o *Channel* se especifican en *properties*.
3. Se especifican *MQEnvironment.HostName*, *MQEnvironment.Port*o *MQEnvironment.Channel* .

Los valores de las propiedades de conexión se establecen de forma predeterminada en el orden que se muestra. *channel* y *connName* en el constructor tienen prioridad sobre los valores de propiedad en el constructor. Los valores de propiedad de constructor tienen prioridad sobre las propiedades MQEnvironment .

El nombre de host, el nombre de canal y el puerto se definen en la clase MQEnvironment .

queueManagerName
Nombre del gestor de colas o grupo de gestores de colas al que conectarse.

Omita el parámetro, o déjelo nulo, o en blanco para realizar una selección de gestor de colas predeterminada. La conexión del gestor de colas predeterminado en un servidor es con el gestor de colas predeterminado en el servidor. La conexión del gestor de colas predeterminado en una conexión de cliente es con el gestor de colas al que está conectado el escucha.

options
Especifique las opciones de conexión de MQCNO . Los valores deben ser aplicables al tipo de conexión que se está realizando. Por ejemplo, si especifica las siguientes propiedades de conexión de servidor para una conexión de cliente, se genera un MQException .

- MQC.MQCNO_FASTPATH_BINDING
- MQC.MQCNO_STANDARD_BINDING

properties

El parámetro properties toma una serie de pares de clave/valor que alteran temporalmente las propiedades establecidas por MQEnvironment; consulte el ejemplo, “Alterar temporalmente las propiedades de MQEnvironment” en la página 1303. Se pueden alterar temporalmente las propiedades siguientes:

- 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

channel
Nombre de un canal de conexión de servidor

connName
Nombre de conexión con el formato *HostName* (*Puerto*).

Puede proporcionar una lista de *nombres de host* y *puertos* como argumento para el constructor `MQQueueManager(String queueManagerName, Hashtable properties)` utilizando CONNECTION_NAME_PROPERTY.

Por ejemplo:

```
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);
```

Cuando se realiza un intento de conexión, la lista de nombres de conexión se procesa en orden. Si el intento de conexión con el primer nombre de host y puerto falla, se intenta la conexión con el segundo par de atributos. El cliente repite este proceso hasta que se establece una conexión satisfactoria o se agota la lista. Si la lista está agotada, se devuelve un código de razón y un código de terminación adecuados a la aplicación cliente.

Cuando no se proporciona un número de puerto para el nombre de conexión, se utiliza el puerto predeterminado (configurado en `mqclient.ini`).

Establecer la lista de conexiones

Puede establecer la lista de conexiones utilizando los métodos siguientes cuando se establecen las opciones de reconexión automática de cliente:

Establecer la lista de conexiones mediante MQSERVER
Puede establecer la lista de conexiones a través del indicador de mandatos.

En el indicador de mandatos, establezca

```
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/Hostname1(Port1),Hostname2(Por2),Hostname3(Port3)
For Example:
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/fred.mq.com(5266),nick.mq.com(6566),jack.mq.com(8413)
```

Si establece la conexión en MQSERVER, no la establezca en la aplicación.

Si establece la lista de conexiones en la aplicación, la aplicación sobrescribe cualquier valor establecido en la variable de entorno MQSERVER.

Establecer la lista de conexiones a través de la aplicación
Puede establecer la lista de conexiones en la aplicación especificando el nombre de host y las propiedades de puerto.

```
String connName = "fred.mq.com(2344), nick.mq.com(3746), chris.mq.com(4288)";
MQQueueManager qm = new MQQueueManager("QM1", "TestChannel", connName);
```

Establezca la lista de conexiones a través de `app.config`

App.config es un archivo XML donde se especifican los pares de clave-valor.

En la lista de conexiones, especifique

```
<app.Settings>
<add key="Connection1" value="Hostname1(Port1)"/>
<add key="Connection2" value="Hostname2(Port2)"/>
<app.Settings>
```

Por ejemplo:

```
<app.Settings>
<add key>="Connection1" value="fred.mq.com(2966)"/>
<add key>="Connection2" value="alex.mq.com(6533)"/>
<app.Settings>
```

Puede cambiar directamente la lista de conexiones en el archivo app.config .

Establezca la lista de conexiones a través de `MQEnvironment`

Para establecer la lista de conexiones a través de MQEnvironment, utilice la propiedad *ConnectionName* .

```
MQEnvironment.ConnectionName = "fred.mq.com(4288),"alex.mq.com(5211);
```

La propiedad *ConnectionName* sobrescribe las propiedades de nombre de host y puerto establecidas en MQEnvironment.

Crear una conexión de cliente

El ejemplo siguiente muestra cómo crear una conexión de cliente con un gestor de colas. Puede crear una conexión de cliente estableciendo las variables MQEnvironment antes de crear un nuevo objeto 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 WebSphere 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 43. Conexión de cliente

Alterar temporalmente las propiedades de `MQEnvironment`

El ejemplo siguiente muestra cómo crear un gestor de colas con su ID de usuario y contraseña definidos en una tabla 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 44. Alteración temporal de las propiedades de `MQEnvironment`

Crear una conexión reconectable

El ejemplo siguiente muestra cómo volver a conectar automáticamente un cliente a un gestor de colas.

```
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
                                                    // 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 reconnect
happens

MQ QueueManager qmgr = new MQQueueManager("qmgrname", properties);
```

Figura 45. Reconexión automática de un cliente a un gestor de colas

Clase MQSubscription .NET

Utilice MQSubscription para solicitar que las publicaciones retenidas se envíen al suscriptor. MQSubscription es una propiedad de un objeto MQTopic abierto para suscripción.

Clase

```
System.Object
    └ IBM.WMQ.MQBase
        └ IBM.WMQ.MQBaseObject
            └ IBM.WMQ.MQManagedObject
                └ IBM.WMQ.MQSubscription
```

```
public class IBM.WMQ.MQSubscription extends IBM.WMQ.MQManagedObject;
```


- "Propiedades" en la página 1304
- "Métodos" en la página 1304
- "Constructores" en la página 1305

Propiedades

Acceda a las propiedades de suscripción utilizando la clase MQManagedObject ; consulte "Propiedades" en la página 1263.

Métodos

Acceda a los métodos de suscripción Inquire, Set y Get utilizando la clase MQManagedObject ; consulte "Métodos" en la página 1264.

public int RequestPublicationUpdate(int *options*);

Genera MQException.

Solicite una publicación actualizada para el tema actual. Si el gestor de colas tiene publicaciones retenidas para el tema, se envían al suscriptor.

Antes de llamar a RequestPublicationUpdate, abra un tema para la suscripción para obtener un objeto MQSubscription .

Normalmente, abra la suscripción con la opción MQC.MQSO_PUBLICATIONS_ON_REQUEST . Si no hay comodines en la serie de tema, sólo se envía una publicación como resultado de esta llamada. Si la serie de tema contiene comodines, es posible que se envíen muchas publicaciones. El método devuelve el número de publicaciones retenidas que se envían a la cola de suscripción. No hay garantía de que se reciban tantas publicaciones, especialmente si son mensajes no persistentes.

options

MQC.MQSRO_FAIL_IF_QUIESCING
El método falla si el gestor de colas está en un estado de inmovilización. En z/OS, para una aplicación CICS o IMS , `MQC.MQSRO_FAIL_IF_QUIESCING` también fuerza que el método falle si la conexión está en un estado inactivo.

MQC.MQSRO_NONE
No se especifica ninguna opción.

Constructores

No hay ningún constructor público .

Se devuelve un objeto `MQSubscription` en la propiedad `SubscriptionReference` de un objeto `MQTopic` que está abierto para suscripción.

Llame al método `RequestPublicationUpdate` . `MQSubscription` es una subclase de `MQManagedObject`. Utilice la referencia para acceder a las propiedades y métodos de `MQManagedObject`.

Clase MQTopic .NET

Utilice `MQTopic` para publicar o suscribir mensajes sobre un tema, o para consultar o establecer atributos de un tema. Cree un objeto `MQTopic` para publicar o suscribirse utilizando un constructor o el método `MQQueueManager.AccessTopic` .

Clase

```
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;
```


- “Propiedades” en la página 1305
- “Métodos” en la página 1306
- “Constructores” en la página 1308

Propiedades

Prueba de `MQException` que se genera al obtener las propiedades.

public Boolean IsDurable {get;}
Propiedad de sólo lectura que devuelve `True` si la suscripción es duradera o `False` de lo contrario. Si el tema se ha abierto para su publicación, la propiedad se ignora y siempre devolverá `False`.

public Boolean IsManaged {get;};
Propiedad de sólo lectura que devuelve `True` si la suscripción está gestionada por el gestor de colas, o `False` de lo contrario. Si el tema se ha abierto para su publicación, la propiedad se ignora y siempre devolverá False.

public Boolean IsSubscribed {get;};
Propiedad de sólo lectura que devuelve `True` si el tema se ha abierto para suscripción y `False` si el tema se ha abierto para publicación.

public MQSubscription SubscriptionReference {get;};
Propiedad de sólo lectura que devuelve el objeto MQSubscription asociado con un objeto de tema abierto para suscripción. La referencia está disponible si desea modificar las opciones de cierre o iniciar cualquiera de los métodos de objetos.

public MQDestination UnmanagedDestinationReference {get;};
Propiedad de sólo lectura que devuelve el MQQueue asociado a una suscripción no gestionada. Es el destino especificado cuando se creó el objeto de tema. La propiedad devuelve un valor nulo para los objetos de tema abiertos para publicación o con una suscripción gestionada.

Métodos

public void Put(MQMessage *message*);
public void Put(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);
Emite MQException.

Publica un mensaje en el tema.

Las modificaciones en el objeto MQMessage después de que se haya realizado la llamada Put no afectan al mensaje real en el tema de publicación o cola WebSphere MQ .

Put actualiza las propiedades MessageId y CorrelationId del objeto MQMessage y no borra los datos del mensaje. Las llamadas Put o Get adicionales hacen referencia a la información actualizada en el objeto MQMessage . Por ejemplo, en el siguiente fragmento de código, el primer mensaje contiene a y el segundo ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

message
Un objeto MQMessage que contiene los datos del descriptor de mensaje y el mensaje que se va a enviar. El descriptor de mensaje se puede modificar como consecuencia de este método. Los valores del descriptor de mensaje inmediatamente después de la finalización de este método son los valores que se colocaron en la cola o se publicaron en el tema.

Se devuelven los siguientes códigos de razón a un cliente reconectable:

- MQRC_CALL_INTERRUPTED si la conexión se interrumpe al ejecutar una llamada Put en un mensaje persistente y la reconexión se realiza correctamente.
- MQRC_NONE si la conexión es satisfactoria al ejecutar una llamada Put en un mensaje no persistente (consulte Recuperación de aplicaciones).

putMessageOptions
Opciones que controlan la acción de la colocación.

Si no se especifica *putMessageOptions* , se utiliza la instancia predeterminada de MQPutMessageOptons .

Si utiliza la opción MQPMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

Nota: Por motivos de simplicidad y rendimiento, si desea transferir un único mensaje a una cola, utilice el objeto MQQueueManager.Put . Debe tener un objeto MQQueue para esto.

public void Get(MQMessage *message*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*, int *MaxMsgSize*);
Emite MQException.

Recupera un mensaje del tema.

Este método utiliza una instancia predeterminada de MQGetMessageOptions para realizar la obtención. La opción de mensaje utilizada es MQGMO_NOWAIT.

Si la obtención falla, el objeto MQMessage no se modifica. Si se ejecuta correctamente, el descriptor de mensaje y las partes de datos de mensaje del MQMessage se sustituyen por el descriptor de mensaje y los datos de mensaje del mensaje de entrada.

Todas las llamadas a WebSphere MQ desde un MQQueueManager determinado son síncronas. Por lo tanto, si realiza una operación get con espera, todas las demás hebras que utilizan el mismo MQQueueManager no pueden realizar más llamadas WebSphere MQ hasta que se realice la llamada Get. Si necesita varias hebras para acceder a WebSphere MQ simultáneamente, cada hebra debe crear su propio objeto MQQueueManager .

message
Contiene el descriptor de mensaje y los datos de mensaje devueltos. Algunos de los campos del descriptor de mensaje son parámetros de entrada. Es importante asegurarse de que los parámetros de entrada MessageId y CorrelationId estén establecidos según sea necesario.

Un cliente reconectable devuelve el código de razón MQRC_BACKED_OUT después de una reconexión satisfactoria, para los mensajes recibidos en MQGM_SYNCPOINT.

getMessageOptions
Opciones que controlan la acción de la obtención.

El uso de la opción MQC.MQGMO_CONVERT puede dar como resultado una excepción con el código de razón MQC.MQRC_CONVERTED_STRING_TOO_BIG al convertir de códigos de caracteres de un solo byte a códigos de doble byte. En este caso, el mensaje se copia en el almacenamiento intermedio sin conversión.

Si no se especifica *getMessageOptions* , la opción de mensaje utilizada es MQGMO_NOWAIT.

Si utiliza la opción MQGMO_LOGICAL_ORDER en un cliente reconectable, se devuelve el código de razón MQRC_RECONNECT_INCOMPATIBLE .

MaxMsgSize
El mensaje más grande que este objeto de mensaje va a recibir. Si el mensaje en la cola es mayor que este tamaño, se produce una de las dos cosas siguientes:

- Si el distintivo MQGMO_ACCEPT_TRUNCATED_MSG se establece en el objeto MQGetMessageOptions , el mensaje se rellena con la mayor cantidad posible de datos de mensaje. Se genera una excepción con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_ACCEPTED .
- Si el distintivo MQGMO_ACCEPT_TRUNCATED_MSG no está establecido, el mensaje se deja en la cola. Se genera una excepción con el código de terminación MQCC_WARNING y el código de razón MQRC_TRUNCATED_MSG_FAILED .

Si no se especifica *MaxMsgSize* , se recupera todo el mensaje.

Constructores

```
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);
```

Acceda a un tema en *queueManager*.

Los objetos de MQTopic están estrechamente relacionados con los objetos de tema administrativo, que a veces se denominan objetos de tema. En la entrada, topicObject apunta a un objeto de tema administrativo. El constructor MQTopic obtiene una serie de tema del objeto de tema y la combina con topicName para crear un nombre de tema. Uno o ambostopicObject o topicName pueden ser nulos. El nombre de tema se compara con el árbol de temas y el nombre del objeto de tema administrativo más coincidente se devuelve en topicObject.

Los temas que están asociados con el objeto MQTopic son el resultado de combinar dos series de tema. La primera serie de tema se define mediante el objeto de tema administrativo identificado por *topicObject*. La segunda serie de tema es *topicString*. La serie de tema resultante asociada con el objeto MQTopic puede identificar varios temas incluyendo comodines.

En función de si el tema está abierto para su publicación o suscripción, puede utilizar los métodos MQTopic.Put para publicar en temas, o los métodos MQTopic.Get para recibir publicaciones en temas. Si desea publicar y suscribirse al mismo tema, debe acceder al tema dos veces, una para publicar y otra para suscribirse.

Si crea un objeto MQTopic para la suscripción, sin proporcionar un objeto MQDestination , se presupone una suscripción gestionada. Si pasa una cola como un objeto MQDestination , se presupone una suscripción no gestionada. Debe asegurarse de que las opciones de suscripción que establezca sean coherentes con la suscripción que se está gestionando o que no está gestionada.

queueManager
Gestor de colas sobre el que acceder a un tema.

destination
destination es una instancia de MQQueue . Al proporcionar *destination*, MQTopic se abre como una suscripción no gestionada. Las publicaciones sobre el tema se entregan en la cola a la que se accede como *destination*.

topicName
Serie de tema que es la segunda parte del nombre de tema. *topicName* se concatena con la serie de tema definida en el objeto de tema administrativo *topicObject* . Puede establecer *topicName* en nulo, en cuyo caso el nombre de tema se define mediante la serie de tema en *topicObject*.

topicObject
En la entrada, *topicObject* es el nombre del objeto de tema que contiene la serie de tema que forma la primera parte del nombre de tema. La serie de tema en *topicObject* se concatena con

topicName. Las reglas para construir nombres de tema se definen en Combinación de series de tema.

En la salida, *topicObject* contiene el nombre del objeto de tema administrativo que es la coincidencia más cercana en el árbol de temas con el tema identificado por el nombre de tema.

openAs
Acceda al tema para publicar o suscribirse. El parámetro sólo puede contener una de estas opciones:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

options

Combine las opciones que controlan la apertura del tema para publicación o suscripción. Utilice las constantes de MQC.MQSO_* para acceder a un tema para la suscripción y las constantes de MQC.MQOO_* para acceder a un tema para su publicación.

Si se necesita más de una opción, añada los valores juntos o combine los valores de opción utilizando el operador OR a nivel de bit.

alternateUserId
Especifique el ID de usuario alternativo que se utiliza para comprobar la autorización necesaria para finalizar la operación. Debe especificar *alternateUserId*, si MQC.MQOO_ALTERNATE_USER_AUTHORITY o MQC.MQSO_ALTERNATE_USER_AUTHORITY se ha establecido en el parámetro de opciones.

subscriptionName
subscriptionName es necesario si se proporcionan las opciones MQC.MQSO_DURABLE o MQC.MQSO_ALTER . En ambos casos, MQTopic se abre implícitamente para la suscripción. Se genera una excepción si se ha establecido MQC.MQSO_DURABLE y la suscripción existe, o si se ha establecido MQC.MQSO_ALTER y la suscripción no existe.

properties
Establezca cualquiera de las propiedades de suscripción especiales listadas utilizando una tabla hash. Las entradas especificadas en la tabla hash se actualizan con valores de salida. Las entradas no se añaden a la tabla hash para informar de los valores de salida.

- 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*);

Acceda a un tema en este gestor de colas.

Los objetos de MQTopic están estrechamente relacionados con los objetos de tema administrativo, que a veces se denominan objetos de tema. En la entrada, topicObject apunta a un objeto de tema administrativo. El constructor MQTopic obtiene una serie de tema del objeto de tema y la combina con topicName para crear un nombre de tema. Uno o ambostopicObject o topicName pueden ser nulos. El nombre de tema se compara con el árbol de temas y el nombre del objeto de tema administrativo más coincidente se devuelve en topicObject.

Los temas que están asociados con el objeto MQTopic son el resultado de combinar dos series de tema. La primera serie de tema se define mediante el objeto de tema administrativo identificado por *topicObject*. La segunda serie de tema es *topicString*. La serie de tema resultante asociada con el objeto MQTopic puede identificar varios temas incluyendo comodines.

En función de si el tema está abierto para su publicación o suscripción, puede utilizar los métodos MQTopic.Put para publicar en temas, o los métodos MQTopic.Get para recibir publicaciones en temas. Si desea publicar y suscribirse al mismo tema, debe acceder al tema dos veces, una para publicar y otra para suscribirse.

Si crea un objeto MQTopic para la suscripción, sin proporcionar un objeto MQDestination , se presupone una suscripción gestionada. Si pasa una cola como un objeto MQDestination , se presupone una suscripción no gestionada. Debe asegurarse de que las opciones de suscripción que establezca sean coherentes con la suscripción que se está gestionando o que no está gestionada.

destination
destination es una instancia de MQQueue . Al proporcionar *destination*, MQTopic se abre como una suscripción no gestionada. Las publicaciones sobre el tema se entregan en la cola a la que se accede como *destination*.

topicName
Serie de tema que es la segunda parte del nombre de tema. *topicName* se concatena con la serie de tema definida en el objeto de tema administrativo *topicObject* . Puede establecer *topicName* en nulo, en cuyo caso el nombre de tema se define mediante la serie de tema en *topicObject*.

topicObject
En la entrada, *topicObject* es el nombre del objeto de tema que contiene la serie de tema que forma la primera parte del nombre de tema. La serie de tema en *topicObject* se concatena con *topicName*. Las reglas para construir nombres de tema se definen en Combinación de series de tema.

En la salida, *topicObject* contiene el nombre del objeto de tema administrativo que es la coincidencia más cercana en el árbol de temas con el tema identificado por el nombre de tema.

openAs
Acceda al tema para publicar o suscribirse. El parámetro sólo puede contener una de estas opciones:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

options

Combine las opciones que controlan la apertura del tema para publicación o suscripción. Utilice las constantes de MQC.MQSO_* para acceder a un tema para la suscripción y las constantes de MQC.MQOO_* para acceder a un tema para su publicación.

Si se necesita más de una opción, añada los valores juntos o combine los valores de opción utilizando el operador OR a nivel de bit.

alternateUserId
Especifique el ID de usuario alternativo que se utiliza para comprobar la autorización necesaria para finalizar la operación. Debe especificar *alternateUserId*, si MQC.MQOO_ALTERNATE_USER_AUTHORITY o MQC.MQSO_ALTERNATE_USER_AUTHORITY se ha establecido en el parámetro de opciones.

subscriptionName
subscriptionName es necesario si se proporcionan las opciones MQC.MQSO_DURABLE o MQC.MQSO_ALTER . En ambos casos, MQTopic se abre implícitamente para la suscripción. Se genera una excepción si se ha establecido MQC.MQSO_DURABLE y la suscripción existe, o si se ha establecido MQC.MQSO_ALTER y la suscripción no existe.

properties
Establezca cualquiera de las propiedades de suscripción especiales listadas utilizando una tabla hash. Las entradas especificadas en la tabla hash se actualizan con valores de salida. Las entradas no se añaden a la tabla hash para informar de los valores de salida.

- 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

Interfaz .NET de IMQObjectTrigger

Implemente IMQObjectTrigger para procesar los mensajes pasados por el supervisor .NET de **runmqdnm** .

Interfaz

```
public interface IBM.WMQMonitor.IMQObjectTrigger();
```

En función de si se especifica el control de punto de sincronización en el mandato **runmqdnm** , el mensaje se elimina de la cola antes o después de que se devuelva el método Execute .

Métodos

void Execute (MQQueueManager *queueManager*, MQQueue *queue*, MQMessage *message*, string *param*);

queueManager
Gestor de colas que aloja la cola que se está supervisando.

queue
Cola que se está supervisando.

message
Mensaje leído de la cola.

param
Datos pasados desde `UserParameter`.

Interfaz .NET de MQC

Consulte una constante `MQI` añadiendo el prefijo `MQC.` al nombre de la constante. `MQC` define todas las constantes utilizadas por `MQI`.

Interfaz

```
System.Object
    │
    └─ IBM.WMQ.MQC
```

```
public interface IBM.WMQ.MQC extends System.Object;
```

Ejemplo

```
MQQueue queue;
queue.closeOptions = MQC.MQCO_DELETE;
```

Identificadores de juego de caracteres para aplicaciones .NET

Descripciones de los juegos de caracteres que puede seleccionar para codificar mensajes .NET IBM WebSphere MQ

Juego de caracteres	Descripción
37	ibm037
437	ibm437 /PC Original
500	ibm500
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 Griego

Juego de caracteres	Descripción
775	ibm775 /PC Báltico
813	iso-8859-7 /greek/ ibm813
838	ibm838
850	ibm850 /PC Latin 1
852	ibm852 /PC Latin 2
855	ibm855 /PC Cirílico
856	ibm856
857	ibm857 /PC Turco
860	ibm860 /PC Portugués
861	ibm861 /PC islandés
862	ibm862 /PC Hebreo
863	ibm863 /PC francés canadiense
864	ibm864 /PC árabe
865	ibm865 /PC Nordic
866	ibm866 /PC Ruso
868	ibm868
869	ibm869 /PC Modern Griego
870	ibm870
871	ibm871
874	ibm874
875	ibm875
912	iso-8859-2 / latin2 / ibm912
913	iso-8859-3 / latin3 / ibm913
914	iso-8859-4 / latin4 / ibm914
915	iso-8859-5 /cirílico/ ibm915
916	iso-8859-8 /hebreo/ ibm916
918	ibm918
920	iso-8859-9 / latin5 / ibm920
921	ibm921
922	ibm922
930	ibm930
932	PC japonés
933	ibm933
935	ibm935
937	ibm937

Juego de caracteres	Descripción
939	ibm939
942	ibm942
943	ibm943
948	ibm948
949	ibm949
950	ibm950 /Big 5 Chino tradicional
954	eucjis
964	ibm964 /CNS 11643 Chino tradicional
970	ibm970
1006	ibm1006
1025	ibm1025
1026	ibm1026
1089	iso-8859-6 /arabic/ ibm1089
1097	ibm1097
1098	ibm1098
1112	ibm1112
1122	ibm1122
1123	ibm1123
1124	ibm1124
1250	Windows Latín 2
1251	Windows cirílico
1252	Windows Latin 1
1253	Windows griego
1254	Windows turco
1255	Windows hebreo
1256	Windows árabe
1257	Windows báltico
1258	Windows Vietnamita
1381	ibm1381
1383	ibm1383
2022	JIS
5601	ksc-5601 Coreano
33722	ibm33722

IBM WebSphere MQ clases C++

Las clases C++ de IBM WebSphere MQ encapsulan la interfaz de cola de mensajes (MQI) de IBM WebSphere MQ . Hay un único archivo de cabecera C++, **imqi.hpp**, que cubre todas estas clases.

Para cada clase, se muestra la siguiente información:

Diagrama de jerarquía de clases
Un diagrama de clase que muestra la clase en su relación de herencia con sus clases padre inmediatas, si las hay.

Otras clases pertinentes
El documento enlaza con otras clases relevantes, como las clases padre, y las clases de objetos utilizados en las firmas de método.

Atributos de objetos
Atributos de la clase. Estos atributos se añaden a los definidos para cualquier clase padre. Muchos atributos reflejan miembros de estructura de datos de WebSphere MQ (consulte “Referencia cruzada de C++ y MQI” en la página 1316). Para obtener descripciones detalladas, consulte “objeto, atributos de” en la página 780.

Constructores
Firmas de los métodos especiales utilizados para crear un objeto de la clase.

Métodos de objeto (public)
Firmas de métodos que requieren una instancia de la clase para su operación y que no tienen restricciones de uso.

Cuando sea aplicable, también se mostrará la siguiente información:

Métodos de clase (public)
Firmas de métodos que no requieren una instancia de la clase para su operación y que no tienen restricciones de uso.

Métodos sobrecargados (clase padre)
Firmas de los métodos virtuales definidos en las clases padre, pero que muestran un comportamiento polimórfico diferente para esta clase.

Métodos de objeto (protegidos)
Firmas de métodos que requieren una instancia de la clase para su operación y que están reservadas para que las utilicen las implementaciones de clases derivadas. Esta sección sólo es de interés para los escritores de clase, en contraposición a los usuarios de clase.

Datos de objeto (protegidos)
Detalles de implementación para datos de instancia de objeto disponibles para las implementaciones de clases derivadas. Esta sección sólo es de interés para los escritores de clase, en contraposición a los usuarios de clase.

códigos de razón
Valores MQRC_ * (consulte Códigos de razón de API) que se pueden esperar de los métodos que fallan. Para obtener una lista exhaustiva de los códigos de razón que se pueden producir para un objeto de una clase, consulte la documentación de la clase padre. La lista documentada de códigos de razón para una clase no incluye los códigos de razón para las clases padre.

Nota:

1. Los objetos de estas clases no son seguros para hebras. Esto garantiza un rendimiento óptimo, pero tenga cuidado de no acceder a ningún objeto desde más de una hebra.
2. Se recomienda que, para un programa multihebra, se utilice un objeto de gestor ImqQueueindependiente para cada hebra. Cada objeto de gestor debe tener su propia colección independiente de otros objetos, asegurándose de que los objetos de distintas hebras estén aislados entre sí.

Las clases son:

- “Clase C++ de registro ImqAuthentication” en la página 1332

- “Clase C++ ImqBinary” en la página 1335
- “Clase C++ ImqCache” en la página 1336
- “Clase C++ ImqChannel” en la página 1340
- “Clase C++ de cabecera ImqCICSBridge” en la página 1345
- “Clase C++ ImqDeadLetterHeader” en la página 1351
- “ImqDistributionListar clase C++” en la página 1354
- “Clase C++ ImqError” en la página 1355
- “Clase C++ ImqGetMessageOptions” en la página 1356
- “Clase C++ ImqHeader” en la página 1360
- “Clase C++ de cabecera ImqIMSBridge” en la página 1361
- “Clase C++ ImqItem” en la página 1364
- “Clase C++ ImqMessage” en la página 1366
- “Clase C++ de rastreador ImqMessage” en la página 1373
- “Clase C++ ImqNamelist” en la página 1376
- “Clase C++ ImqObject” en la página 1377
- “Clase C++ ImqProcess” en la página 1383
- “Clase C++ ImqPutMessageOptions” en la página 1385
- “Clase C++ ImqQueue” en la página 1387
- “Clase C++ del gestor de ImqQueue” en la página 1398
- “Clase C++ de cabecera ImqReference” en la página 1414
- “Clase C++ ImqString” en la página 1417
- “Clase C++ ImqTrigger” en la página 1422
- “Clase C++ de cabecera ImqWork” en la página 1425

Referencia cruzada de C++ y MQI

Esta colección de temas contiene información relacionada con C++ para la MQI.

Lea esta información junto con “Tipos de datos utilizados en la MQI” en la página 218.

Esta tabla relaciona estructuras de datos MQI con las clases C++ y archivos de inclusión. Los temas siguientes muestran información de referencia cruzada para cada clase C++. Estas referencias cruzadas están relacionadas con el uso de las interfaces de procedimiento subyacentes de WebSphere MQ . Las clases ImqBinary, ImqDistributionList e ImqString no tienen atributos que entren en esta categoría y se excluyen.

Tabla 610. Estructura de datos, clase y referencia cruzada de archivo de inclusión

Estructura de datos	**Clase**	**Archivo Include**
MQAIRE	Registro ImqAuthentication	imqair.hpp
	ImqBinary	imqbin.hpp
	ImqCache	imqcac.hpp
MQCD	ImqChannel	imqchl.hpp
MQCIH	Cabecera ImqCICSBridge	imqcih.hpp
MQDLH	ImqDeadLetterHeader	imqdlh.hpp
MQOR	Lista de ImqDistribution	imqdst.hpp
	ImqError	imqerr.hpp

Tabla 610. Estructura de datos, clase y referencia cruzada de archivo de inclusión (continuación)		
Estructura de datos	**Clase**	**Archivo Include**
MQGMO	ImqGetMessageOptions	imqgmo.hpp
	ImqHeader	imqhdr.hpp
MQIIH	Cabecera ImqIMSBridge	imqiih.hpp
	ImqItem	imqitm.hpp
MQMD	ImqMessage	imqmsg.hpp
	Rastreador de 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	Gestor de ImqQueue	imqmgr.hpp
MQRMH	Cabecera ImqReference	imqrfh.hpp
	ImqString	imqstr.hpp
MQTM	ImqTrigger	imqtrg.hpp
MQTMC		
MQTMC2	ImqTrigger	imqtrg.hpp
MQXQH		
MQWIH	Cabecera ImqWork	imqwih.hpp

Referencia cruzada de registro ImqAuthentication

Referencia cruzada de atributos, estructuras de datos, campos y llamadas para la clase C++ de registro ImqAuthentication.

Atributo	**Estructura de datos**	**Campo**	**Llamada**
Nombre de conexión	MQAIRE	AuthInfoConnName	MQCONNX
contraseña	MQAIRE	LDAPPassword	MQCONNX
tipo	MQAIRE	AuthInfoType	MQCONNX
nombre de usuario	MQAIRE	LDAPUserNamePtr	MQCONNX
	MQAIRE	Desplazamiento de LDAPUserName	MQCONNX
	MQAIRE	LDAPUserNameLongitud	MQCONNX

Referencia cruzada de ImqCache

Referencia cruzada de atributos y llamadas para la clase C++ ImqCache .

Atributo	Llamada
almacenamiento intermedio automático	MQGET
Longitud de almacenamiento intermedio	MQGET
puntero de almacenamiento intermedio	MQGET, MQPUT
Longitud de datos	MQGET
Desplazamiento de datos	MQGET
puntero de datos	MQGET
Longitud del mensaje	MQGET, MQPUT

Referencia cruzada de ImqChannel

Referencia cruzada de atributos, estructuras de datos, campos y llamadas para la clase C++ ImqChannel .

Atributo	Estructura de datos	Campo	Llamada
latido del corazón por lotes	MQCD	BatchHeartbeat	MQCONNX
nombre de canal	MQCD	ChannelName	MQCONNX
Nombre de conexión	MQCD	ConnectionName	MQCONNX
	MQCD	Nombre de ShortConnection	MQCONNX
Compresión de cabecera	MQCD	Lista de HdrComp	MQCONNX
intervalo de latido cardíaco	MQCD	HeartbeatInterval	MQCONNX
Intervalo de mantenimiento activado	MQCD	Intervalo de mantenimiento de activación	MQCONNX
dirección local	MQCD	LocalAddress	MQCONNX
longitud máxima del mensaje	MQCD	MaxMsgLength	MQCONNX
Compresión de mensaje	MQCD	Lista de MsgComp	MQCONNX
nombre modalidad	MQCD	ModeName	MQCONNX
contraseña	MQCD	Contraseña	MQCONNX
recuento de salidas de recepción	MQCD		MQCONNX
nombres de salida de recepción	MQCD	ReceiveExit	MQCONNX
	MQCD	ReceiveExitsdefinido	MQCONNX
	MQCD	ReceiveExitPtr	MQCONNX
recibir datos de usuario	MQCD	ReceiveUserData	MQCONNX
	MQCD	ReceiveUserDataPtr	MQCONNX
Nombre de salida de seguridad	MQCD	SecurityExit	MQCONNX
datos de usuario de seguridad	MQCD	Datos de SecurityUser	MQCONNX
recuento de salidas de envío	MQCD		MQCONNX
nombres de salida de envío	MQCD	SendExit	MQCONNX
	MQCD	SendExitsdefinido	MQCONNX

Atributo	Estructura de datos	Campo	Llamada
	MQCD	SendExitPtr	MQCONNX
enviar datos de usuario	MQCD	SendUserData	MQCONNX
	MQCD	SendUserDataPtr	MQCONNX
CipherSpec de SSL	MQCD	Especificación sslCipher	MQCONNX
Tipo de autenticación de cliente SSL	MQCD	Autenticación de sslClient	MQCONNX
Nombre de igual SSL	MQCD	sslPeerName	MQCONNX
nombre del programa de transacciones	MQCD	TpName	MQCONNX
tipo de transporte	MQCD	TransportType	MQCONNX
ID de usuario	MQCD	UserIdentifier	MQCONNX

ImqCICSBridgeReferencia cruzada de cabecera

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ de cabecera ImqCICSBridge.

Atributo	Estructura de datos	Campo
código de terminación anómala de puente	MQCIH	AbendCode
Descriptor ADS	MQCIH	AdsDescriptor
identificador de atención	MQCIH	AttentionId
autenticador	MQCIH	Authenticator
código de terminación de puente	MQCIH	Código de BridgeCompletion
desplazamiento de error de puente	MQCIH	ErrorOffset
código de razón de puente	MQCIH	BridgeReason
código de cancelación de puente	MQCIH	CancelCode
tarea conversacional	MQCIH	ConversationalTask
Posición del cursor	MQCIH	CursorPosition
señal de recurso	MQCIH	Recurso
tiempo de mantenimiento del recurso	MQCIH	FacilityKeepTime
recurso como	MQCIH	FacilityLike
función	MQCIH	Función
obtener intervalo de espera	MQCIH	GetWaitInterval
Tipo de enlace	MQCIH	LinkType
identificador de transacción siguiente	MQCIH	NextTransactionId
longitud de datos de salida	MQCIH	OutputDataLength
formato de respuesta	MQCIH	ReplyToFormat
código de retorno de puente	MQCIH	ReturnCode
código de inicio	MQCIH	StartCode

Atributo	Estructura de datos	Campo
estado de finalización de tarea	MQCIH	TaskEndStatus
Identificador de transacción	MQCIH	TransactionId
control uow	MQCIH	UowControl
versión	MQCIH	Versión

Referencia cruzada ImqDeadLetterHeader

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ ImqDeadLetterHeader .

Atributo	Estructura de datos	Campo
código de razón de mensaje no enviado	MQDLH	Razón
Nombre del gestor de colas de destino	MQDLH	DestQMgrName
Nombre de la cola de destino	MQDLH	DestQName
Nombre de aplicación de transferencia	MQDLH	PutApplName
Tipo de aplicación de transferencia	MQDLH	PutApplType
Fecha de transferencia	MQDLH	PutDate
Hora de transferencia	MQDLH	PutTime

Referencia cruzada ImqError

Referencia cruzada de atributos y llamadas para la clase C++ ImqError .

Atributo	Llamada
código de terminación	MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET
código de razón	MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET

Referencia cruzada ImqGetMessageOptions

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ ImqGetMessageOptions .

Atributo	Estructura de datos	Campo
Estado de grupo	MQGMO	GroupStatus
opciones de coincidencia	MQGMO	MatchOptions
token de mensaje	MQGMO	MessageToken
opciones	MQGMO	Opciones
Nombre de cola resuelto	MQGMO	ResolvedQName
longitud devuelta	MQGMO	ReturnedLength
segmentación	MQGMO	Segmentation
estado de segmento	MQGMO	SegmentStatus
	MQGMO	Signal1

Atributo	Estructura de datos	Campo
	MQGMO	Signal2
participación de punto de sincronismo	MQGMO	Opciones
Intervalo de espera	MQGMO	WaitInterval

Referencia cruzada de ImqHeader

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ ImqHeader .

Atributo	Estructura de datos	Campo
juego de caracteres	MQDLH, MQIIH	CodedCharSetId
codificación	MQDLH, MQIIH	Codificación
formato	MQDLH, MQIIH	Formato
distintivos de cabecera	MQIIH, MQRMH	Distintivos

Referencia cruzada de cabecera ImqIMSBridge

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ de registro ImqAuthentication.

Atributo	Estructura de datos	Campo
autenticador	MQIIH	Authenticator
Modalidad de confirmación	MQIIH	CommitMode
Alteración temporal del terminal lógico	MQIIH	LTermOverride
Nombre de la correlación de servicios del formato del mensaje	MQIIH	MFSMapName
formato de respuesta	MQIIH	ReplyToFormat
Ámbito de seguridad	MQIIH	SecurityScope
id de instancia de transacción	MQIIH	TranInstanceId
ESTADO DE TRANSACCIÓN	MQIIH	TranState

Referencia cruzada de ImqItem

Referencia cruzada de atributos y llamadas para la clase C++ ImqItem .

Atributo	Llamada
id de estructura	MQGET

Referencia cruzada de ImqMessage

Referencia cruzada de atributos, estructuras de datos, campos y llamadas para la clase C++ ImqMessage .

Atributo	Estructura de datos	Campo	Llamada
Datos de ID de aplicación	MQMD	ApplIdentityData	
Datos de origen de aplicación	MQMD	ApplOriginData	

Atributo	Estructura de datos	Campo	Llamada
Recuento de restituciones	MQMD	BackoutCount	
juego de caracteres	MQMD	CodedCharSetId	
codificación	MQMD	Codificación	
caducidad	MQMD	Caducidad	
formato	MQMD	Formato	
Distintivos de mensaje	MQMD	MsgFlags	
tipo de mensaje	MQMD	MsgType	
offset	MQMD	Desplazamiento	
Longitud original	MQMD	OriginalLength	
persistencia	MQMD	Persistence	
priority	MQMD	Priority	
Nombre de aplicación de transferencia	MQMD	PutApplName	
Tipo de aplicación de transferencia	MQMD	PutApplType	
Fecha de transferencia	MQMD	PutDate	
Hora de transferencia	MQMD	PutTime	
Nombre de gestor de cola de respuestas	MQMD	GestorColasRespuesta	
Nombre de cola de respuestas	MQMD	ReplyToQ	
informe	MQMD	Informe	
número de secuencia	MQMD	MsgSeqNumber	
longitud total de mensaje		DataLength	MQGET
ID de usuario	MQMD	UserIdentifier	

Referencia cruzada de ImqMessageTracker

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ de rastreador ImqMessage.

Atributo	Estructura de datos	Campo
Señal de contabilidad	MQMD	AccountingToken
ID de correlación	MQMD	CorrelId
información de retorno	MQMD	Comentarios
ID de grupo	MQMD	GroupId
ID de mensaje	MQMD	MsgId

Referencia cruzada de ImqNamelist

Referencia cruzada de atributos, consultas y llamadas para la clase C++ ImqNamelist .

Atributo	Consulta	Llamada
Número de nombres	MQIA_NAME_COUNT	MQINQ

Atributo	Consulta	Llamada
Nombre de lista de nombres	MQCA_NAMELIST_NAME	MQINQ

Referencia cruzada de ImqObject

Referencia cruzada de atributos, estructuras de datos, campos, consultas y llamadas para la clase C++ ImqObject .

Atributo	Estructura de datos	Campo	Consulta	Llamada
Fecha de modificación			MQCA_ALTERATION_DATE	MQINQ
Hora de modificación			MQCA_ALTERATION_TIME	MQINQ
ID de usuario alternativo	MQOD	AlternateUserId		
ID de seguridad alternativo				
opciones de cierre				MQCLOSE
description			MQCA_Q_DESC, MQCA_Q_MGR_DESC, MQCA_PROCESS_DESC	MQINQ
nombre	MQOD	ObjectName	MQCA_Q_MGR_NAME, MQCQ_Q_NAME, MQCA_PROCESS_NAME	MQINQ
Opciones abiertas				MQOPEN
estado abierto				MQOPEN, MQCLOSE
identificador de gestor de colas	identificador de gestor de colas		MQCA_Q_MGR_IDENTIFIER	MQINQ

Referencia cruzada de ImqProcess

Referencia cruzada de atributos, consultas y llamadas para la clase C++ de registro ImqAuthentication.

Atributo	Consulta	Llamada
ID de aplicación	MQCA_APPL_ID	MQINQ
Tipo de aplicación	MQIA_APPL_TYPE	MQINQ
Datos de entorno	MQCA_ENV_DATOS	MQINQ
datos de usuario	MQCA_USER_DATA	MQINQ

ImqPutMessageOptions referencia cruzada

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ de registro ImqAuthentication.

Tabla 611. Referencia cruzada ImqPutMessageOptions		
Atributo	**Estructura de datos**	**Campo**
referencia de contexto	MQPMO	Contexto
	MQPMO	InvalidDestCount
	MQPMO	KnownDestCount
opciones	MQPMO	Opciones
campos de registro	MQPMO	PutMsgRecFields
Nombre del gestor de colas resuelto	MQPMO	ResolvedQMgrName
Nombre de cola resuelto	MQPMO	ResolvedQName
	MQPMO	Tiempo de espera
	MQPMO	UnknownDestCount
participación de punto de sincronismo	MQPMO	Opciones

Referencia cruzada de ImqQueue

Referencia cruzada de atributos, estructuras de datos, campos, consultas y llamadas para la clase C++ ImqQueue .

Tabla 612. Referencia cruzada de ImqQueue				
Atributo	**Estructura de datos**	**Campo**	**Consulta**	**Llamada**
Nombre de reposición en cola para restitución			MQCA_BACKOUT_REQ_Q_NAME	MQINQ
umbral de restituciones			Umbral MQIA_BACKOUT_THRESHOLD	MQINQ
nombre de cola base			MQCA_BASE_Q_NAME	MQINQ
Nombre de clúster			NOMBRE_CLÚSTER MQCA_clúster	MQINQ
Nombre de la lista de nombres del clúster			MQCA_XX_ENCODE_CASE_ONE nombre_clúster	MQINQ
Rango de carga trabajo del clúster			MQIA_CLWL_Q_RANK	MQINQ
Prioridad de carga de trabajo del clúster			MQIA_CLWL_Q_PRIORITY	MQINQ
Cola de uso de carga de trabajo de clúster			MQIA_CLWL_USEQ	MQINQ
fecha de creación			MQCA_CREATION_DATE	MQINQ
Hora de creación			MQCA_CREATION_TIME	MQINQ
Profundidad actual			MQIA_PROFUNDIDAD_Q_ACTUAL	MQINQ
enlace predeterminado			MQIA_DEF_BIND	MQINQ

Tabla 612. Referencia cruzada de ImqQueue (continuación)

Atributo	Estructura de datos	Campo	Consulta	Llamada
Opción abierta de entrada predeterminada			MQIA_DEF_INPUT_OPEN_OPTION	MQINQ
Persistencia predeterminada			MQIA_PERSISTENCIA	MQINQ
Prioridad predeterminada			MQIA_PRIORIDAD_DEF_PRIORIDAD	MQINQ
Tipo de definición			MQIA_DEFINITION_TYPE	MQINQ
suceso de profundidad alta			MQIA_Q_DEPTH_HIGH_EVENT	MQINQ
límite alto de profundidad			MQIA_Q_DEPTH_HIGH_LIMIT	MQINQ
suceso de profundidad baja			MQIA_Q_DEPTH_LOW_EVENT	MQINQ
límite bajo de profundidad			MQIA_Q_DEPTH_LOW_LIMIT	MQINQ
suceso de profundidad máxima			MQIA_Q_DEPTH_MAX_LIMIT	MQINQ
listas de distribución			MQIA_DIST_LISTS	MQINQ, MQSET
Nombre de la cola dinámica	MQOD	DynamicQName		
Copia en disco de restitución de obtención			MQIA_HARDEN_GET_BACKOUT	MQINQ
Tipo de índice			TIPO_ÍNDICE	MQINQ
inhibir obtención			INHIBIDORES DE MQIA_GET	MQINQ, MQSET
inhibir colocación			INHIBIDORES DE MQIA_PUT	MQINQ, MQSET
Nombre cola iniciación			MQCA_INITIATION_Q_NAME	MQINQ
Profundidad máxima			MQIA_MAX_Q_DEPTH	MQINQ
longitud máxima del mensaje			MQIA_MAX_MSG_LENGTH	MQINQ
Secuencia de entrega de mensajes			MQIA_MSG_DELIVERY_SEQUENCE	MQINQ
siguiente cola distribuida				

Tabla 612. Referencia cruzada de ImqQueue (continuación)

Atributo	Estructura de datos	Campo	Consulta	Llamada
Clase de mensajes no permanentes			MQIA_NPM_CLASS	MQINQ
Cuenta de entradas abiertas			MQIA_RECUENTO_ENTRADA_ABIERTA	MQINQ
Cuenta de salidas abiertas			MQIA_RECUENTO_SALIDA_ABIERTA	MQINQ
cola distribuida anterior				
nombre de proceso			MQCA_PROCESS_NAME	MQINQ
Contabilidad de la cola			MQIA_CUENTA_Q	MQINQ
nombre del gestor de colas	MQOD	ObjectQMgrName		
Supervisión de la cola			MQIA_MONITORING_Q	MQINQ
estadísticas de cola			MQIA_ESTADÍSTICAS_Q	MQINQ
Tipo de cola			MQIA_Q_TYPE	MQINQ
Nombre de gestor de colas remoto			MQCA_REMOTE_Q_MGR_NAME	MQINQ
Nombre de cola remota			MQCA_REMOTE_Q_NAME	MQINQ
Nombre del gestor de colas resuelto	MQOD	ResolvedQMgrName		
Nombre de cola resuelto	MQOD	ResolvedQName		
Intervalo de retención			MQIA_INTERVALO_RETENCIÓN	MQINQ
ámbito			MQIA_ÁMBITO	MQINQ
intervalo de servicio			MQIA_Q_SERVICE_INTERVAL	MQINQ
suceso de intervalo de servicio			MQIA_Q_SERVICE_INTERVAL_EVENT	MQINQ
Posibilidad de compartición			MQIA_COMPARTIBILIDAD	MQINQ
clase de almacenamiento			CLASE_ALMACENAMIENTO_MQCA	MQINQ
Nombre de cola de transmisión			MQCA_XMIT_Q_NAME	MQINQ
Activar control			MQIA_TRIGGER_CONTROL	MQINQ, MQSET

Tabla 612. Referencia cruzada de ImqQueue (continuación)				
Atributo	**Estructura de datos**	**Campo**	**Consulta**	**Llamada**
Datos desencadenantes			MQCA_TRIGGER_DATA	MQINQ, MQSET
Profundidad de desencadenante			MQIA_PROFUNDIDAD	MQINQ, MQSET
Prioridad de mensajes desencadenantes			MQIA_TRIGGER_MSG_PRIORITY	MQINQ, MQSET
tipo de activador			MQIA_TIPO_TRIGGER_TYPE	MQINQ, MQSET
utilización			MQIA_USAGE	MQINQ

Referencia cruzada de ImqQueueManager

Referencias cruzadas de atributos, estructuras de datos, campos, consultas y llamadas para la clase C++ del gestor ImqQueue.

Atributo	**Estructura de datos**	**Campo**	**Consulta**	**Llamada**
alteración temporal de conexiones de contabilidad			MQIA_ACCOUNTING_CONN_OVERRIDE	MQINQ
Intervalo contable			MQIA_INTERVALO_CONTABILIDAD	MQINQ
registro de actividad			MQIA_GRABACIÓN_ACTIVIDAD	MQINQ
Adoptar nueva comprobación MCA			MQIA_ADOPTNEWMCA_CHECK	MQINQ
Adoptar nuevo tipo MCA			MQIA_ADOPTNEWMCA_TYPE	MQINQ
Tipo de autenticación	MQCSP	AuthenticationType		MQCONNX
suceso de autorización			SUCESO_AUTORIZACIÓN_MQIA_EVENT	MQINQ
Opciones de inicio	MQBO	Opciones		MQBEGIN
suceso de puente			MQIA_SUCESO_PUENTE	MQINQ
Definición automática de canal			MQIA_CANAL_AUTO_DEF	MQINQ
suceso de definición automática de canal			MQIA_SUCESO_AUTO_CANAL	MQIA

Atributo	Estructura de datos	Campo	Consulta	Llamada
Salida de definición automática de canal			MQIA_SALIDA_AUTO_CANAL	MQIA
suceso de canal			MQIA_SUCESO_CANAL	MQINQ
Adaptadores del iniciador de canal			MQIA_CHINIT_ADAPTERS	MQINQ
Control del iniciador de canal			CONTROL MQIA_CHINIT_DE	MQINQ
Asignadores de tareas del iniciador de canal			MQIA_CHINIT_DISPATCHERS	MQINQ
Inicio automático del rastreo del iniciador de canal			MQIA_CHINIT_TRACE_AUTO_START	MQINQ
Tamaño de tabla de rastreo del iniciador de canal			MQIA_CHINIT_TRACE_TABLE_SIZE	MQINQ
Supervisión de canal			MQIA_MONITORING_CHANNEL	MQINQ
referencia de canal	MQCD	ChannelType		MQCONNX
Estadísticas del canal			MQIA_STATISTICS_CHANNEL	MQINQ
juego de caracteres			MQIA_CODED_CHAR_SET_ID	MQINQ
Supervisión de clúster emisor			MQIA_MONITORING_AUTO_CLUSSDR	MQINQ
Estadística de clúster emisor			MQIA_STATISTICS_AUTO_CLUSSDR	MQINQ
Datos de carga de trabajo de clúster			MQCA_CLUSTER_WORKLOAD_DATA	MQINQ
salida de carga de trabajo de clúster			MQCA_CLUSTER_WORKLOAD_EXIT	MQINQ
Longitud de la carga de trabajo de clúster			MQIA_CLUSTER_WORKLOAD_LENGTH	MQINQ
mru de carga de trabajo de clúster			MQIA_CLWL_MRU_CHANNELS	MQINQ
Cola de uso de carga de trabajo de clúster			MQIA_CLWL_USEQ	MQINQ
suceso de mandato			MQIA_SUCESO_MANDATO	MQINQ

Atributo	Estructura de datos	Campo	Consulta	Llamada
Nombre de cola de entrada de mandatos			MQCA_COMMAND_INPUT_Q_NAME	MQINQ
Nivel de mandato			MQIA_COMMAND_LEVEL	MQINQ
Control del servidor de mandatos			MQIA_CMD_SERVER_CONTROL	MQINQ
Opciones de conexión	MQCNO	Opciones		MQCONN, MQCONNX
ID de conexión	MQCNO	ConnectionId		MQCONNX
estado de conexión				MQCONN, MQCONNX, MQDISC
etiqueta de conexión	MQCD	ConnTag		MQCONNX
hardware de cifrado	MQSCO	CryptoHardware		MQCONNX
nombre de cola de mensajes no entregados			MQCA_DEAD_LETTER_Q_NAME	MQINQ
nombre de cola de transmisión predeterminada			MQCA_DEF_XMIT_Q_NAME	MQINQ
listas de distribución			MQIA_DIST_LISTS	MQINQ
grupo dns			MQCA_DNS_GROUP	MQINQ
dns wlm			MQIA_DNS_WLM	MQINQ
primer registro de autenticación	MQSCO	AuthInfoRecOffset		MQCONNX
	MQSCO	AuthInfoRecPtr		MQCONNX
suceso de inhibición			SUCESO INHIBIDOR DE MQIA_ENT	MQINQ
Versión de dirección IP			MQIA_IP_ADDRESS_VERSION	MQINQ
repositorio de claves	MQSCO	KeyRepository		MQCONNX
recuento de restablecimiento de clave	MQSCO	Recuento de KeyReset		MQCONNX
Temporizador de escucha			MQIA_LISTENER_TIMER	MQINQ
suceso local			MQIA_SUCESO_LOCAL	MQINQ

Atributo	Estructura de datos	Campo	Consulta	Llamada
LoggerEvent			MQIA_LOGGER_EVENT	MQINQ
Nombre de grupo LU			MQCA_LU_XX_ENCODE_CASE_ONE nombre_grupo	MQINQ
Nombre de LU			MQCA_LU_NAME	MQINQ
Sufijo de brazo lu62			MQCA_LU62_ARM_SUFFIX	MQINQ
Canales lu62			MQIA_LU62_CHANNELS	MQINQ
máximo de canales activos			MQIA_CANALES_ACTIVOS	MQINQ
Canales máximos			MQIA_MÁX_CANALES	MQINQ
máximo de manejadores			MQIA_MÁX_DESCRIPTORES de contexto	MQINQ
longitud máxima del mensaje			MQIA_MAX_MSG_LENGTH	MQINQ
Prioridad máxima			MQIA_PRIORIDAD_MÁX	MQINQ
Máx. mensajes no confirmados			MQIA_MAX_UNCOMMITTED_MSGS	MQINQ
Contabilidad de MQI			MQIA_CUENTA_MQI	MQINQ
Estadísticas MQI			MQIA_ESTADÍSTICAS_MQI	MQINQ
máximo de puerto de salida			MQIA_PUERTO_SALIDA máx	MQINQ
mínimo de puerto de salida			MQIA_PUERTO_SALIDA min	MQINQ
contraseña	MQCSP	CSPPasswordPtr		MQCONNX
	MQCSP	CSPPasswordOffset		MQCONNX
	MQCSP	CSPPasswordLength		MQCONNX
suceso de rendimiento			MQIA_SUCESO_RENDIMIENTO	MQINQ
plataforma			MQIA_PLATFORM	MQINQ
Contabilidad de la cola			MQIA_CUENTA_Q	MQINQ
Supervisión de la cola			MQIA_MONITORING_Q	MQINQ
estadísticas de cola			MQIA_ESTADÍSTICAS_Q	MQINQ
Tiempo de espera de recepción			MQIA_TIEMPO_ESPERA	MQINQ

Atributo	Estructura de datos	Campo	Consulta	Llamada
tiempo de espera de recepción mínimo			MQIA_RECEIVE_TIMEOUT_MIN	MQINQ
Tipo de tiempo de espera de recepción			MQIA_RECEIVE_TIMEOUT_TYPE	MQINQ
suceso remoto			MQIA_REMOTE_EVENT	MQINQ
Nombre de depósito			MQCA_REPOSITORY_NAME	MQINQ
Lista nombres repositorio			MQCA_REPOSITORY_NAMELIST	MQINQ
nombre de gestor de colas compartido			MQIA_SHARED_Q_Q_MGR_NAME	MQINQ
suceso ssl			MQIA_SSL_EVENT	MQINQ
Fips ssl			MQIA_SSL_FIPS_REQUIRED	MQINQ
Cuenta restablecimiento clave SSL			MQIA_SSL_RESET_COUNT	MQINQ
suceso de inicio-detención			MQIA_START_STOP_EVENT	MQINQ
Intervalo de estadísticas			MQIA_INTERVALO_ESTADÍSTICAS	MQINQ
Disponibilidad de punto de sincronismo			MQIA_SYNCPOINT	MQINQ
canales tcp			MQIA_TCP_CANALES	MQINQ
Keep alive de TCP			MQIA_TCP_KEEP_ALIVE	MQINQ
Nombre TCP			MQCA_TCP_NAME	MQINQ
Tipo de pila TCP			MQIA_TCP_STACK_TYPE	MQINQ
Registro de la ruta de rastreo			MQIA_GRABACIÓN_RUTA_RASTREO	MQINQ
Activar intervalo			MQIA_INTERVALO_DESENCADENANTE	MQINQ
ID de usuario	MQCSP	CSPUserIdPtr		MQCONNX
	MQCSP	CSPUserIdDesplazamiento		MQCONNX
	MQCSP	CSPUserIdLongitud		MQCONNX

ImqReferenceReferencia cruzada de cabecera

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ de registro ImqAuthentication.

Atributo	Estructura de datos	Campo
entorno de destino	MQRMH	DestEnvLongitud, DestEnvDesplazamiento
nombre de destino	MQRMH	DestNameLongitud, DestNameDesplazamiento
ID de instancia	MQRMH	ObjectInstanceId
longitud lógica	MQRMH	DataLogicalLength
desplazamiento lógico	MQRMH	DataLogicalOffset
desplazamiento lógico 2	MQRMH	DataLogicalOffset2
Tipo de referencia	MQRMH	ObjectType
Entorno de origen	MQRMH	SrcEnvLongitud, SrcEnvDesplazamiento
Nombre del origen	MQRMH	SrcNameLongitud, SrcNameDesplazamiento

Referencia cruzada de ImqTrigger

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ de registro ImqAuthentication.

Tabla 613. Referencia cruzada de ImqTrigger

Atributo	Estructura de datos	Campo
ID de aplicación	MQTM	ApplId
Tipo de aplicación	MQTM	ApplType
Datos de entorno	MQTM	EnvData
nombre de proceso	MQTM	ProcessName
nombre de cola	MQTM	QName
Datos desencadenantes	MQTM	TriggerData
datos de usuario	MQTM	UserData

Referencia cruzada de cabecera ImqWork

Referencia cruzada de atributos, estructuras de datos y campos para la clase C++ de registro ImqAuthentication.

Atributo	Estructura de datos	Campo
token de mensaje	MQWIH	MessageToken
nombre de servicio	MQWIH	ServiceName
paso de servicio	MQWIH	ServiceStep

Clase C++ de registro ImqAuthentication

Esta clase encapsula un registro de información de autenticación (MQAIR) para su uso durante la ejecución del método ImqQueueManager: :connect, para conexiones de cliente SSL personalizadas.

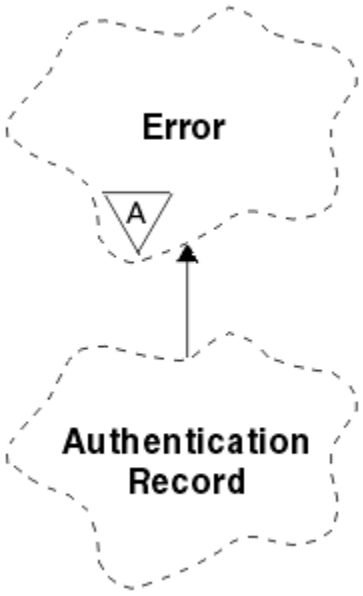


Figura 46. Clase de registro ImqAuthentication

Consulte la descripción del método ImqQueueManager: :connect para obtener más detalles. Esta clase no está disponible en la plataforma z/OS .

- "Atributos de objetos" en la página 1333
- "Constructores" en la página 1333
- "Métodos de objeto (public)" en la página 1334
- "Métodos de objeto (protegidos)" en la página 1334

Atributos de objetos

Nombre de conexión
El nombre de la conexión con el servidor CRL de LDAP. Es la dirección IP o el nombre DNS, seguido opcionalmente por el número de puerto, entre paréntesis.

referencia de conexión
Referencia a un objeto de gestor ImqQueueque proporciona la conexión necesaria a un gestor de colas (local). El valor inicial es cero. No confunda esto con el nombre del gestor de colas que identifica un gestor de colas (posiblemente remoto) para una cola con nombre.

siguiente registro de autenticación
Siguiente objeto de esta clase, en ningún orden concreto, que tenga la misma **referencia de conexión** que este objeto. El valor inicial es cero.

contraseña
Una contraseña proporcionada para la autenticación de conexión con el servidor CRL de LDAP.

registro de autenticación anterior
Objeto anterior de esta clase, en ningún orden concreto, que tenga la misma **referencia de conexión** que este objeto. El valor inicial es cero.

type
El tipo de información de autenticación contenida en el registro.

nombre de usuario
Un identificador de usuario proporcionado para la autorización al servidor CRL de LDAP.

Constructores

Registro ImqAuthentication();
El constructor predeterminado.

Métodos de objeto (public)

void operator = (const ImqAuthenticationRecord & *aire*);
Copia datos de instancia de *air*, sustituyendo los datos de instancia existentes.

const ImqString & connectionName () Const.
Devuelve el **nombre de conexión**.

void setConnectionName (const ImqString & *nombre*);
Establece el **nombre de conexión**.

void setConnectionName (const char * *nombre* = 0);
Establece el **nombre de conexión**.

ImqQueueManager * connectionReference () Const.
Devuelve la **referencia de conexión**.

void setConnectionReference (ImqQueueManager & *gestor*);
Establece la **referencia de conexión**.

void setConnectionReference (ImqQueueManager * *gestor* = 0);
Establece la **referencia de conexión**.

void copyOut (MQAIR * *pAir*);
Copia datos de instancia en *pAir*, sustituyendo los datos de instancia existentes. Esto puede implicar la asignación de almacenamiento dependiente.

void clear (MQAIR * *pAir*);
Borra la estructura y libera el almacenamiento dependiente al que hace referencia *pAir*.

ImqAuthenticationRegistro * nextAuthenticationRecord () Const.
Devuelve el **siguiente registro de autenticación**.

const ImqString & password () Const.
Devuelve la **contraseña**.

void setPassword (const ImqString & *contraseña*);
Establece la **contraseña**.

void setPassword (const char * *contraseña* = 0);
Establece la **contraseña**.

ImqAuthenticationRegistro * previousAuthenticationRegistro () Const.
Devuelve el **registro de autenticación anterior**.

Tipo MQLONG () Const.
Devuelve el **tipo**.

void setType (const MQLONG *tipo*);
Establece el **tipo**.

const ImqString & userName () Const.
Devuelve el **nombre de usuario**.

void setUserName (const ImqString & *nombre*);
Establece el **nombre de usuario**.

void setUserName (const char * *nombre* = 0);
Establece el **nombre de usuario**.

Métodos de objeto (protegidos)

void setNextAuthenticationRecord (ImqAuthenticationRecord * *pAir* = 0);
Establece el **siguiente registro de autenticación**.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de registros de autenticación.

void setPreviousAuthenticationRecord (ImqAuthenticationRecord * *pAir* = 0);
Establece el **registro de autenticación anterior**.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de registros de autenticación.

Clase C++ ImqBinary

Esta clase encapsula una matriz de bytes binarios que se puede utilizar para los valores ImqMessage **señal de contabilidad**, **id de correlación**y **id de mensaje** . Permite una fácil asignación, copia y comparación.

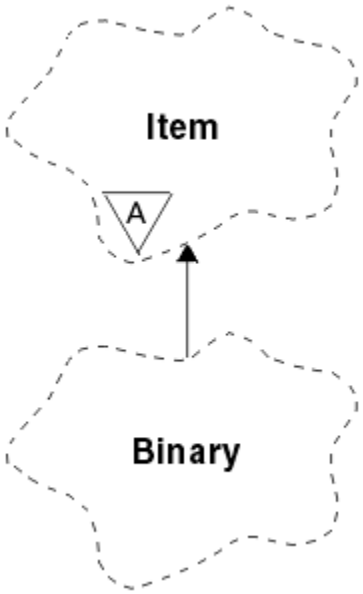


Figura 47. Clase ImqBinary

- "Atributos de objetos" en la página 1335
- "Constructores" en la página 1335
- "Métodos ImqItem sobrecargados" en la página 1336
- "Métodos de objeto (public)" en la página 1336
- "Métodos de objeto (protegidos)" en la página 1336
- "códigos de razón" en la página 1336

Atributos de objetos

data
Una matriz de bytes de datos binarios. El valor inicial es nulo.

Longitud de datos
Número de bytes. El valor inicial es cero.

puntero de datos
La dirección del primer byte de los **datos**. El valor inicial es cero.

Constructores

ImqBinary( );
El constructor predeterminado.

ImqBinary(const ImqBinary & ***binario*** **);**
El constructor de copia.

ImqBinary(const void * ***datos*****, const size_t** ***longitud*** **);**
Copia *longitud* bytes de *datos*.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut(ImqMessage &*msg*);
Copia los **datos** en el almacenamiento intermedio de mensajes, sustituyendo cualquier contenido existente. Establece el formato *msg* en MQFMT_NONE.

Consulte la descripción del método de clase ImqItem para obtener más detalles.

virtual ImqBoolean pasteIn(ImqMessage &*msg*);
Establece los **datos** transfiriendo los datos restantes del almacenamiento intermedio de mensajes, sustituyendo los **datos**existentes.

Para ser satisfactorio, el **formato** ImqMessage debe ser MQFMT_NONE.

Consulte la descripción del método de clase ImqItem para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqBinary &*binario*);
Copia bytes de *binario*.

ImqBoolean operator == (const ImqBinary &*binario*);
Compara este objeto con *binario*. Devuelve FALSE si no es igual y TRUE de lo contrario. Los objetos son iguales si tienen la misma **longitud de datos** y los bytes coinciden.

ImqBoolean copyOut(void **buffer*, const size_t *length*, const char *pad* = 0);
Copia hasta *longitud* bytes del **puntero de datos** al *almacenamiento intermedio*. Si la **longitud de datos** es insuficiente, el espacio restante en el *almacenamiento intermedio* se rellena con *pad* bytes. *buffer* puede ser cero si *length* también es cero. *longitud* no debe ser negativa. Devuelve TRUE si es satisfactorio.

size_t dataLength() const ;
Devuelve la **longitud de datos**.

ImqBoolean setDataLength(const size_t *longitud*);
Establece la **longitud de datos**. Si la **longitud de datos** se cambia como resultado de este método, los datos del objeto no se inicializan. Devuelve TRUE si es satisfactorio.

void * dataPointer() const ;
Devuelve el **puntero de datos**.

ImqBoolean isNull() const ;
Devuelve TRUE si la **longitud de datos** es cero, o si todos los **datos** bytes son cero. De lo contrario, devuelve FALSE.

ImqBoolean set(const void **buffer*, const size_t *longitud*);
Copia *longitud* bytes de *almacenamiento intermedio*. Devuelve TRUE si es satisfactorio.

Métodos de objeto (protegidos)

void clear();
Reduce la **longitud de datos** a cero.

códigos de razón

- MQRC_NO_BUFFER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_INCONSISTENT_FORMAT

Clase C++ ImqCache

Utilice esta clase para mantener o ordenar datos en la memoria.

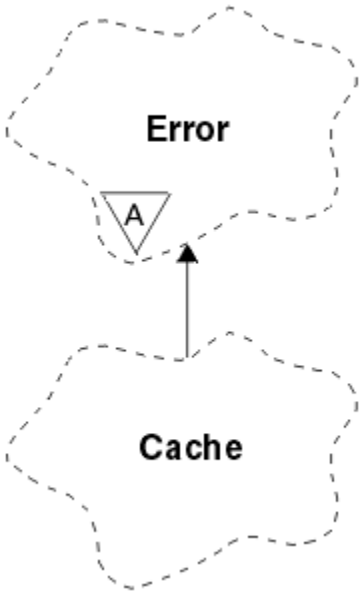


Figura 48. Clase ImqCache

Utilice esta clase para mantener o ordenar datos en la memoria. Puede nombrar un almacenamiento intermedio de memoria de tamaño fijo, o el sistema puede proporcionar una cantidad flexible de memoria automáticamente. Esta clase está relacionada con las llamadas MQI listadas en "Referencia cruzada de ImqCache" en la página 1317.

Atributos de objetos

almacenamiento intermedio automático
Indica si la memoria de almacenamiento intermedio la gestiona automáticamente el sistema (TRUE) o la proporciona el usuario (FALSE). Inicialmente se establece en TRUE.

Este atributo no se ha establecido directamente. Se establece indirectamente utilizando el almacenamiento intermedio **useEmpty** o el método **useFullBuffer** .

Si se proporciona almacenamiento de usuario, este atributo es FALSE, la memoria de almacenamiento intermedio no puede crecer y se pueden producir errores de desbordamiento de almacenamiento intermedio. La dirección y la longitud del almacenamiento intermedio permanecen constantes.

Si no se proporciona almacenamiento de usuario, este atributo es TRUE, y la memoria de almacenamiento intermedio puede crecer de forma incremental para acomodar una cantidad arbitraria de datos de mensaje. Sin embargo, cuando el almacenamiento intermedio crece, la dirección del almacenamiento intermedio puede cambiar, por lo que debe tener cuidado al utilizar el **puntero de almacenamiento intermedio** y el **puntero de datos**.

Longitud de almacenamiento intermedio
Número de bytes de memoria en el almacenamiento intermedio. El valor inicial es cero.

puntero de almacenamiento intermedio
La dirección de la memoria de almacenamiento intermedio. El valor inicial es nulo.

Longitud de datos
El número de bytes que suceden al **puntero de datos**. Debe ser igual o menor que la **longitud de mensaje**. El valor inicial es cero.

Desplazamiento de datos
Número de bytes que preceden al **puntero de datos**. Debe ser igual o menor que la **longitud de mensaje**. El valor inicial es cero.

puntero de datos
La dirección de la parte del almacenamiento intermedio que se va a escribir o leer en la siguiente. El valor inicial es nulo.

Longitud del mensaje
Número de bytes de datos significativos en el almacenamiento intermedio. El valor inicial es cero.

Constructores

ImqCache( );
El constructor predeterminado.

ImqCache(const ImqCache & ***memoria caché*** **);**
El constructor de copia.

Métodos de objeto (public)

void operator = (const ImqCache & ***caché*** **);**
Copia hasta **longitud de mensaje** bytes de datos del objeto *cache* en el objeto. Si **almacenamiento intermedio automático** es FALSE, la **longitud de almacenamiento intermedio** ya debe ser suficiente para acomodar los datos copiados.

ImqBoolean automaticBuffer() const ;
Devuelve el valor de **almacenamiento intermedio automático** .

size_t bufferLength() const ;
Devuelve la **longitud de almacenamiento intermedio**.

char * bufferPointer() const ;
Devuelve el **puntero de almacenamiento intermedio**.

void clearMessage();
Establece la **longitud de mensaje** y el **desplazamiento de datos** en cero.

size_t dataLength() const ;
Devuelve la **longitud de datos**.

size_t dataOffset() const ;
Devuelve el **desplazamiento de datos**.

ImqBoolean setDataOffset(const size_t ***desplazamiento*** **);**
Establece el **desplazamiento de datos**. La **longitud de mensaje** se incrementa si es necesario para asegurarse de que no es menor que el **desplazamiento de datos**. Este método devuelve TRUE si es satisfactorio.

char * dataPointer() const ;
Devuelve una copia del **puntero de datos**.

size_t messageLength() const ;
Devuelve la **longitud de mensaje**.

ImqBoolean setMessageLength(const size_t ***longitud*** **);**
Establece la **longitud de mensaje**. Aumenta la **longitud del almacenamiento intermedio** si es necesario para asegurarse de que la **longitud del mensaje** no es mayor que la **longitud del almacenamiento intermedio**. Reduce el **desplazamiento de datos** si es necesario para asegurarse de que no es mayor que la **longitud de mensaje**. Devuelve TRUE si es satisfactorio.

ImqBoolean moreBytes(const size_t ***bytes-necesario*** **);**
Asegura que *bytes-necesarios* más bytes están disponibles (para escritura) entre el **puntero de datos** y el final del almacenamiento intermedio. Devuelve TRUE si es satisfactorio.

Si **almacenamiento intermedio automático** es TRUE, se adquiere más memoria según sea necesario; de lo contrario, la **longitud del almacenamiento intermedio** ya debe ser adecuada.

ImqBoolean read(const size_t *longitud*, char * & *almacenamiento intermedio externo*);
Copia *longitud* bytes, del almacenamiento intermedio que empieza en la posición **puntero de datos** , en el *almacenamiento intermedio externo*. Una vez copiados los datos, el **desplazamiento de datos** se incrementa en *longitud*. Este método devuelve TRUE si es satisfactorio.

ImqBoolean resizeBuffer(const size_t *longitud*);
Varía la **longitud del almacenamiento intermedio**, siempre que el **almacenamiento intermedio automático** sea TRUE. Esto se consigue reasignando la memoria intermedia. Hasta **longitud de mensaje** bytes de datos del almacenamiento intermedio existente se copian en el nuevo. El número máximo copiado es de *longitud* bytes. Se cambia el **puntero de almacenamiento intermedio** . La **longitud de mensaje** y el **desplazamiento de datos** se conservan tan cerca como sea posible dentro de los confines del nuevo almacenamiento intermedio. Devuelve TRUE si es satisfactorio, y FALSE si **almacenamiento intermedio automático** es FALSE.

Nota: Este método puede fallar con MQRC_STORAGE_NOT_AVAILABLE si hay algún problema con los recursos del sistema.

ImqBoolean useEmptyBuffer(const char * *external-buffer*, const size_t *longitud*);
Identifica un almacenamiento intermedio de usuario vacío, estableciendo el **puntero de almacenamiento intermedio** para que apunte a *almacenamiento intermedio externo*, la **longitud de almacenamiento intermedio** a *longitud*y la **longitud de mensaje** a cero. Realiza un **clearMessage**. Si el almacenamiento intermedio está completamente preparado con datos, utilice el método **useFullBuffer** en su lugar. Si el almacenamiento intermedio está parcialmente preparado con datos, utilice el método **setMessageLength** para indicar la cantidad correcta. Este método devuelve TRUE si es satisfactorio.

Este método se puede utilizar para identificar una cantidad fija de memoria, tal como se ha descrito anteriormente (*almacenamiento intermedio externo* no es nulo y *longitud* no es cero), en cuyo caso el **almacenamiento intermedio automático** se establece en FALSE, o se puede utilizar para revertir a la memoria flexible gestionada por el sistema (*almacenamiento intermedio externo* es nulo y *longitud* es cero), en cuyo caso el **almacenamiento intermedio automático** se establece en TRUE.

ImqBoolean useFullBuffer(const char * *externalBuffer*, const size_t *longitud*);
En cuanto a **useEmptyBuffer**, excepto que la **longitud del mensaje** se establece en *length*. Devuelve TRUE si es satisfactorio.

ImqBoolean write(const size_t *longitud*, const char * *almacenamiento intermedio externo*);
Copia *longitud* bytes, del *almacenamiento intermedio externo*, en el almacenamiento intermedio empezando en la posición de **puntero de datos** . Una vez copiados los datos, el **desplazamiento de datos** aumenta en *longitud*, y la **longitud de mensaje** aumenta si es necesario para asegurarse de que no es menor que el nuevo valor de **desplazamiento de datos** . Este método devuelve TRUE si es satisfactorio.

Si **almacenamiento intermedio automático** es TRUE, se garantiza una cantidad adecuada de memoria; de lo contrario, el **desplazamiento de datos** final no debe superar la **longitud del almacenamiento intermedio**.

códigos de razón

- MQRC_BUFFER_NOT_AUTOMATIC
- MQRC_DATA_TRUNCADO
- MQRC_INSUFICIENT_BUFFER
- MQRC_INSUFICIENT_DATA
- MQRC_NULL_POINTER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_LONGITUD_CERO

Clase C++ ImqChannel

Esta clase encapsula una definición de canal (MQCD) para su uso durante la ejecución del método Manager: :connect, para conexiones de cliente personalizadas.

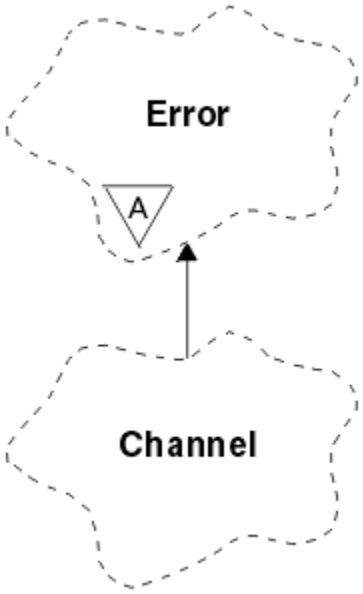


Figura 49. Clase ImqChannel

Consulte la descripción del método Manager: :connect y Programa de ejemplo HELLO WORLD (imqwrld.cpp) , para obtener más detalles. No todos los métodos listados son aplicables a todas las plataformas; consulte las descripciones de los mandatos DEFINE CHANNEL y ALTER CHANNEL en Referencia de MQSC para obtener más detalles. La clase ImqChannel no está soportada en z/OS.

Atributos de objetos

latido por lotes
El número de milisegundos entre comprobaciones de que un canal remoto está activo. El valor inicial es 0.

Nombre de canal
El nombre del canal. El valor inicial es nulo.

Nombre de conexión
El nombre de la conexión. Por ejemplo, la dirección IP de un sistema host. El valor inicial es nulo.

Compresión de cabecera
La lista de métodos de compresión de datos de cabecera que el canal admite. Todos los valores iniciales se establecen en MQCOMPRESS_NOT_AVAILABLE.

intervalo de latido
Número de segundos entre comprobaciones de que una conexión sigue funcionando. El valor inicial es 300.

Intervalo de mantenimiento activado
Número de segundos pasados a la pila de comunicaciones especificando la temporización de mantener activo para el canal. El valor inicial es MQKAI_AUTO.

dirección local
La dirección de comunicaciones locales para el canal.

longitud máxima del mensaje
Longitud máxima de mensaje soportada por el canal en una sola comunicación. El valor inicial es 4 194 304.

Compresión de mensaje
La lista de métodos de compresión de datos de mensaje que el canal admite. Todos los valores iniciales se establecen en MQCOMPRESS_NOT_AVAILABLE.

nombre modalidad
El nombre de la modalidad. El valor inicial es nulo.

password
Una contraseña proporcionada para la autenticación de conexión. El valor inicial es nulo.

recuento de salidas de recepción
Número de salidas de recepción. El valor inicial es cero. Este atributo es de sólo lectura.

nombres de salida de recepción
Los nombres de las salidas de recepción.

recibir datos de usuario
Datos asociados con salidas de recepción.

Nombre de salida de seguridad
El nombre de una salida de seguridad que se invocará en el lado del servidor de la conexión. El valor inicial es nulo.

datos de usuario de seguridad
Datos que deben pasarse a la salida de seguridad. El valor inicial es nulo.

recuento de salidas de envío
Número de salidas de envío. El valor inicial es cero. Este atributo es de sólo lectura.

nombres de salida de envío
Los nombres de las salidas de envío.

enviar datos de usuario
Datos asociados con salidas de envío.

CipherSpec de SSL
CipherSpec para su uso con SSL.

Tipo de autenticación de cliente SSL
Tipo de autenticación de cliente para utilizar con SSL.

Nombre de igual SSL
Nombre de igual para utilizar con SSL.

nombre del programa de transacciones
El nombre del programa de transacción. El valor inicial es nulo.

Tipo de transporte
El tipo de transporte de la conexión. El valor inicial es MQXPT_LU62.

ID de usuario
Identificador de usuario proporcionado para la autorización. El valor inicial es nulo.

Constructores

ImqChannel( ) ;
El constructor predeterminado.

ImqChannel(const ImqChannel & ***canal*** **);**
El constructor de copia.

Métodos de objeto (public)

void operator = (const ImqChannel &*canal*);
Copia datos de instancia de *channel*, sustituyendo los datos de instancia existentes.

MQLONG batchHeartBeat () Const.
Devuelve el **latido por lotes**.

ImqBoolean setBatchHeartBeat(const MQLONG *latido* = 0L);
Establece el **latido por lotes** . Este método devuelve TRUE si es satisfactorio.

ImqString channelName() Const.
Devuelve el **nombre de canal**.

ImqBoolean setChannelNombre (const char * *name* = 0);
Establece el **nombre de canal**. Este método devuelve TRUE si es satisfactorio.

ImqString connectionName() Const.
Devuelve el **nombre de conexión**.

ImqBoolean setConnectionNombre (const char * *name* = 0);
Establece el **nombre de conexión**. Este método devuelve TRUE si es satisfactorio.

size_t headerCompressionRecuento () Const.
Devuelve el recuento de técnicas de compresión de datos de cabecera soportadas.

ImqBoolean headerCompression(recuento de const size_t, MQLONG compress []) Const.
Devuelve copias de las técnicas de compresión de datos de cabecera soportadas en **compress**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setHeaderCompression (const size_t count, const MQLONG compress []);

Establece las técnicas de compresión de datos de cabecera soportadas en **compress**.

Establece el recuento de las técnicas de compresión de datos de cabecera soportadas en **recuento**.

Este método devuelve TRUE si es satisfactorio.

Intervalo MQLONG heartBeat() Const.
Devuelve el **intervalo de latido**.

ImqBoolean setHeartBeatInterval(const MQLONG *intervalo* = 300L);
Establece el **intervalo de latido**. Este método devuelve TRUE si es satisfactorio.

Intervalo MQLONG keepAlive() Const.
Devuelve el **intervalo de mantener activo**.

ImqBoolean setKeepAliveInterval(const MQLONG *intervalo* = MQKAI_AUTO);
Establece el **intervalo de mantener activo**. Este método devuelve TRUE si es satisfactorio.

ImqString localAddress() const;
Devuelve la **dirección local**.

ImqBoolean setLocalAddress (const char * *address* = 0);
Establece la **dirección local**. Este método devuelve TRUE si es satisfactorio.

MQLONG maximumMessageLongitud () Const.
Devuelve la **longitud máxima de mensaje**.

ImqBoolean setMaximumMessageLength(const MQLONG *length* = 4194304L);
Establece la **longitud máxima de mensaje**. Este método devuelve TRUE si es satisfactorio.

size_t messageCompressionRecuento () Const.
Devuelve el recuento de técnicas de compresión de datos de mensajes soportadas.

ImqBoolean messageCompression(const size_t count, MQLONG compress []) const;
Devuelve copias de las técnicas de compresión de datos de mensaje soportadas en **comprimir**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setMessageCompression (const size_t count, const MQLONG compress []);

Establece las técnicas de compresión de datos de mensaje soportadas para comprimir.

Establece el recuento de las técnicas de compresión de datos de mensajes soportadas.

Este método devuelve TRUE si es satisfactorio.

ImqString modeName() Const.
Devuelve el **nombre de modalidad**.

ImqBoolean setModeName (const char * *nombre* = 0);
Establece el **nombre de modalidad**. Este método devuelve TRUE si es satisfactorio.

ImqString contraseña () Const.
Devuelve la **contraseña**.

ImqBoolean setPassword(const char * *contraseña* = 0);
Establece la **contraseña**. Este método devuelve TRUE si es satisfactorio.

size_t receiveExitRecuento () Const.
Devuelve el **recuento de salidas de recepción**.

ImqString receiveExitNombre ();
Devuelve el primero de los **nombres de salida de recepción**, si los hay. Si el **recuento de salidas de recepción** es cero, devuelve una serie vacía.

ImqBoolean receiveExitNames (const size_t *recuento*, ImqString * *nombres* []);
Devuelve copias de los **nombres de salida de recepción** en *nombres*. Establece cualquier *nombres* que supere el **recuento de salidas de recepción** en series nulas. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveExitName(const char * *name* = 0);
Establece los **nombres de salida de recepción** en el único *nombre*. *nombre* puede estar en blanco o ser nulo. Establece el **recuento de salidas de recepción** en 1 o cero. Borra **recibir datos de usuario**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveExitNames(const size_t *recuento*, const char * *nombres* []);
Establece los **nombres de salida de recepción** en *nombres*. Los valores de *nombres* individuales no deben estar en blanco ni ser nulos. Establece el **recuento de salidas de recepción** en *recuento*. Borra **recibir datos de usuario**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveExitNames(const size_t *recuento*, const ImqString * *nombres* []);
Establece los **nombres de salida de recepción** en *nombres*. Los valores de *nombres* individuales no deben estar en blanco ni ser nulos. Establece el **recuento de salidas de recepción** en *recuento*. Borra **recibir datos de usuario**. Este método devuelve TRUE si es satisfactorio.

ImqString receiveUserData ();
Devuelve el primero de los elementos **recibir datos de usuario** , si los hay. Si el **recuento de salidas de recepción** es cero, devuelve una serie vacía.

ImqBoolean receiveUserData (const size_t *recuento*, ImqString * *datos* []);
Devuelve copias de los elementos **recibir datos de usuario** en *datos*. Establece cualquier *dato* que supere el **recuento de salidas de recepción** en series nulas. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveUserData(const char * *data* = 0);
Establece los **datos de usuario de recepción** en el elemento único *datos*. Si *datos* no es nulo, el **recuento de salidas de recepción** debe ser como mínimo 1. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveUserData(const size_t *recuento*, const char * *datos* []);
Establece **recibir datos de usuario** en *datos*. *recuento* no debe ser mayor que el **recuento de salidas de recepción**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setReceiveUserData(const size_t *recuento*, const ImqString * *datos* []);
Establece **recibir datos de usuario** en *datos*. *recuento* no debe ser mayor que el **recuento de salidas de recepción**. Este método devuelve TRUE si es satisfactorio.

ImqString securityExitNombre () Const.
Devuelve el **nombre de salida de seguridad**.

ImqBoolean setSecurityExitName(const char * *name* = 0);
Establece el **nombre de salida de seguridad**. Este método devuelve TRUE si es satisfactorio.

ImqString securityUserData () Const.
Devuelve los **datos de usuario de seguridad**.

ImqBoolean setSecurityUserData(const char * *data* = 0);
Establece los **datos de usuario de seguridad**. Este método devuelve TRUE si es satisfactorio.

size_t sendExitRecuento () Const.
Devuelve el **recuento de salidas de envío**.

ImqString sendExitNombre ();
Devuelve el primero de los **nombres de salida de envío**, si los hay. Devuelve una serie vacía si el **recuento de salidas de envío** es cero.

ImqBoolean sendExitNames (const size_t *recuento*, ImqString * *nombres* []);
Devuelve copias de los **nombres de salida de envío** en *nombres*. Establece cualquier *nombres* que supere el **recuento de salidas de envío** en series nulas. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendExitName(const char * *name* = 0);
Establece los **nombres de salida de envío** en el único *nombre*. *nombre* puede estar en blanco o ser nulo. Establece el **recuento de salidas de envío** en 1 o cero. Borra los **datos de usuario de envío**. Este método devuelve TRUE si es satisfactorio

ImqBoolean setSendExitNames(const size_t *recuento*, const char * *nombres* []);
Establece los **nombres de salida de envío** en *nombres*. Los valores de *nombres* individuales no deben estar en blanco ni ser nulos. Establece el **recuento de salidas de envío** en *recuento*. Borra los **datos de usuario de envío**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendExitNames(const size_t *recuento*, const ImqString * *nombres* []);
Establece los **nombres de salida de envío** en *nombres*. Los valores de *nombres* individuales no deben estar en blanco ni ser nulos. Establece el **recuento de salidas de envío** en *recuento*. Borra los **datos de usuario de envío**. Este método devuelve TRUE si es satisfactorio.

ImqString sendUserDatos ();
Devuelve el primero de los elementos **enviar datos de usuario** , si los hay. Devuelve una serie vacía si el **recuento de salidas de envío** es cero.

ImqBoolean sendUserData (const size_t *recuento*, ImqString * *datos* []);
Devuelve copias de los elementos **enviar datos de usuario** en *datos*. Establece cualquier *dato* que supere el **recuento de salidas de envío** en series nulas. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendUserData(const char * *data* = 0);
Establece el **envío de datos de usuario** en el elemento único *datos*. Si *datos* no es nulo, el **recuento de salidas de envío** debe ser como mínimo 1. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendUserData(const size_t *recuento*, const char * *datos* []);
Establece **enviar datos de usuario** en *datos*. *recuento* no debe ser mayor que el **recuento de salidas de envío**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setSendUserData(const size_t *recuento*, const ImqString * *datos* []);
Establece **enviar datos de usuario** en *datos*. *recuento* no debe ser mayor que el **recuento de salidas de envío**. Este método devuelve TRUE si es satisfactorio.

ImqString sslCipherSpecification () Const.
Devuelve la especificación de cifrado SSL.

ImqBoolean setSslCipherSpecification(const char * *nombre* = 0);
Establece la especificación de cifrado SSL. Este método devuelve TRUE si es satisfactorio.

Autenticación MQLONG sslClient() Const.
Devuelve el tipo de autenticación de cliente SSL.

ImqBoolean setSslClientAuthentication(const MQLONG *auth* = MQSCA_REQUIRED);
Establece el tipo de autenticación de cliente SSL. Este método devuelve TRUE si es satisfactorio.

ImqString sslPeerNombre () Const.
Devuelve el nombre de igual SSL.

ImqBoolean setSslPeerName(const char * *nombre* = 0);
Establece el nombre de igual SSL. Este método devuelve TRUE si es satisfactorio.

ImqString transactionProgramNombre () Const.
Devuelve el **nombre de programa de transacción**.

ImqBoolean setTransactionProgramName(const char * *name* = 0);
Establece el **nombre de programa de transacción**. Este método devuelve TRUE si es satisfactorio.

MQLONG transportType() Const.
Devuelve el **tipo de transporte**.

ImqBoolean setTransportType (const MQLONG *type* = MQXPT_LU62);
Establece el **tipo de transporte**. Este método devuelve TRUE si es satisfactorio.

ImqString userId() Const.
Devuelve el **ID de usuario**.

ImqBoolean setUserId (const char * *id* = 0);
Establece el **ID de usuario**. Este método devuelve TRUE si es satisfactorio.

códigos de razón

- MQRC_DATA_LENGTH_ERROR
- MQRC_ITEM_COUNT_ERROR
- MQRC_NULL_POINTER
- MQRC_SOURCE_BUFFER_ERROR

Clase C++ de cabecera ImqCICSBridge

Esta clase encapsula características específicas de la estructura de datos MQCIH.

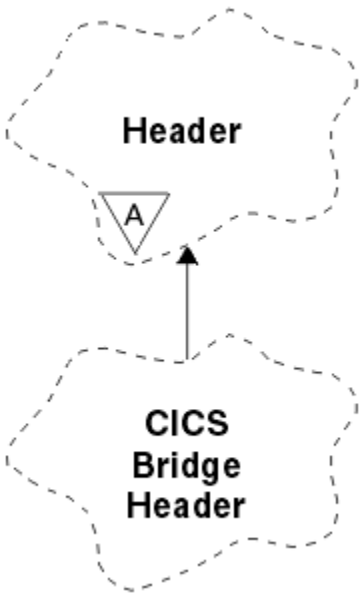


Figura 50. Clase de cabecera ImqCICSBridge

Los objetos de esta clase los utilizan las aplicaciones que envían mensajes al puente CICS a través de WebSphere MQ para z/OS.

- "Atributos de objetos" en la página 1346
- "Constructores" en la página 1348
- "Métodos ImqItem sobrecargados" en la página 1348
- "Métodos de objeto (public)" en la página 1348

- “Datos de objeto (protegidos)” en la página 1351
- “códigos de razón” en la página 1351
- “Códigos de retorno” en la página 1351

Atributos de objetos

Descriptor ADS
Enviar/recibir descriptor ADS. Se establece utilizando MQCADSD_NONE. El valor inicial es MQCADSD_NONE. Son posibles los siguientes valores adicionales:

- MQCADSD_NONE
- MQCADSD_SEND
- MQCADSD_RECV
- MQCADSD_MSGFORMAT

identificador de atención
Tecla AID. El campo debe tener la longitud MQ_ATTENTION_ID_LENGTH.

autenticador
Contraseña o passticket de RACF . El valor inicial contiene espacios en blanco, de longitud MQ_AUTHENTICATOR_LENGTH.

código de terminación anómala de puente
Código de terminación anómala de puente, de longitud MQ_ABEND_CODE_LENGTH. El valor inicial es de cuatro caracteres en blanco. El valor devuelto en este campo depende del código de retorno. Consulte Tabla 614 en la página 1351 para obtener más detalles.

código de cancelación de puente
Código de transacción de terminación anómala de puente. El campo está reservado, debe contener espacios en blanco y tener la longitud MQ_CANCEL_CODE_LENGTH.

código de terminación de puente
Código de terminación, que puede contener el código de terminación de WebSphere MQ o el valor EIBRESP de CICS . El campo tiene el valor inicial de MQCC_OK. El valor devuelto en este campo depende del código de retorno. Consulte Tabla 614 en la página 1351 para obtener más detalles.

desplazamiento de error de puente
Desplazamiento de error de puente. El valor inicial es cero. Este atributo es de sólo lectura.

código de razón de puente
Código de razón. Este campo puede contener la razón WebSphere MQ o el valor CICS EIBRESP2 . El campo tiene el valor inicial de MQRC_NONE. El valor devuelto en este campo depende del código de retorno. Consulte Tabla 614 en la página 1351 para obtener más detalles.

código de retorno de puente
Código de retorno del puente CICS . El valor inicial es MQCRC_OK.

tarea conversacional
Si la tarea puede ser conversacional. El valor inicial es MQCCT_NO. Son posibles los siguientes valores adicionales:

- MQCCT_YES
- MQCCT_NO

Posición del cursor
Posición del cursor. El valor inicial es cero.

tiempo de mantenimiento del recurso
Tiempo de release del recurso de puente CICS .

recurso como
Atributo emulado de terminal. El campo debe tener la longitud MQ_FACILITY_LIKE_LENGTH.

señal de recurso
Valor de señal BVT. El campo debe tener la longitud MQ_FACILITY_LENGTH. El valor inicial es MQCFAC_NONE.

Función
Función, que puede contener el nombre de llamada WebSphere MQ o la función EIBFN CICS . El campo tiene el valor inicial de MQCFUNC_NONE, con la longitud MQ_FUNCTION_LENGTH. El valor devuelto en este campo depende del código de retorno. Consulte Tabla 614 en la página 1351 para obtener más detalles.

Los siguientes valores adicionales son posibles cuando la función contiene un nombre de llamada WebSphere MQ :

- MQCFUNC_MQCONN
- MQCFUNC_MQGET
- MQCFUNC_MQINQ
- MQCFUNC_NONE
- MQCFUNC_MQOPEN
- MQCFUNC_PUT
- MQCFUNC_MQPUT1

obtener intervalo de espera
Intervalo de espera para una llamada MQGET emitida por la tarea de puente CICS . El valor inicial es MQCGWI_DEFAULT. El campo sólo se aplica cuando **uow control** tiene el valor MQCUOWC_FIRST. Son posibles los siguientes valores adicionales:

- MQCGWI_DEFAULT
- MQWI_UNLIMITED

Tipo de enlace
Tipo de enlace. El valor inicial es MQCLT_PROGRAM. Son posibles los siguientes valores adicionales:

- MQCLT_PROGRAM
- TRANSACCIÓN_MQC

identificador de transacción siguiente
ID de la siguiente transacción a adjuntar. El campo debe tener la longitud MQ_TRANSACTION_ID_LENGTH.

longitud de datos de salida
Longitud de datos de COMMAREA. El valor inicial es MQCODL_AS_INPUT.

formato de respuesta
Nombre de formato del mensaje de respuesta. El valor inicial es MQFMT_NONE con la longitud MQ_FORMAT_LENGTH.

código de inicio
Código de inicio de transacción. El campo debe tener la longitud MQ_START_CODE_LENGTH. El valor inicial es MQCSC_NONE. Son posibles los siguientes valores adicionales:

- MQCSC_START
- MQCSC_STARTDATA
- MQCSC_TERMINPUT
- MQCSC_NONE

estado de finalización de tarea
Estado de finalización de tarea. El valor inicial es MQCTES_NOSYNC. Son posibles los siguientes valores adicionales:

- MQCTES_COMMIT
- MQCTES_BACKOUT

- MQCTES_ENDTASK
- MQCTES_NOSYNC

Identificador de transacción
ID de la transacción a adjuntar. El valor inicial debe contener espacios en blanco y debe tener la longitud MQ_TRANSACTION_ID_LENGTH. El campo sólo se aplica cuando **uow control** tiene el valor MQCUOWC_FIRST o MQCUOWC_ONLY.

Control de UOW
Control de UOW. El valor inicial es MQCUOWC_ONLY. Son posibles los siguientes valores adicionales:

- MQCUOWC_FIRST
- MQCUOWC_MIDDLE
- MQCUOWC_LAST
- MQCUOWC_ONLY
- MQCUOWC_COMMIT
- MQCUOWC_BACKOUT
- MQCUOWC_CONTINUE

versión
El número de versión de MQCIH. El valor inicial es MQCIH_VERSION_2. El único otro valor soportado es MQCIH_VERSION_1.

Constructores

ImqCICSBridgeHeader ();
El constructor predeterminado.

ImqCICSBridgeHeader (const ImqCICSBridgeHeader & *cabecera*);
El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut(ImqMessage & *msg*);
Inserta una estructura de datos MQCIH en el almacenamiento intermedio de mensajes al principio, moviendo los datos de mensaje existentes más adelante y establece el formato de mensaje en MQFMT_CICS.

Consulte la descripción del método de clase padre para obtener más detalles.

virtual ImqBoolean pasteIn(ImqMessage & *msg*);
Lee una estructura de datos MQCIH del almacenamiento intermedio de mensajes. Para que la codificación del objeto *msg* sea correcta, debe ser MQENC_NATIVE. Recuperar mensajes con MQGMO_CONVERT a MQENC_NATIVE. Para ser satisfactorio, el formato ImqMessage debe ser MQFMT_CICS.

Consulte la descripción del método de clase padre para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqCICSBridgeHeader & *cabecera*);
Copia datos de instancia de la cabecera de , sustituyendo los datos de instancia existentes.

MQLONG ADSDescriptor () Const.
Devuelve una copia del **descriptor ADS**.

void setADSDescriptor(const MQLONG *descriptor* = MQCADSD_NONE);
Establece el **Descriptor ADS**.

ImqString attentionIdentifier() Const.
Devuelve una copia del **identificador de atención**, rellenado con espacios en blanco finales hasta la longitud MQ_ATTENTION_ID_LENGTH.

void setAttentionIdentifier (const char * *data* = 0);
Establece el **identificador de atención**, rellenado con espacios en blanco finales en la longitud MQ_ATTENTION_ID_LENGTH. Si no se proporciona ningún *dato* , restablece el **identificador de atención** en el valor inicial.

Autenticador ImqString () Const.
Devuelve una copia del **autenticador**, rellenada con espacios en blanco finales hasta la longitud MQ_AUTHENTICATOR_LENGTH.

void setAuthenticator(const char * *data* = 0);
Establece el **autenticador**, rellenado con espacios en blanco finales en la longitud MQ_AUTHENTICATOR_LENGTH. Si no se proporcionan *datos* , restablece el **autenticador** al valor inicial.

ImqString bridgeAbendCode () Const.
Devuelve una copia del **código de terminación anómala de puente**, rellenado con espacios en blanco finales hasta la longitud MQ_ABEND_CODE_LENGTH.

ImqString bridgeCancelCódigo () Const.
Devuelve una copia del **código de cancelación de puente**, rellenado con espacios en blanco finales hasta la longitud MQ_CANCEL_CODE_LENGTH.

void setBridgeCancelCode(const char * *data* = 0);
Establece el **código de cancelación de puente**, rellenado con espacios en blanco finales en la longitud MQ_CANCEL_CODE_LENGTH. Si no se proporcionan *datos* , restablece el **código de cancelación de puente** al valor inicial.

MQLONG bridgeCompletionCódigo () Const.
Devuelve una copia del **código de terminación de puente**.

MQLONG bridgeErrorOffset () Const.
Devuelve una copia del **desplazamiento de error de puente**.

MQLONG bridgeReasonCódigo () Const.
Devuelve una copia del **código de razón de puente**.

MQLONG bridgeReturnCódigo () Const.
Devuelve el **código de retorno de puente**.

MQLONG conversationalTask() Const.
Devuelve una copia de la **tarea conversacional**.

void setConversationalTask (const MQLONG *tarea* = MQCCT_NO);
Establece la **tarea conversacional**.

MQLONG cursorPosition() Const.
Devuelve una copia de la **posición del cursor**.

void setCursorPosition (const MQLONG *position* = 0);
Establece la **posición del cursor**.

MQLONG facilityKeepHora () Const.
Devuelve una copia del **tiempo de mantenimiento del recurso**.

void setFacilityKeepTime(const MQLONG *time* = 0);
Establece el **tiempo de mantenimiento del recurso**.

ImqString facilityLike() Const.
Devuelve una copia del recurso **como**, rellenada con espacios en blanco finales hasta la longitud MQ_FACILITY_LIKE_LENGTH.

void setFacilityLike (const char * *nombre* = 0);
Establece el recurso **como**, rellenado con espacios en blanco finales en la longitud MQ_FACILITY_LIKE_LENGTH. Si no se proporciona ningún *nombre* , restablece el recurso **como** el valor inicial.

ImqBinary facilityToken() Const.
Devuelve una copia de la **señal de recurso**.

ImqBoolean setFacilityToken (const ImqBinary & *señal*);
Establece la **señal de recurso**. La **longitud de datos** de *señal* debe ser cero o MQ_FACILITY_LENGTH. Devuelve TRUE si es satisfactorio.

void setFacilityToken (const MQBYTE8 *señal* = 0);
Establece la **señal de recurso**. *token* puede ser cero, que es lo mismo que especificar MQCFAC_NONE. Si *token* es distinto de cero, debe direccionar MQ_FACILITY_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como, por ejemplo, MQCFAC_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma. Por ejemplo, (MQBYTE *) MQCFAC_NONE.

Función ImqString () Const.
Devuelve una copia de la **función**, rellenada con espacios en blanco finales hasta la longitud MQ_FUNCTION_LENGTH.

MQLONG getWaitIntervalo () Const.
Devuelve una copia del **intervalo de espera de obtención**.

void setGetWaitInterval(const MQLONG *intervalo* = MQCGWI_DEFA
Establece el **intervalo de espera de obtención**.

MQLONG linkType() Const.
Devuelve una copia del **tipo de enlace**.

void setLinkType (const MQLONG *type* = MQCLT_PROGRAM);
Establece el **tipo de enlace**.

ImqString nextTransactionIdentificador () Const.
Devuelve una copia de los datos del **siguiente identificador de transacción** , rellenados con blancos de cola hasta la longitud MQ_TRANSACTION_ID_LENGTH.

MQLONG outputDataLongitud () Const.
Devuelve una copia de la **longitud de datos de salida**.

void setOutputDataLength(const MQLONG *length* = MQCODL_AS_INPUT);
Establece la **longitud de datos de salida**.

ImqString replyToFormato () Const.
Devuelve una copia del nombre de **formato de respuesta** , rellenado con blancos de cola hasta la longitud MQ_FORMAT_LENGTH.

void setReplyToFormat(const char * *nombre* = 0);
Establece el **formato de respuesta**, rellenado con espacios en blanco finales en la longitud MQ_FORMAT_LENGTH. Si no se proporciona ningún *nombre* , restablece el **formato de respuesta** al valor inicial.

ImqString startCode() Const.
Devuelve una copia del **código de inicio**, rellenado con espacios en blanco finales hasta la longitud MQ_START_CODE_LENGTH.

void setStartCode (const char * *data* = 0);
Establece los datos del **código de inicio** , rellenados con espacios en blanco finales en la longitud MQ_START_CODE_LENGTH. Si no se proporcionan *datos* , restablece el **código de inicio** en el valor inicial.

MQLONG taskEndEstado () Const.
Devuelve una copia del **estado de finalización de tarea**.

ImqString transactionIdentifier() Const.
Devuelve una copia de los datos del **identificador de transacción** , rellenados con espacios en blanco finales hasta la longitud MQ_TRANSACTION_ID_LENGTH.

void setTransactionIdentifier (const char * *data* = 0);
Establece el **identificador de transacción**, rellenado con espacios en blanco finales en la longitud MQ_TRANSACTION_ID_LENGTH. Si no se proporcionan *datos* , restablece el **identificador de transacción** en el valor inicial.

MQLONG UOWControl () Const.
Devuelve una copia del **control de UOW**.

void setUOWControl(const MQLONG *control* = MQCUOWC_ONLY);
Establece el **control de UOW**.

MQLONG versión () Const.
Devuelve el número de **versión** .

ImqBoolean setVersion(const MQLONG *versión* = MQCIH_VERSION_2);
Establece el número de **versión** . Devuelve TRUE si es satisfactorio.

Datos de objeto (protegidos)

MQLONG *olVersion*
El número máximo de versión de MQCIH que se puede acomodar en el almacenamiento asignado para *opcih*.

PMQCIH *opcih*
La dirección de una estructura de datos MQCIH. La cantidad de almacenamiento asignada se indica mediante *olVersion*.

códigos de razón

- MQRC_BINARY_DATA_LENGTH_ERROR
- VERSIÓN_ERROR_MQRC

Códigos de retorno

Tabla 614. Códigos de retorno de clase de cabecera ImqCICSBridge

Código de retorno	Función	CompCode	Razón	Código de terminación anómala
MQCRC_OK				
ERROR DE MQCRC_BRIDGE_			MQFB_CICS	
MQCRC_MQ_API_ERROR	Nombre de llamada de WebSphere MQ	WebSphere MQ CompCode	WebSphere MQ Razón	
MQCRC_BRIDGE_TIMEOUT	Nombre de llamada de WebSphere MQ	WebSphere MQ CompCode	WebSphere MQ Razón	
MQCRC_CICS_EXEC_ERROR	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_SECURITY_ERROR	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_PROGRAM_NOT_AVAILABLE	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_TRANSID_NOT_AVAILABLE	CICS EIBFN	CICS EIBRESP	CICS EIBRESP2	
MQCRC_BRIDGE_ABEND				ABCODE de CICS
MQCRC_APPLICATION_ABEND				ABCODE de CICS

Clase C++ ImqDeadLetterHeader

Esta clase encapsula características de la estructura de datos MQDLH.

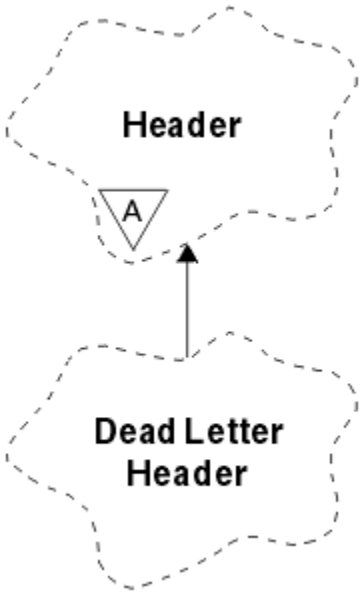


Figura 51. clase ImqDeadLetterHeader

Los objetos de esta clase suelen ser utilizados por una aplicación que encuentra un mensaje que no se puede procesar. Un nuevo mensaje que comprende una cabecera de mensaje no entregado y el contenido del mensaje se coloca en la cola de mensajes no entregados y el mensaje se descarta.

- "Atributos de objetos" en la página 1352
- "Constructores" en la página 1353
- "Métodos ImqItem sobrecargados" en la página 1353
- "Métodos de objeto (public)" en la página 1353
- "Datos de objeto (protegidos)" en la página 1354
- "códigos de razón" en la página 1354

Atributos de objetos

código de razón de mensaje no enviado
La razón por la que el mensaje ha llegado a la cola de mensajes no entregados. El valor inicial es MQRC_NONE.

Nombre del gestor de colas de destino
El nombre del gestor de colas de destino original. El nombre es una serie de longitud MQ_Q_MGR_NAME_LENGTH. Su valor inicial es nulo.

Nombre de la cola de destino
El nombre de la cola de destino original. El nombre es una serie de longitud MQ_Q_NAME_LENGTH. Su valor inicial es nulo.

Nombre de aplicación de transferencia
Nombre de la aplicación que transfiere el mensaje a la cola de mensajes no entregados. El nombre es una serie de longitud MQ_PUT_APPL_NAME_LENGTH. Su valor inicial es nulo.

Tipo de aplicación de transferencia
El tipo de aplicación que coloca el mensaje en la cola de mensajes no entregados. El valor inicial es cero.

Fecha de transferencia
La fecha en la que el mensaje se transfirió a la cola de mensajes no entregados. La fecha es una serie de longitud MQ_PUT_DATE_LENGTH. Su valor inicial es una serie nula.

Hora de transferencia
Hora a la que se transfirió el mensaje a la cola de mensajes no entregados. La hora es una serie de longitud MQ_PUT_TIME_LENGTH. Su valor inicial es una serie nula.

Constructores

ImqDeadLetterHeader( );
El constructor predeterminado.

ImqDeadLetterHeader(const ImqDeadLetterHeader & *cabecera*);
El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut(ImqMessage & *msg*);
Inserta una estructura de datos MQDLH en el almacenamiento intermedio de mensajes al principio, moviendo los datos de mensaje existentes más adelante. Establece el formato *msg* en MQFMT_DEAD_LETTER_HEADER.

Consulte la descripción del método de clase ImqHeader en la página "Clase C++ ImqHeader" en la página 1360 para obtener más detalles.

virtual ImqBoolean pasteIn(ImqMessage & *msg*);
Lee una estructura de datos MQDLH del almacenamiento intermedio de mensajes.

Para ser satisfactorio, el **formato** ImqMessage debe ser MQFMT_DEAD_LETTER_HEADER.

Consulte la descripción del método de clase ImqHeader en la página "Clase C++ ImqHeader" en la página 1360 para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqDeadLetterHeader & *cabecera*);
Copia los datos de instancia de *cabecera*, sustituyendo los datos de instancia existentes.

MQLONG deadLetterReasonCode() const ;
Devuelve el **código de razón de mensaje no enviado**.

void setDeadLetterReasonCode(const MQLONG *razón*);
Establece el **código de razón de mensaje no enviado**.

ImqString destinationQueueManagerName() const ;
Devuelve el **nombre de gestor de colas de destino**, despojado de los espacios en blanco finales.

void setDestinationQueueManagerName(const char * *nombre*);
Establece el **nombre de gestor de colas de destino**. Trunca los datos más largos que MQ_Q_MGR_NAME_LENGTH (48 caracteres).

ImqString destinationQueueNombre() const ;
Devuelve una copia del **nombre de cola de destino**, sin espacios en blanco finales.

void setDestinationQueueName(const char * *nombre*);
Establece el **nombre de cola de destino**. Trunca los datos más largos que MQ_Q_NAME_LENGTH (48 caracteres).

ImqString putApplicationNombre() const ;
Devuelve una copia del **nombre de aplicación de colocación**, despojada de los espacios en blanco finales.

void setPutApplicationName(const char * *name* = 0);
Establece el **nombre de aplicación de colocación**. Trunca los datos más largos que MQ_PUT_APPL_NAME_LENGTH (28 caracteres).

MQLONG putApplicationTipo() const ;
Devuelve el **tipo de aplicación put**.

void setPutApplicationType(const MQLONG *type* = MQAT_NO_CONTEXT);
Establece el **tipo de aplicación put**.

ImqString putDate() const ;
Devuelve una copia de la **fecha de colocación**, despojada de los espacios en blanco finales.

void setPutDate(const char * *date* = 0);
Establece la **fecha de colocación**. Trunca los datos más largos que MQ_PUT_DATE_LENGTH (8 caracteres).

ImqString putTime() const ;
Devuelve una copia de la **hora de colocación**, eliminada de los espacios en blanco finales.

void setPutTime(const char * *time* = 0);
Establece la **hora de colocación**. Trunca los datos más largos que MQ_PUT_TIME_LENGTH (8 caracteres).

Datos de objeto (protegidos)

MQDLH *omqdlh*
Estructura de datos MQDLH.

códigos de razón

- MQRC_INCONSISTENT_FORMAT
- MQRC_STRUC_ID_ERROR
- MQRC_ENCODING_ERROR

ImqDistributionListar clase C++

Esta clase encapsula una lista de distribución dinámica que hace referencia a una o más colas con el fin de enviar un mensaje o mensajes a varios destinos.

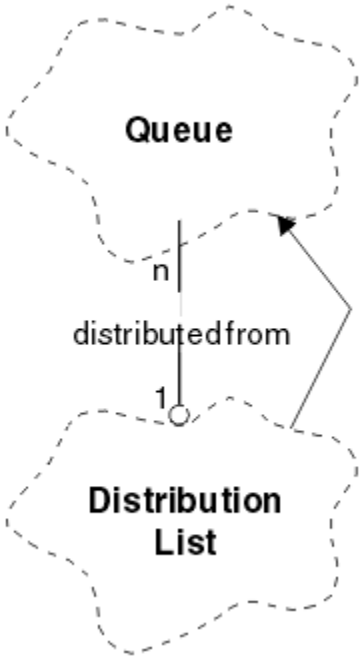


Figura 52. Clase de lista ImqDistribution

- "Atributos de objetos" en la página 1355
- "Constructores" en la página 1355
- "Métodos de objeto (public)" en la página 1355

- “Métodos de objeto (protegidos)” en la página 1355

Atributos de objetos

primera cola distribuida
El primero de uno o más objetos de clase, en ningún orden concreto, en el que la **referencia de lista de distribución** se dirige a este objeto.

Inicialmente no hay tales objetos. Para abrir una lista de ImqDistributioncorrectamente, debe haber al menos un objeto de este tipo.

Nota: Cuando se abre un objeto de lista ImqDistribution, los objetos abiertos que hacen referencia a él se cierran automáticamente.

Constructores

ImqDistributionList ();
El constructor predeterminado.

ImqDistributionList (const ImqDistributionList & *lista*);
El constructor de copia.

Métodos de objeto (public)

void operator = (const ImqDistributionList & *lista*);
Se anula la referencia de todos los objetos que hacen referencia a **este** objeto antes de copiarlos. Ningún objeto hará referencia a **este** objeto después de la invocación de este método.

*** firstDistributedQueue() const ;**
Devuelve la **primera cola distribuida**.

Métodos de objeto (protegidos)

void setFirstDistributedQueue(* *queue* = 0);
Establece la **primera cola distribuida**.

Clase C++ ImqError

Esta clase abstracta proporciona información sobre los errores asociados a un objeto.

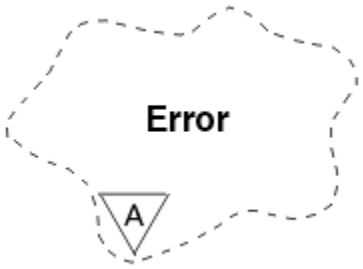


Figura 53. Clase ImqError

- “Atributos de objetos” en la página 1356
- “Constructores” en la página 1356
- “Métodos de objeto (public)” en la página 1356
- “Métodos de objeto (protegidos)” en la página 1356
- “códigos de razón” en la página 1356

Atributos de objetos

código de terminación
El código de terminación más reciente. El valor inicial es cero. Son posibles los siguientes valores adicionales:

- MQCC_OK
- MQCC_WARNING
- MQCC_FAILED

código de razón
El código de razón más reciente. El valor inicial es cero.

Constructores

ImqError( );
El constructor predeterminado.

ImqError(const ImqError & ***error*** **);**
El constructor de copia.

Métodos de objeto (public)

void operator = (const ImqError & ***error*** **);**
Copia datos de instancia de *error*, sustituyendo los datos de instancia existentes.

void clearErrorCódigos();
Establece el **código de terminación** y el **código de razón** en cero.

MQLONG completionCode() const ;
Devuelve el **código de terminación**.

MQLONG reasonCode() const ;
Devuelve el **código de razón**.

Métodos de objeto (protegidos)

ImqBoolean checkReadPointer(const void * ***pointer*****, const size_t** ***longitud*** **);**
Verifica que la combinación de puntero y longitud es válida para el acceso de sólo lectura y devuelve TRUE si es satisfactoria.

ImqBoolean checkWritePuntero(const void * ***pointer*****, const size_t** ***longitud*** **);**
Verifica que la combinación de puntero y longitud es válida para el acceso de lectura-escritura y devuelve TRUE si es satisfactoria.

void setCompletionCode(const MQLONG ***code*** **= 0);**
Establece el **código de terminación**.

void setReasonCode(const MQLONG ***code*** **= 0);**
Establece el **código de razón**.

códigos de razón

- MQRC_BUFFER_ERROR

Clase C++ ImqGetMessageOptions

Esta clase encapsula la estructura de datos MQGMO

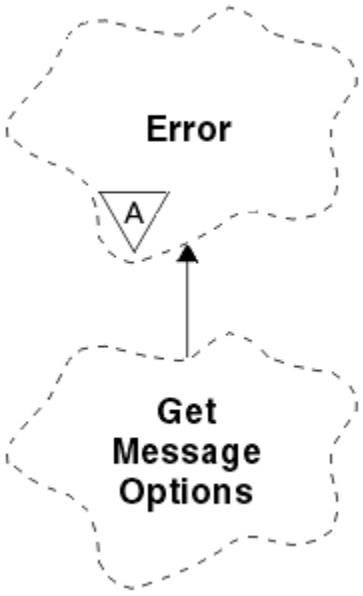


Figura 54. Clase ImqGetMessageOptions

Atributos de objetos

Estado de grupo
Estado de un mensaje para un grupo de mensajes. El valor inicial es MQGS_NOT_IN_GROUP. Son posibles los siguientes valores adicionales:

- MQGS_MSG_IN_GROUP
- MQGS_LAST_MSG_IN_GROUP

opciones de coincidencia
Opciones para seleccionar mensajes entrantes. El valor inicial es MQMO_MATCH_MSG_ID | MQMO_MATCH_CORREL_ID. Son posibles los siguientes valores adicionales:

- MQMO_ID_grupo
- MQMO_MATCH_MSG_SEQ_NUMBER
- MQMO_MATCH_OFFSET
- MQMO_MSG_TOKEN
- MQMO_NONE

token de mensaje
Señal de mensaje. Un valor binario (MQBYTE16) de longitud MQ_MSG_TOKEN_LENGTH. El valor inicial es MQMTOK_NONE.

opciones
Opciones aplicables a un mensaje. El valor inicial es MQGMO_NO_WAIT. Son posibles los siguientes valores adicionales:

- MQGMO_WAIT
- MQGMO_SYNCPOINT

- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_NO_SYNCPOINT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_LOCK
- MQGMO_UNLOCK
- MQGMO_ACCEPT_TRUNCATED_MSG
- MQGMO_SET_SIGNAL
- MQGMO_FAIL_IF_QUIESCING
- MQGMO_CONVERT
- MQGMO_LOGICAL_ORDER
- MQGMO_COMPLETE_MSG
- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_NONE

Nombre de cola resuelto
Nombre de cola resuelto. Este atributo es de sólo lectura. Los nombres nunca tienen más de 48 caracteres y pueden rellenarse hasta esa longitud con nulos. El valor inicial es una serie nula.

longitud devuelta
Longitud devuelta. El valor inicial es MQRL_UNDEFINED. Este atributo es de sólo lectura.

Segmentación
La capacidad de segmentar un mensaje. El valor inicial es MQSEG_INHIBIDO. El valor adicional, MQSEG_ALLOWED, es posible.

estado de segmento
El estado de segmentación de un mensaje. El valor inicial es MQSS_NOT_A_SEGMENT. Son posibles los siguientes valores adicionales:

- SEGMENTO_MQSS
- MQSS_LAST_SEGMENT

participación de punto de sincronismo
TRUE cuando los mensajes se recuperan bajo control de punto de sincronismo.

Intervalo de espera
El tiempo que el método **get** de la clase se detiene mientras espera a que llegue un mensaje adecuado, si todavía no hay uno disponible. El valor inicial es cero, lo que afecta a una espera indefinida. El valor adicional, MQWI_UNLIMITED, es posible. Este atributo se ignora a menos que las **opciones** incluyan MQGMO_WAIT.

Constructores

ImqGetMessageOptions( );
El constructor predeterminado.

ImqGetMessageOptions(const ImqGetMessageOptions & ***gmo*** **);**
El constructor de copia.

Métodos de objeto (public)

void operator = (const ImqGetMessageOptions &*gmo*);
Copia datos de instancia de *gmo*, sustituyendo los datos de instancia existentes.

MQCHAR groupStatus() const ;
Devuelve el **estado de grupo**.

void setGroupStatus(const MQCHAR *estado*);
Establece el **estado de grupo**.

MQLONG matchOptions() const ;
Devuelve las **opciones de coincidencia**.

void setMatchOptions(const MQLONG *opciones*);
Establece las **opciones de coincidencia**.

ImqBinary messageToken() Const.
Devuelve la **señal de mensaje**.

ImqBoolean setMessageToken (const ImqBinary &*señal*);
Establece la **señal de mensaje**. La **longitud de datos** de *señal* debe ser cero o MQ_MSG_TOKEN_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setMessageToken (const MQBYTE16 *señal* = 0);
Establece la **señal de mensaje**. *token* puede ser cero, que es lo mismo que especificar MQMTOK_NONE. Si *token* es distinto de cero, debe direccionar MQ_MSG_TOKEN_LENGTH bytes de datos binarios.

Cuando se utilizan valores predefinidos, como MQMTOK_NONE, es posible que no sea necesario realizar una conversión para garantizar una coincidencia de firma, por ejemplo (MQBYTE *) MQMTOK_NONE.

MQLONG opciones() const ;
Devuelve las **opciones**.

void setOptions(const MQLONG *opciones*);
Establece las **opciones**, incluido el valor de **participación de punto de sincronismo** .

ImqString resolvedQueueNombre() const ;
Devuelve una copia del **nombre de cola resuelto**.

MQLONG returnedLength() Const.
Devuelve la **longitud devuelta**.

MQCHAR segmentación() const ;
Devuelve la **segmentación**.

void setSegmentation(const MQCHAR *valor*);
Establece la **segmentación**.

MQCHAR segmentStatus() const ;
Devuelve el **estado de segmento**.

void setSegmentStatus(const MQCHAR *estado*);
Establece el **estado de segmento**.

ImqBoolean syncPointParticipación() const ;
Devuelve el valor de **participación de punto de sincronismo** , que es TRUE si las **opciones** incluyen MQGMO_SYNCPOINT o MQGMO_SYNCPOINT_IF_PERSISTENT.

void setSyncPointParticipation(const ImqBoolean *sincronización*);
Establece el valor de **participación de punto de sincronismo** . Si *sync* es TRUE, altera las **opciones** para incluir MQGMO_SYNCPOINT y excluir MQGMO_NO_SYNCPOINT y MQGMO_SYNCPOINT_IF_PERSISTENT. Si *sync* es FALSE, modifica las **opciones** para incluir MQGMO_NO_SYNCPOINT y excluir MQGMO_SYNCPOINT y MQGMO_SYNCPOINT_IF_PERSISTENT.

MQLONG waitInterval() const ;
Devuelve el **intervalo de espera**.

void setWaitInterval(const MQLONG *intervalo*);
Establece el **intervalo de espera**.

Métodos de objeto (protegidos)

static void setVersionSupported(const MQLONG);
Establece la versión de **MQGMO** . El valor predeterminado es **MQGMO_VERSION_3**.

Datos de objeto (protegidos)

MQGMO *omqgmo*
Una estructura de datos MQGMO Versión 2. Acceda a los campos MQGMO soportados sólo para MQGMO_VERSION_2 .

PMQGMO *opgmo*
La dirección de una estructura de datos MQGMO. El número de versión para esta dirección se indica en *olVersion*. Inspeccione el número de versión antes de acceder a los campos MQGMO, para asegurarse de que están presentes.

MQLONG *olVersion*
El número de versión de la estructura de datos MQGMO direccionado por *opgmo*.

códigos de razón

- MQRC_BINARY_DATA_LENGTH_ERROR

Clase C++ ImqHeader

Esta clase abstracta encapsula características comunes de la estructura de datos MQDLH.

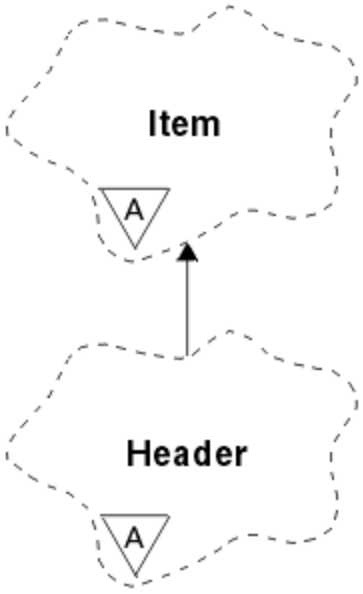


Figura 55. Clase ImqHeader

- "Atributos de objetos" en la página 1360
- "Constructores" en la página 1361
- "Métodos de objeto (public)" en la página 1361

Atributos de objetos

juego de caracteres
Identificador del juego de caracteres codificado original. Inicialmente MQCCSI_Q_MGR.

Encoding
La codificación original. Inicialmente MQENC_NATIVE.

Format
El formato original. Inicialmente MQFMT_NONE.

distintivos de cabecera
Los valores iniciales son:

- Cero para objetos de la clase ImqDeadLetterHeader
- MQIIH_NONE para objetos de la clase de cabecera ImqIMSBridge
- MQRMHF_LAST para objetos de la clase de cabecera ImqReference
- MQCIH_NONE para objetos de la clase de cabecera ImqCICSBridge
- MQWIH_NONE para objetos de la clase de cabecera ImqWork

Constructores

ImqHeader( );
El constructor predeterminado.

ImqHeader(const ImqHeader & ***cabecera*** **);**
El constructor de copia.

Métodos de objeto (public)

void operator = (const ImqHeader & ***cabecera*** **);**
Copia los datos de instancia de la cabecera de , sustituyendo los datos de instancia existentes.

virtual MQLONG characterSet() const ;
Devuelve el **juego de caracteres**.

virtual void setCharacterSet(const MQLONG ***ccsid*** **= MQCCSI_Q_MGR);**
Establece el **juego de caracteres**.

Codificación() MQLONG virtual const ;
Devuelve la **codificación**.

virtual void setEncoding(const MQLONG ***encoding*** **= MQENC_NATIVE);**
Establece la **codificación**.

virtual ImqString formato() const ;
Devuelve una copia del **formato**, incluidos los espacios en blanco finales.

virtual void setFormat(const char * ***name*** **= 0);**
Establece el **formato**, rellenado con 8 caracteres con blancos de cola.

virtual MQLONG headerFlags() const ;
Devuelve los **distintivos de cabecera**.

virtual void setHeaderFlags(const MQLONG ***flags*** **= 0);**
Establece los **distintivos de cabecera**.

Clase C++ de cabecera ImqIMSBridge

Esta clase encapsula características de la estructura de datos MQIIH.

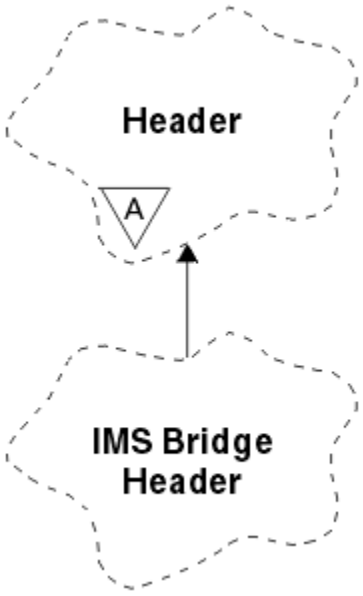


Figura 56. Clase de cabecera ImqIMSBridge

Los objetos de esta clase los utilizan las aplicaciones que envían mensajes al puente IMS a través de WebSphere MQ para z/OS.

Nota: El **juego de caracteres** ImqHeader y la **codificación** deben tener valores predeterminados y no deben establecerse en ningún otro valor.

- "Atributos de objetos" en la página 1362
- "Constructores" en la página 1363
- "Métodos ImqItem sobrecargados" en la página 1363
- "Métodos de objeto (public)" en la página 1363
- "Datos de objeto (protegidos)" en la página 1364
- "códigos de razón" en la página 1364

Atributos de objetos

autenticador
Contraseña o passticket de RACF, de longitud MQ_AUTHENTICATOR_LENGTH. El valor inicial es MQIAUT_NONE.

Modalidad de confirmación
Modalidad de confirmación. Consulte la publicación *OTMA User's Guide* para obtener más información sobre las modalidades de confirmación de IMS . El valor inicial es MQICM_COMMIT_THEN_SEND. El valor adicional, MQICM_SEND_THEN_COMMIT, es posible.

Alteración temporal del terminal lógico
Alteración temporal de terminal lógico, de longitud MQ_LTERM_OVERRIDE_LENGTH. El valor inicial es una serie nula.

Nombre de la correlación de servicios del formato del mensaje
Nombre de correlación MFS, de longitud MQ_MFS_MAP_NAME_LENGTH. El valor inicial es una serie nula.

formato de respuesta
Formato de cualquier respuesta, de longitud MQ_FORMAT_LENGTH. El valor inicial es MQFMT_NONE.

Ámbito de seguridad
Ámbito del proceso de seguridad de IMS . El valor inicial es MQISS_CHECK. El valor adicional, MQISS_FULL, es posible.

ID de instancia de transacción
Identidad de instancia de transacción, un valor binario (MQBYTE16) de longitud MQ_TRAN_INSTANCE_ID_LENGTH. El valor inicial es MQITII_NONE.

ESTADO DE TRANSACCIÓN
Estado de la conversación de IMS . El valor inicial es MQITS_NOT_IN_CONVERSATION. El valor adicional, MQITS_IN_CONVERSATION, es posible.

Constructores

ImqIMSBridgeCabecera ();
El constructor predeterminado.

Cabecera ImqIMSBridge(const ImqIMSBridgeHeader & *header*);
El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut(ImqMessage & *msg*);
Inserta una estructura de datos MQIIH en el almacenamiento intermedio de mensajes al principio, moviendo los datos de mensajes existentes más adelante. Establece el formato *msg* en MQFMT_IMS.

Consulte la descripción del método de clase padre para obtener más detalles.

virtual ImqBoolean pasteIn(ImqMessage & *msg*);
Lee una estructura de datos MQIIH del almacenamiento intermedio de mensajes.

Para que la operación sea satisfactoria, la **codificación** del objeto *msg* debe ser MQENC_NATIVE. Recuperar mensajes con MQGMO_CONVERT a MQENC_NATIVE.

Para ser satisfactorio, el **formato** ImqMessage debe ser MQFMT_IMS.

Consulte la descripción del método de clase padre para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqIMSBridgeHeader & *cabecera*);
Copia los datos de instancia de la cabecera de , sustituyendo los datos de instancia existentes.

ImqString autenticador() const ;
Devuelve una copia del **autenticador**, rellenada con espacios en blanco finales hasta la longitud MQ_AUTHENTICATOR_LENGTH.

void setAuthenticator(const char * *nombre*);
Establece el **autenticador**.

MQCHAR commitMode() const ;
Devuelve la **modalidad de confirmación**.

void setCommitMode(const MQCHAR *modalidad*);
Establece la **modalidad de confirmación**.

ImqString logicalTerminalOverride() const ;
Devuelve una copia de la **alteración temporal de terminal lógico**.

void setLogicalTerminalOverride(const char * *alterar temporalmente*);
Establece la **alteración temporal de terminal lógico**.

ImqString messageFormatServicesMapNombre() const ;
Devuelve una copia del **nombre de correlación de servicios de formato de mensaje**.

void setMessageFormatServicesMapName(const char * *nombre*);
Establece el **nombre de correlación de servicios de formato de mensaje**.

ImqString replyToFormato() const ;
Devuelve una copia del **formato de respuesta**, rellenado con espacios en blanco finales hasta la longitud MQ_FORMAT_LENGTH.

void setReplyToFormat(const char * *format*);
Establece el **formato de respuesta**, rellenado con espacios en blanco finales en la longitud MQ_FORMAT_LENGTH.

MQCHAR securityScope() const ;
Devuelve el **ámbito de seguridad**.

void setSecurityScope(const MQCHAR *ámbito*);
Establece el **ámbito de seguridad**.

ImqBinary transactionInstanceId() const ;
Devuelve una copia del **ID de instancia de transacción**.

ImqBoolean setTransactionInstanceId(const ImqBinary & *id*);
Establece el **ID de instancia de transacción**. La **longitud de datos** de *señal* debe ser cero o MQ_TRAN_INSTANCE_ID_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setTransactionInstanceId(const MQBYTE16 *id* = 0);
Establece el **ID de instancia de transacción**. *id* puede ser cero, que es lo mismo que especificar MQITII_NONE. Si *id* es distinto de cero, debe direccionar MQ_TRAN_INSTANCE_ID_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como MQITII_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma, por ejemplo (MQBYTE *) MQITII_NONE.

MQCHAR transactionState() const ;
Devuelve el **estado de transacción**.

void setTransactionState(const MQCHAR *estado*);
Establece el **estado de transacción**.

Datos de objeto (protegidos)

MQIIH *omqiih*
Estructura de datos MQIIH.

códigos de razón

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR

Clase C++ ImqItem

Esta clase abstracta representa un elemento, quizás uno de varios, dentro de un mensaje.

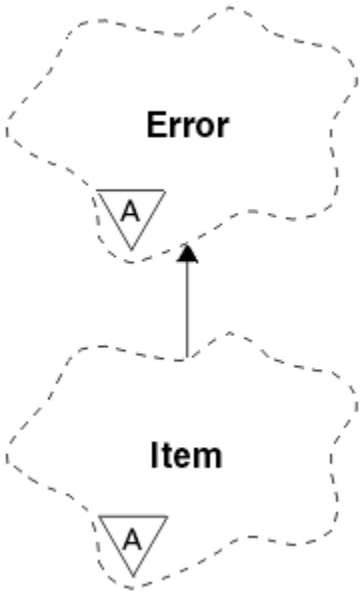


Figura 57. Clase ImqItem

Los elementos se concatenan en un almacenamiento intermedio de mensajes. Cada especialización está asociada con una estructura de datos determinada que empieza con un ID de estructura.

Los métodos polimórficos de esta clase abstracta permiten que los elementos se copien a y desde los mensajes. La clase ImqMessage **readItem** y los métodos **writeItem** proporcionan otro estilo de invocación de estos métodos polimórficos que es más natural para los programas de aplicación.

- "Atributos de objetos" en la página 1365
- "Constructores" en la página 1365
- "Métodos de clase (public)" en la página 1365
- "Métodos de objeto (public)" en la página 1366
- "códigos de razón" en la página 1366

Atributos de objetos

ID de estructura
Una serie de cuatro caracteres al principio de la estructura de datos. Este atributo es de sólo lectura. Considere este atributo para las clases derivadas. No se incluye automáticamente.

Constructores

ImqItem( );
El constructor predeterminado.

ImqItem(const ImqItem & *elemento*);
El constructor de copia.

Métodos de clase (public)

static ImqBoolean structureIdIs(const char * *structure-id-to-test*, const ImqMessage & *msg*);
Devuelve TRUE si el **ID de estructura** del siguiente ImqItem en el *msg* entrante es el mismo que *structure-id-to-test*. El siguiente elemento se identifica como la parte del almacenamiento intermedio de mensajes a la que se dirige actualmente el **puntero de datos** ImqCache . Este método se basa en el **ID de estructura** y, por lo tanto, no se garantiza que funcione para todas las clases derivadas de ImqItem .

Métodos de objeto (public)

void operator = (const ImqItem & ***elemento*** **);**
Copia datos de instancia de *elemento*, sustituyendo los datos de instancia existentes.

virtual ImqBoolean copyOut(ImqMessage & ***msg*** **) = 0;**
Escribe este objeto como el siguiente elemento en un almacenamiento intermedio de mensajes de salida, añadiéndolo a los elementos existentes. Si la operación de grabación es satisfactoria, aumenta la **longitud de datos** ImqCache . Este método devuelve TRUE si es satisfactorio.

Altere temporalmente este método para trabajar con una subclase específica.

virtual ImqBoolean pasteIn(ImqMessage & ***msg*** **) = 0;**
Lee este objeto *de forma destructiva* desde el almacenamiento intermedio de mensajes de entrada. La lectura es destructiva en la que se mueve el **puntero de datos** ImqCache . Sin embargo, el contenido del almacenamiento intermedio sigue siendo el mismo, por lo que los datos se pueden volver a leer restableciendo el **puntero de datos** ImqCache .

La clase (sub) de este objeto debe ser coherente con el **ID de estructura** que se encuentra a continuación en el almacenamiento intermedio de mensajes del objeto *msg* .

La **codificación** del objeto *msg* debe ser MQENC_NATIVE. Se recomienda que los mensajes se recuperen con la **codificación** ImqMessage establecida en MQENC_NATIVE, y con las opciones de ImqGetMessageOptions que incluyen MQGMO_CONVERT.

Si la operación de lectura es satisfactoria, la **longitud de datos** ImqCache se reduce. Este método devuelve TRUE si es satisfactorio.

Altere temporalmente este método para trabajar con una subclase específica.

códigos de razón

- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR
- MQRC_INCONSISTENT_FORMAT
- MQRC_INSUFICIENT_BUFFER
- MQRC_INSUFICIENT_DATA

Clase C++ ImqMessage

Esta clase encapsula una estructura de datos MQMD y también maneja la construcción y reconstrucción de datos de mensaje.

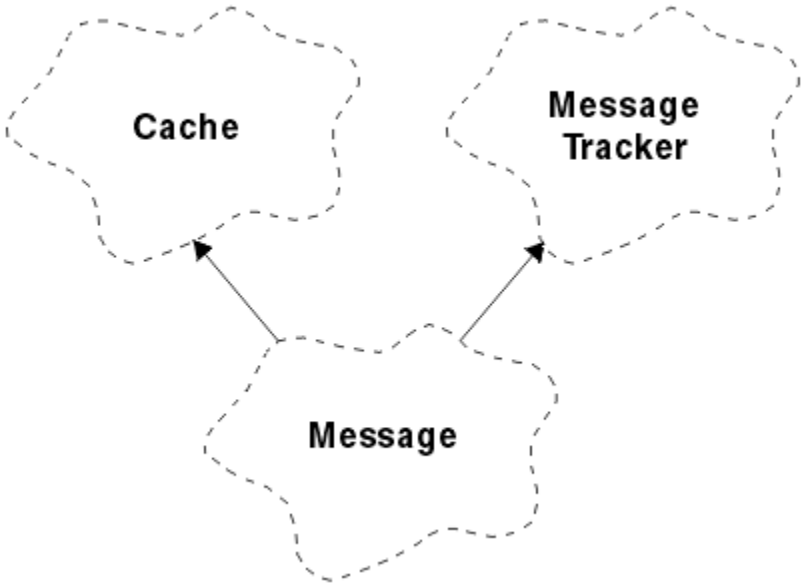


Figura 58. Clase ImqMessage

- "Atributos de objetos" en la página 1367
- "Constructores" en la página 1371
- "Métodos de objeto (public)" en la página 1371
- "Métodos de objeto (protegidos)" en la página 1373
- "Datos de objeto (protegidos)" en la página 1373

Atributos de objetos

Datos de ID de aplicación
Información de identidad asociada a un mensaje. El valor inicial es una serie nula.

Datos de origen de aplicación
Información de origen asociada a un mensaje. El valor inicial es una serie nula.

Recuento de restituciones
El número de veces que un mensaje se ha recuperado provisionalmente y se ha restituido posteriormente. El valor inicial es cero. Este atributo es de sólo lectura.

juego de caracteres
ID de juego de caracteres codificado. El valor inicial es MQCCSI_Q_MGR. Son posibles los siguientes valores adicionales:

- MQCCSI_INHERIT
- MQCCSI_EMBEDDED

También puede utilizar un ID de juego de caracteres codificado de su elección. Para obtener información sobre esto, consulte "Conversión de páginas de códigos" en la página 920.

Encoding
La codificación de máquina de los datos del mensaje. El valor inicial es MQENC_NATIVE.

caducidad
Cantidad dependiente del tiempo que controla cuánto tiempo WebSphere MQ retiene un mensaje no recuperado antes de descartarlo. El valor inicial es MQEI_UNLIMITED.

Format
El nombre del formato (plantilla) que describe el diseño de los datos en el almacenamiento intermedio. Los nombres de más de ocho caracteres se truncan a ocho caracteres. Los nombres siempre se rellenan con espacios en blanco hasta los ocho caracteres. El valor de constante inicial es MQFMT_NONE. Son posibles las siguientes constantes adicionales:

- MQFMT_ADMIN
- MQFMT_CICS
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_DIST_HEADER
- MQFMT_EVENT
- MQFMT_IMS
- MQFMT_IMS_VAR_STRING
- MQFMT_MD_EXTENSION
- MQFMT_PCF
- MQFMT_REF_MSG_HEADER
- MQFMT_RF_HEADER
- MQFMT_STRING
- MQFMT_TRIGGER
- MQFMT_WORK_INFO_HEADER
- MQFMT_XMIT_Q_HEADER

También puede utilizar una serie específica de la aplicación de su elección. Para obtener más información sobre esto, consulte el campo "Format (MQCHAR8)" en la página 408 del descriptor de mensaje (MQMD).

Distintivos de mensaje
Información de control de segmentación. El valor inicial es MQMF_SEGMENTATION_INHIBIDO. Son posibles los siguientes valores adicionales:

- MQMF_SEGMENTATION_ALLOWED
- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- SEGMENTO MQMF_SEGMENT
- MQMF_LAST_SEGMENT
- MQMF_NONE

tipo de mensaje
La amplia categorización de un mensaje. El valor inicial es MQMT_DATAGRAM. Son posibles los siguientes valores adicionales:

- MQMT_SYSTEM_FIRST
- MQMT_SYSTEM_LAST
- MQMT_DATAGRAM
- MQMT_REQUEST
- MQMT_REPLY
- MQMT_REPORT
- MQMT_APPL_PRIMERO
- MQMT_APPL_LAST

También puede utilizar un valor específico de la aplicación de su elección. Para obtener más información sobre esto, consulte el campo "MsgType (MQLONG)" en la página 419 del descriptor de mensaje (MQMD).

desplazamiento
Información de desplazamiento. El valor inicial es cero.

Longitud original
La longitud original de un mensaje segmentado. El valor inicial es MQOL_UNDEFINED.

Persistencia
Indica que el mensaje es importante y que se debe realizar una copia de seguridad en todo momento utilizando el almacenamiento persistente. Esta opción implica una penalización del rendimiento. El valor inicial es MQPER_PERSISTENCE_AS_Q_DEF. Son posibles los siguientes valores adicionales:

- MQPER_PERSISTENT
- MQPER_NOT_PERSISTENT

priority
Prioridad relativa para la transmisión y entrega. Los mensajes de la misma prioridad suelen entregarse en la misma secuencia en la que se suministraron (aunque hay varios criterios que deben cumplirse para garantizarlo). El valor inicial es MQPRI_PRIORITY_AS_Q_DEF.

Validación de la propiedad
Especifica si la validación de las propiedades debe tener lugar cuando se establece una propiedad del mensaje. El valor inicial es **MQCMHO_DEFAULT_VALIDATION**. Son posibles los siguientes valores adicionales:

- MQCMHO_VALIDAR
- MQCMHO_NO_VALIDATION

Los métodos siguientes actúan en la **validación de propiedades**:

MQLONG propertyValidation() Const.
Devuelve la opción **property validation** .

void setPropertyValidation (const MQLONG *opción*);
Establece la opción **property validation** .

Nombre de aplicación de transferencia
El nombre de la aplicación que ha colocado un mensaje. El valor inicial es una serie nula.

Tipo de aplicación de transferencia
El tipo de aplicación que ha colocado un mensaje. El valor inicial es MQAT_NO_CONTEXT. Son posibles los siguientes valores adicionales:

- MQAT_AIX
- MQAT_CICS
- MQAT_CICS_BRIDGE
- MQAT_DOS
- MQAT_IMS
- MQAT_IMS_BRIDGE
- MQAT_MVS
- MQAT_NOTES_AGENT
- MQAT_OS2
- MQAT_OS390
- MQAT_OS400
- MQAT_QMGR
- MQAT_UNIX
- MQAT_WINDOWS
- MQAT_WINDOWS_NT
- MQAT_XCF
- MQAT_DEFAULT
- MQAT_DESCONOCIDO

- MQAT_USER_FIRST
- MQAT_USER_LAST

También puede utilizar una serie específica de la aplicación de su elección. Para obtener más información sobre esto, consulte el campo "PutApplType (MQLONG)" en la página 425 del descriptor de mensaje (MQMD).

Fecha de transferencia
La fecha en la que se ha colocado un mensaje. El valor inicial es una serie nula.

Hora de transferencia
Hora a la que se ha colocado un mensaje. El valor inicial es una serie nula.

Nombre de gestor de cola de respuestas
El nombre del gestor de colas al que se debe enviar cualquier respuesta. El valor inicial es una serie nula.

Nombre de cola de respuestas
El nombre de la cola a la que se debe enviar cualquier respuesta. El valor inicial es una serie nula.

report
Información de comentarios asociada a un mensaje. El valor inicial es MQRO_NONE. Son posibles los siguientes valores adicionales:

- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA *
- MQRO_EXPIRATION
- MQRO_EXPIRATION_WITH_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA *
- MQRO_COA
- MQRO_COA_WITH_DATA
- MQRO_COA_WITH_FULL_DATA *
- MQRO_COD
- MQRO_COD_WITH_DATA
- MQRO_COD_WITH_FULL_DATA *
- MQRO_PAN
- MQRO_NAN
- MQRO_NEW_MSG_ID
- MQRO_NEW_CORREL_ID
- MQRO_COPY_MSG_ID_TO_CORREL_ID
- MQRO_PASS_CORREL_ID
- MQRO_DEAD_LETTER_Q
- MQRO_DISCARD_MSG

donde * indica valores que no están soportados en WebSphere MQ para z/OS.

número de secuencia
Información de secuencia que identifica un mensaje dentro de un grupo. El valor inicial es uno.

longitud total de mensaje
Número de bytes que estaban disponibles durante el intento más reciente de leer un mensaje. Este número será mayor que la **longitud de mensaje** ImqCache si el último mensaje se ha truncado, o si el último mensaje no se ha leído porque se habría producido un truncamiento. Este atributo es de sólo lectura. El valor inicial es cero.

Este atributo puede ser útil en cualquier situación que implique mensajes truncados.

ID de usuario
Identidad de usuario asociada a un mensaje. El valor inicial es una serie nula.

Constructores

ImqMessage( );
El constructor predeterminado.

ImqMessage(const ImqMessage & ***msg*** **);**
El constructor de copia. Consulte el método **operator =** para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqMessage & ***msg*** **);**
Copia el MQMD y los datos de mensaje de *msg*. Si el usuario ha proporcionado un almacenamiento intermedio para este objeto, la cantidad de datos copiados está restringida al tamaño de almacenamiento intermedio disponible. De lo contrario, el sistema se asegura de que haya disponible un almacenamiento intermedio de tamaño adecuado para los datos copiados.

ImqString applicationIdDatos() const ;
Devuelve una copia de los **datos de ID de aplicación**.

void setApplicationIdData(const char * ***data*** **= 0);**
Establece los **datos de ID de aplicación**.

ImqString applicationOriginDatos() const ;
Devuelve una copia de los **datos de origen de aplicación**.

void setApplicationOriginData(const char * ***data*** **= 0);**
Establece los **datos de origen de aplicación**.

MQLONG backoutCount() const ;
Devuelve el **recuento de restituciones**.

MQLONG characterSet() const ;
Devuelve el **juego de caracteres**.

void setCharacterSet(const MQLONG ***ccsid*** **= MQCCSI_Q_MGR);**
Establece el **juego de caracteres**.

MQLONG codificación() const ;
Devuelve la **codificación**.

void setEncoding(const MQLONG ***encoding*** **= MQENC_NATIVE);**
Establece la **codificación**.

MQLONG Caducidad() const ;
Devuelve la **caducidad**.

void setExpiry(const MQLONG ***caducidad*** **);**
Establece la **caducidad**.

ImqString formato() const ;
Devuelve una copia del **formato**, incluidos los espacios en blanco finales.

ImqBoolean formatIs(const char * ***formato-a-prueba*** **) const ;**
Devuelve TRUE si el formato de es el mismo que el formato de prueba de .

void setFormat(const char * ***name*** **= 0);**
Establece el **formato**, rellenado a ocho caracteres con blancos de cola.

MQLONG messageFlags() const ;
Devuelve los **distintivos de mensaje**.

void setMessageFlags(const MQLONG ***distintivos*** **);**
Establece los **distintivos de mensaje**.

MQLONG messageType() const ;
Devuelve el **tipo de mensaje**.

void setMessageType(const MQLONG *tipo*);
Establece el **tipo de mensaje**.

MQLONG Desplazamiento() const ;
Devuelve el **desplazamiento**.

void setOffset(const MQLONG *desplazamiento*);
Establece el **desplazamiento**.

MQLONG originalLength() const ;
Devuelve la **longitud original**.

void setOriginalLength(const MQLONG *longitud*);
Establece la **longitud original**.

MQLONG persistencia() const ;
Devuelve la **persistencia**.

void setPersistence(const MQLONG *persistencia*);
Establece la **persistencia**.

MQLONG prioridad() const ;
Devuelve la **prioridad**.

void setPriority(const MQLONG *prioridad*);
Establece la **prioridad**.

ImqString putApplicationNombre() const ;
Devuelve una copia del **nombre de aplicación de colocación**.

void setPutApplicationName(const char * *name* = 0);
Establece el **nombre de aplicación de colocación**.

MQLONG putApplicationTipo() const ;
Devuelve el **tipo de aplicación put**.

void setPutApplicationType(const MQLONG *type* = MQAT_NO_CONTEXT);
Establece el **tipo de aplicación put**.

ImqString putDate() const ;
Devuelve una copia de la **fecha de colocación**.

void setPutDate(const char * *date* = 0);
Establece la **fecha de colocación**.

ImqString putTime() const ;
Devuelve una copia de la **hora de colocación**.

void setPutTime(const char * *time* = 0);
Establece la **hora de colocación**.

ImqBoolean readItem(ImqItem & *elemento*);
Lee en el objeto *item* del almacenamiento intermedio de mensajes, utilizando el método ImqItem **pasteIn** . Devuelve TRUE si es satisfactorio.

ImqString replyToQueueManagerNombre() const ;
Devuelve una copia del **nombre de gestor de colas de respuesta**.

void setReplyToQueueManagerName(const char * *name* = 0);
Establece el **nombre de gestor de colas de respuesta**.

ImqString replyToQueueName() const ;
Devuelve una copia del **nombre de cola de respuesta**.

void setReplyToQueueName(const char * *name* = 0);
Establece el **nombre de cola de respuesta**.

MQLONG informe() const ;
Devuelve el **informe**.

void setReport(const MQLONG *informe*);
Establece el **informe**.

MQLONG sequenceNumber() const ;
Devuelve el **número de secuencia**.

void setSequenceNumber(const MQLONG *número*);
Establece el **número de secuencia**.

size_t totalMessageLongitud() const ;
Devuelve la **longitud total de mensaje**.

ImqString userId() const ;
Devuelve una copia del **ID de usuario**.

void setUserId(const char * *id* = 0);
Establece el **ID de usuario**.

ImqBoolean writeItem(ImqItem & *elemento*);
Escribe desde el objeto *item* en el almacenamiento intermedio de mensajes, utilizando el método ImqItem **copyOut** . La escritura puede tomar la forma de inserción, sustitución o adición: depende de la clase del objeto *elemento* . Este método devuelve TRUE si es satisfactorio.

Métodos de objeto (protegidos)

static void setVersionSupported(const MQLONG);
Establece la **versión de MQMD**. El valor predeterminado es **MQMD_VERSION_2**.

Datos de objeto (protegidos)

MQMD1 *omqmd*
(Solo WebSphere MQ para z/OS .) La estructura de datos MQMD.

MQMD2 *omqmd*
(Plataformas distintas de z/OS.) La estructura de datos MQMD.

Clase C++ de rastreador ImqMessage

Esta clase encapsula los atributos de un objeto ImqMessage o ImqQueue que se pueden asociar con cualquiera de los objetos.

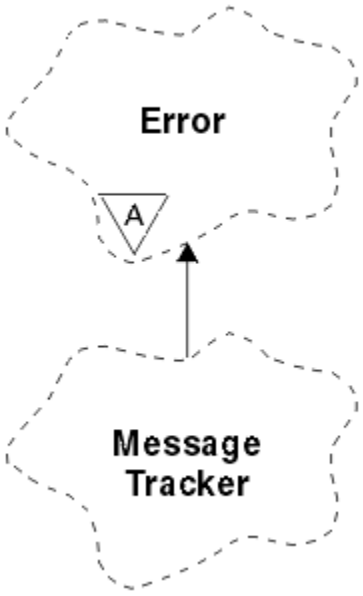


Figura 59. Clase de rastreador ImqMessage

Esta clase está relacionada con las llamadas MQI listadas en “Referencia cruzada de ImqMessageTracker” en la página 1322.

- “Atributos de objetos” en la página 1374

- “Constructores” en la página 1375
- “Métodos de objeto (public)” en la página 1375
- “códigos de razón” en la página 1376

Atributos de objetos

Señal de contabilidad
Un valor binario (MQBYTE32) de longitud MQ_ACCOUNTING_TOKEN_LENGTH. El valor inicial es MQACT_NONE.

ID de correlación
Un valor binario (MQBYTE24) de longitud MQ_CORREL_ID_LENGTH que se asigna para correlacionar mensajes. El valor inicial es MQCI_NONE. El valor adicional, MQCI_NEW_SESSION, es posible.

Feedback
Información de retroalimentación que se enviará con un mensaje. El valor inicial es MQFB_NONE. Son posibles los siguientes valores adicionales:

- MQFB_SISTEMA_PRIMERO
- MQFB_SYSTEM_LAST
- MQFB_APPL_FIRST
- MQFB_APPL_LAST
- MQFB_COA
- MQFB_COD
- MQFB_EXPIRATION
- MQFB_PAN
- MQFB_NAN
- MQFB_QUIT
- MQFB_DATA_LENGTH_ZERO
- MQFB_DATA_LENGTH_NEGATIVO
- MQFB_DATA_LENGTH_TOO_BIG
- MQFB_BUFFER_OVERFLOW
- MQFB_LENGTH_OFF_BY_ONE
- MQFB_IIH_ERROR
- MQFB_NOT_AUTHORIZED_FOR_IMS
- MQFB_IMS_ERROR
- MQFB_IMS_FIRST
- MQFB_IMS_LAST
- MQFB_CICS_APPL_ABENDED
- MQFB_CICS_APPL_NOT_STARTED
- MQFB_CICS_BRIDGE_FAILURE
- MQFB_CICS_CCSID_ERROR
- MQFB_CICS_CIH_ERROR
- MQFB_CICS_COMMAREA_ERROR
- MQFB_CICS_CORREL_ID_ERROR
- MQFB_CICS_DLQ_ERROR
- MQFB_CICS_ENCODING_ERROR
- MQFB_CICS_INTERNAL_ERROR
- MQFB_CICS_NOT_AUTHORIZED

- MQFB_CICS_UOW_BACKED_OUT
- MQFB_CICS_UOW_ERROR

También puede utilizar una serie específica de la aplicación de su elección. Para obtener más información sobre esto, consulte el campo "Feedback (MQLONG)" en la página 404 del descriptor de mensaje (MQMD).

ID de grupo
Un valor binario (MQBYTE24) de longitud MQ_GROUP_ID_LENGTH exclusivo dentro de una cola. El valor inicial es MQGI_NONE.

ID de mensaje
Un valor binario (MQBYTE24) de longitud MQ_MSG_ID_LENGTH exclusivo dentro de una cola. El valor inicial es MQMI_NONE.

Constructores

ImqMessageTracker ();
El constructor predeterminado.

ImqMessageTracker (const ImqMessageTracker & *rastreador*);
El constructor de copia. Consulte el método **operator =** para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqMessageTracker & *rastreador*);
Copia datos de instancia de *Tracker*, sustituyendo los datos de instancia existentes.

ImqBinary accountingToken() const ;
Devuelve una copia de la **señal de contabilidad**.

ImqBoolean setAccountingToken(const ImqBinary & *señal*);
Establece la **señal de contabilidad**. La **longitud de datos** de *señal* debe ser cero o MQ_ACCOUNTING_TOKEN_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setAccountingToken(const MQBYTE32 *señal* = 0);
Establece la **señal de contabilidad**. *token* puede ser cero, que es lo mismo que especificar MQACT_NONE. Si *token* es distinto de cero, debe direccionar MQ_ACCOUNTING_TOKEN_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como, por ejemplo, MQACT_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma; por ejemplo, (MQBYTE *) MQACT_NONE.

ImqBinary correlationId() const ;
Devuelve una copia del **ID de correlación**.

ImqBoolean setCorrelationId(const ImqBinary & *señal*);
Establece el **ID de correlación**. La **longitud de datos** de *señal* debe ser cero o MQ_CORREL_ID_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setCorrelationId(const MQBYTE24 *id* = 0);
Establece el **ID de correlación**. *id* puede ser cero, que es lo mismo que especificar MQCI_NONE. Si *id* es distinto de cero, debe direccionar MQ_CORREL_ID_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como, por ejemplo, MQCI_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma; por ejemplo, (MQBYTE *) MQCI_NONE.

MQLONG comentarios() const ;
Devuelve el **feedback**.

void setFeedback(const MQLONG *feedback*);
Establece los **comentarios**.

ImqBinary groupId() const ;
Devuelve una copia del **id de grupo**.

ImqBoolean setGroupId(const ImqBinary & *señal*);
Establece el **id de grupo**. La **longitud de datos** de *señal* debe ser cero o MQ_GROUP_ID_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setGroupId(const MQBYTE24 *id* = 0);
Establece el **id de grupo**. *id* puede ser cero, lo que equivale a especificar MQGI_NONE. Si *id* no es cero, debe direccionar MQ_GROUP_ID_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como MQGI_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma, por ejemplo (MQBYTE *) MQGI_NONE.

ImqBinary messageId() const ;
Devuelve una copia del **id de mensaje**.

ImqBoolean setMessageId(const ImqBinary & *señal*);
Establece el **id de mensaje**. La **longitud de datos** de *símbolo* debe ser cero o MQ_MSG_ID_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setMessageId(const MQBYTE24 *id* = 0);
Establece el **id de mensaje**. *id* puede ser cero, que es lo mismo que especificar MQMI_NONE. Si *id* es distinto de cero, debe direccionar los bytes MQ_MSG_ID_LENGTH de datos binarios. Cuando se utilizan valores predefinidos como MQMI_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma, por ejemplo (MQBYTE *) MQMI_NONE.

códigos de razón

- MQRC_BINARY_DATA_LENGTH_ERROR

Clase C++ ImqNamelist

Esta clase encapsula una lista de nombres.

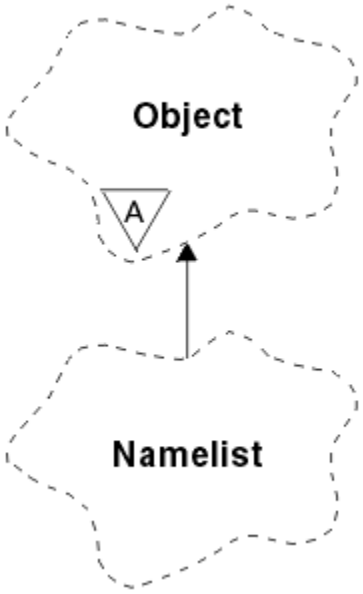


Figura 60. Clase ImqNamelist

Esta clase está relacionada con las llamadas MQI listadas en “Referencia cruzada de ImqNamelist” en la página 1322.

- “Atributos de objetos” en la página 1377
- “Constructores” en la página 1377
- “Métodos de objeto (public)” en la página 1377
- “códigos de razón” en la página 1377

Atributos de objetos

Número de nombres
El número de nombres de objeto en **nombres de lista de nombres**. Este atributo es de sólo lectura.

nombres de lista de nombres
Nombres de objeto, cuyo número se indica mediante el **recuento de nombres**. Este atributo es de sólo lectura.

Constructores

ImqNamelist( );
El constructor predeterminado.

ImqNamelist(const ImqNamelist & ***lista*** **);**
El constructor de copia. El **estado de apertura** de ImqObject es false.

ImqNamelist(const char * ***nombre*****);**
Establece el nombre de ImqObject en **name**.

Métodos de objeto (public)

void operator = (const ImqNamelist & ***lista*** **);**
Copia datos de instancia de *lista*, sustituyendo los datos de instancia existentes. El **estado de apertura** de ImqObject es false.

ImqBoolean nameCount(MQLONG & ***recuento*** **);**
Proporciona una copia del **recuento de nombres**. Devuelve TRUE si es satisfactorio.

MQLONG nameCount ();
Devuelve el **recuento de nombres** sin ninguna indicación de posibles errores.

ImqBoolean namelistName (const MQLONG ***índice*****, ImqString &** ***nombre*** **);**
Proporciona una copia de uno de los **nombres de lista de nombres** por índice basado en cero. Devuelve TRUE si es satisfactorio.

ImqString namelistName (const MQLONG ***índice*** **);**
Devuelve uno de los **nombres de lista de nombres** por índice basado en cero sin ninguna indicación de posibles errores.

códigos de razón

- MQRC_INDEX_ERROR
- MQRC_INDEX_NOT_PRESENT

Clase C++ ImqObject

Esta clase es abstracta. Cuando se destruye un objeto de esta clase, se cierra automáticamente y se interrumpe la conexión del gestor ImqQueue.

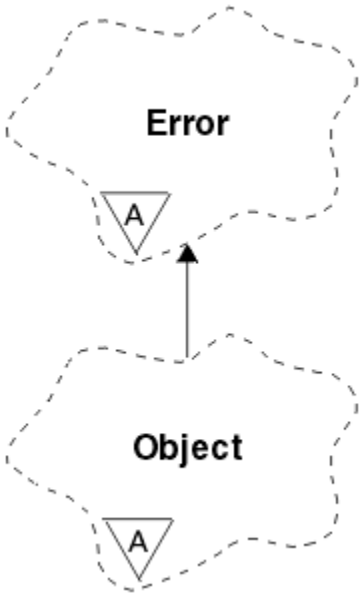


Figura 61. Clase ImqObject

Esta clase está relacionada con las llamadas MQI listadas en “Referencia cruzada de ImqObject” en la página 1323.

Atributos de clase

comportamiento
Controla el comportamiento de la apertura implícita.

IMQ_IMPL_OPEN (8L)
Se permite la apertura implícita. Éste es el valor predeterminado.

Atributos de objetos

Fecha de modificación
La fecha de modificación. Este atributo es de sólo lectura.

Hora de modificación
El tiempo de alteración. Este atributo es de sólo lectura.

ID de usuario alternativo
El ID de usuario alternativo, con un máximo de MQ_USER_ID_LENGTH caracteres. El valor inicial es una serie nula.

ID de seguridad alternativo
El ID de seguridad alternativo. Un valor binario (MQBYTE40) de longitud MQ_SECURITY_ID_LENGTH. El valor inicial es MQSID_NONE.

opciones de cierre
Opciones que se aplican cuando se cierra un objeto. El valor inicial es MQCO_NONE. Este atributo se ignora durante las operaciones de reapertura implícitas, donde siempre se utiliza un valor de MQCO_NONE.

referencia de conexión
Referencia a un objeto de gestor ImqQueueque proporciona la conexión necesaria a un gestor de colas (local). Para un objeto de gestor ImqQueue, es el propio objeto. El valor inicial es cero.

Nota: No confunda esto con el **nombre de gestor de colas** que identifica un gestor de colas (posiblemente remoto) para una cola con nombre.

description
Nombre descriptivo (hasta 64 caracteres) del gestor de colas, cola, lista de nombres o proceso. Este atributo es de sólo lectura.

name
El nombre (hasta 48 caracteres) del gestor de colas, cola, lista de nombres o proceso. El valor inicial es una serie nula. El nombre de una cola modelo cambia después de una **apertura** por el nombre de la cola dinámica resultante.

Nota: Un gestor de ImqQueuepuede tener un nombre nulo, que representa el gestor de colas predeterminado. El nombre cambia al gestor de colas real después de una **apertura**correcta. Una lista de ImqDistributiones dinámica y debe tener un nombre nulo.

siguiente objeto gestionado
Este es el siguiente objeto de esta clase, en ningún orden concreto, que tiene la misma **referencia de conexión** que este objeto. El valor inicial es cero.

Opciones abiertas
Opciones que se aplican cuando se abre un objeto. El valor inicial es MQOO_INQUIRE. Hay dos formas de establecer los valores adecuados:

1. No establezca las **opciones de apertura** y no utilice el método **open** . WebSphere MQ ajusta automáticamente las **opciones de apertura** y abre, reabre y cierra automáticamente los objetos según sea necesario. Esto puede dar como resultado operaciones de reapertura innecesarias, porque WebSphere MQ utiliza el método **openFor** y esto añade **opciones de apertura** sólo de forma incremental.
2. Establezca las **opciones de apertura** antes de utilizar cualquier método que dé como resultado una llamada MQI (consulte “Referencia cruzada de C++ y MQI” en la página 1316). Esto garantiza que no se produzcan operaciones de reapertura innecesarias. Establezca las opciones de apertura de forma explícita si es probable que se produzca alguno de los posibles problemas de reapertura (consulte Reabrir).

   Si utiliza el método **open** , ***debe*** asegurarse de que las **opciones de apertura** son las adecuadas en primer lugar. Sin embargo, el uso del método **open** no es obligatorio; WebSphere MQ sigue mostrando el mismo comportamiento que en el caso 1, pero en esta circunstancia, el comportamiento es eficiente.

Cero no es un valor válido; establezca el valor adecuado antes de intentar abrir el objeto. Esto se puede realizar utilizando **setOpenOptions**(*lOpenOptions*) seguido de **open**() o **openFor**(*lRequiredOpenOption*).

Nota:

1. MQOO_OUTPUT se sustituye por MQOO_INQUIRE durante el método **open** para una lista de distribución, ya que MQOO_OUTPUT es la única **opción de apertura** válida en este momento. Sin embargo, se recomienda establecer MQOO_OUTPUT de forma explícita en los programas de aplicación que utilizan el método **open** .
2. Especifique MQOO_RESOLVE_NAMES si desea utilizar los atributos **nombre de gestor de colas resuelto** y **nombre de cola resuelto** de la clase.

estado abierto
Si el objeto está abierto (TRUE) o cerrado (FALSE). El valor inicial es FALSE. Este atributo es de sólo lectura.

objeto gestionado anterior
El objeto anterior de esta clase, en ningún orden concreto, que tiene la misma **referencia de conexión** que este objeto. El valor inicial es cero.

identificador de gestor de colas
El identificador del gestor de colas. Este atributo es de sólo lectura.

Constructores

ImqObject( );
El constructor predeterminado.

ImqObject(const ImqObject & ***objeto*** **);**
El constructor de copia. El **estado abierto** será FALSE.

Métodos de clase (public)

comportamiento MQLONG estático ();
Devuelve el **comportamiento**.

void setBehavior(const MQLONG ***behavior*** **= 0);**
Establece el **comportamiento**.

Métodos de objeto (public)

void operator = (const ImqObject & ***objeto*** **);**
Realiza un cierre si es necesario y copia los datos de instancia del objeto de . El **estado abierto** será FALSE.

ImqBoolean alterationDate(ImqString & ***fecha*** **);**
Proporciona una copia de la **fecha de modificación**. Devuelve TRUE si es satisfactorio.

ImqString alterationDate( );
Devuelve la **fecha de modificación** sin ninguna indicación de posibles errores.

ImqBoolean alterationTime(ImqString & ***hora*** **);**
Proporciona una copia del **tiempo de modificación**. Devuelve TRUE si es satisfactorio.

ImqString alterationTime( );
Devuelve el **tiempo de modificación** sin ninguna indicación de posibles errores.

ImqString alternateUserId() const ;
Devuelve una copia del **ID de usuario alternativo**.

ImqBoolean setAlternateUserId(const char * ***id*** **);**
Establece el **ID de usuario alternativo**. El **ID de usuario alternativo** sólo se puede establecer cuando el **estado abierto** es FALSE. Este método devuelve TRUE si es satisfactorio.

ImqBinary alternateSecurityId () Const.
Devuelve una copia del **ID de seguridad alternativo.**

ImqBoolean setAlternateSecurityId(const ImqBinary & ***señal*** **);**
Establece el **ID de seguridad alternativo**. El **ID de seguridad alternativo** sólo se puede establecer mientras el **estado abierto** es FALSE. La longitud de datos de *señal* debe ser cero o MQ_SECURITY_ID_LENGTH. Devuelve TRUE si es satisfactorio.

ImqBoolean setAlternateSecurityId(const MQBYTE* ***señal*** **= 0);**
Establece el **ID de seguridad alternativo**. *token* puede ser cero, que es lo mismo que especificar MQSID_NONE. Si *token* no es cero, debe direccionar MQ_SECURITY_ID_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como, por ejemplo, MQSID_NONE, es posible que tenga que realizar una conversión para garantizar la coincidencia de firma; por ejemplo, (MQBYTE *) MQSID_NONE.

El **ID de seguridad alternativo** sólo se puede establecer mientras el **estado abierto** es TRUE. Devuelve TRUE si es satisfactorio.

ImqBoolean setAlternateSecurityId(const unsigned char * *id* = 0);
Establece el **ID de seguridad alternativo**.

ImqBoolean cerrar();
Establece el **estado abierto** en FALSE. Devuelve TRUE si es satisfactorio.

MQLONG closeOptions() const ;
Devuelve las **opciones de cierre**.

void setCloseOptions(const MQLONG *opciones*);
Establece las **opciones de cierre**.

ImqQueueManager * connectionReference() const ;
Devuelve la **referencia de conexión**.

void setConnectionReference(ImqQueueManager & *gestor*);
Establece la **referencia de conexión**.

void setConnectionReference(ImqQueueManager * *manager* = 0);
Establece la **referencia de conexión**.

virtual ImqBoolean description(ImqString & *descripción*) = 0;
Proporciona una copia de la **descripción**. Devuelve TRUE si es satisfactorio.

ImqString descripción();
Devuelve una copia de la **descripción** sin ninguna indicación de posibles errores.

virtual ImqBoolean nombre(ImqString & *nombre*);
Proporciona una copia del **nombre**. Devuelve TRUE si es satisfactorio.

ImqString nombre();
Devuelve una copia del **nombre** sin ninguna indicación de posibles errores.

ImqBoolean setName(const char * *nombre* = 0);
Establece el **nombre**. El **nombre** sólo se puede establecer mientras el **estado abierto** es FALSE y, para un gestor ImqQueue, mientras que el **estado de conexión** es FALSE. Devuelve TRUE si es satisfactorio.

ImqObject * nextManagedObject() const ;
Devuelve el **siguiente objeto gestionado**.

ImqBoolean open();
Cambia el **estado abierto** a TRUE abriendo el objeto según sea necesario, utilizando entre otros atributos las **opciones de apertura** y el **nombre**. Este método utiliza la información de **referencia de conexión** y el método ImqQueueManager **connect** si es necesario para asegurarse de que el estado de conexión de ImqQueueManager es TRUE. Devuelve el **estado abierto**.

ImqBoolean openFor(const MQLONG *opciones-necesarias* = 0);
Intenta asegurarse de que el objeto está abierto con **opciones de apertura**, o con **opciones de apertura** que garantizan el comportamiento implícito en el valor del parámetro *opciones-necesarias* .

Si *opciones-necesarias* es cero, la entrada es necesaria y cualquier opción de entrada es suficiente. Por lo tanto, si las **opciones de apertura** ya contienen una de las siguientes:

- MQOO_INPUT_AS_Q_DEF
- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE

las **opciones de apertura** ya son satisfactorias y no se modifican; si las **opciones de apertura** todavía no contienen ninguna de estas opciones, MQOO_INPUT_AS_Q_DEF se establece en las **opciones de apertura**.

Si *opciones-necesarias* es distinto de cero, las opciones necesarias se añaden a las **opciones de apertura**; si *opciones-necesarias* es cualquiera de estas opciones, las demás se restablecerán.

Si se cambia alguna de las **opciones de apertura** y el objeto ya está abierto, el objeto se cierra temporalmente y se vuelve a abrir para ajustar las **opciones de apertura**.

Devuelve TRUE si es satisfactorio. El éxito indica que el objeto está abierto con las opciones adecuadas.

MQLONG openOptions() const ;
Devuelve las **opciones de apertura**.

ImqBoolean setOpenOptions(const MQLONG *opciones*);
Establece las **opciones de apertura**. Las **opciones de apertura** sólo se pueden establecer mientras el **estado de apertura** es FALSE. Devuelve TRUE si es satisfactorio.

ImqBoolean openStatus() const ;
Devuelve el **estado abierto**.

ImqObject * previousManagedObject() const ;
Devuelve el **objeto gestionado anterior**.

ImqBoolean queueManagerIdentifier (ImqString & *id*);
Proporciona una copia del **identificador de gestor de colas**. Devuelve TRUE si es satisfactorio.

ImqString queueManagerIdentificador ();
Devuelve el **identificador de gestor de colas** sin ninguna indicación de posibles errores.

Métodos de objeto (protegidos)

virtual ImqBoolean closeTemporarily();
Cierra un objeto de forma segura antes de volver a abrirlo. Devuelve TRUE si es satisfactorio. Este método presupone que el **estado abierto** es TRUE.

MQHCONN connectionHandle() const ;
Devuelve el MQHCONN asociado a la **referencia de conexión**. Este valor es cero si no hay ninguna **referencia de conexión** o si el gestor no está conectado.

ImqBoolean inquire(const MQLONG *int-attr*, MQLONG & *valor*);
Devuelve un valor entero, cuyo índice es un valor MQIA_ *. En caso de error, el valor se establece en MQIAV_UNDEFINED.

ImqBoolean inquire(const MQLONG *char-attr*, char * & *buffer*, const size_t *longitud*);
Devuelve una serie de caracteres, cuyo índice es un valor MQCA_ *.

Nota: Ambos métodos sólo devuelven un único valor de atributo. Si se necesita una *instantánea* de más de un valor, donde los valores son coherentes entre sí durante un instante, WebSphere MQ C++ no proporciona este recurso y debe utilizar la llamada MQINQ con los parámetros adecuados.

vacío virtual openInformationDisperse();
Dispersa información de la sección de variables de la estructura de datos MQOD inmediatamente después de una llamada MQOPEN.

virtual ImqBoolean openInformationPrepare();
Prepara información para la sección de variables de la estructura de datos MQOD inmediatamente antes de una llamada MQOPEN y devuelve TRUE si es satisfactorio.

ImqBoolean set(const MQLONG *int-attr*, const MQLONG *valor*);
Establece un atributo de entero WebSphere MQ .

ImqBoolean set(const MQLONG *atr-car*, const char * *buffer*, const size_t *longitud-necesaria*);
Establece un atributo de carácter WebSphere MQ .

void setNextManagedObject(const ImqObject * *objeto* = 0);
Establece el **siguiente objeto gestionado**.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de objetos gestionados.

void setPreviousManagedObject(const ImqObject * *objeto* = 0);
Establece el **objeto gestionado anterior**.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de objetos gestionados.

Datos de objeto (protegidos)

MQHOBJ *ohobj*
El descriptor de objeto WebSphere MQ (sólo es válido cuando el **estado abierto** es TRUE).

MQOD *po_omq*
La estructura de datos MQOD incorporada. La cantidad de almacenamiento asignado para esta estructura de datos es la necesaria para una MQOD Versión 2. Inspeccione el número de versión *(omqod.Version)* y acceda a los otros campos de la forma siguiente:

MQOD_VERSION_1
Se puede acceder a todos los demás campos de *omqod* .

MQOD_VERSION_2
Se puede acceder a todos los demás campos de *omqod* .

MQOD_VERSION_3
omqod.pmqod es un puntero a un MQOD asignado dinámicamente, más grande. No se puede acceder a ningún otro campo de *omqod* . Se puede acceder a todos los campos direccionado por *omqod.pmqod* .

Nota: *omqod.pmqod.Version* puede ser menor que *omqod.Version*, lo que indica que el cliente MQI de WebSphere MQ tiene más funcionalidad que el servidor WebSphere MQ .

códigos de razón

- MQRC_ATTRIBUTE_LOCKED
- MQRC_INCONSISTENT_OBJECT_STATE
- MQRC_NO_CONNECTION_REFERENCE
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_REOPEN_SAVED_CONTEXT_ERR
- (códigos de razón de MQCLOSE)
- (códigos de razón de MQCONN)
- (códigos de razón de MQINQ)
- (códigos de razón de MQOPEN)
- (códigos de razón de MQSET)

Clase C++ ImqProcess

Esta clase encapsula un proceso de aplicación (un objeto WebSphere MQ de tipo MQOT_PROCESS) que puede desencadenar un supervisor desencadenante.

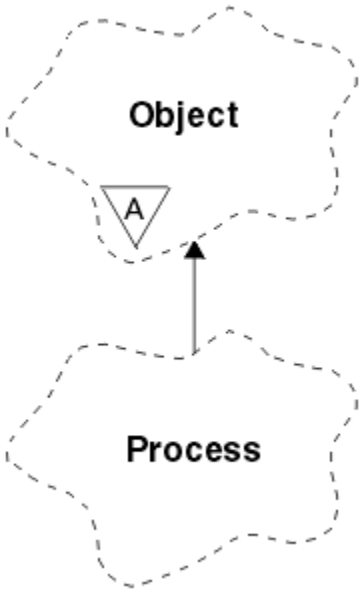


Figura 62. Clase ImqProcess

Atributos de objetos

ID de aplicación
La identidad del proceso de aplicación. Este atributo es de sólo lectura.

Tipo de aplicación
El tipo del proceso de aplicación. Este atributo es de sólo lectura.

Datos de entorno
La información de entorno para el proceso. Este atributo es de sólo lectura.

datos de usuario
Datos de usuario para el proceso. Este atributo es de sólo lectura.

Constructores

ImqProcess( );
El constructor predeterminado.

ImqProcess(const ImqProcess & *proceso*);
El constructor de copia. El **estado abierto** de ImqObject es FALSE.

ImqProcess(const char * *nombre*);
Establece el **nombre** ImqObject .

Métodos de objeto (public)

void operator = (const ImqProcess & *proceso*);
Realiza un cierre si es necesario y, a continuación, copia los datos de instancia de *proceso*. El **estado abierto** de ImqObject será FALSE.

ImqBoolean applicationId(ImqString & *id*);
Proporciona una copia del **ID de aplicación**. Devuelve TRUE si es satisfactorio.

ImqString applicationId( );
Devuelve el **ID de aplicación** sin ninguna indicación de posibles errores.

ImqBoolean applicationType(MQLONG & *tipo*);
Proporciona una copia del **tipo de aplicación**. Devuelve TRUE si es satisfactorio.

MQLONG applicationType();
Devuelve el **tipo de aplicación** sin ninguna indicación de posibles errores.

ImqBoolean environmentData(ImqString & *datos*);
Proporciona una copia de los **datos de entorno**. Devuelve TRUE si es satisfactorio.

ImqString environmentData( );
Devuelve los **datos de entorno** sin ninguna indicación de posibles errores.

ImqBoolean userData(ImqString & *datos*);
Proporciona una copia de los **datos de usuario**. Devuelve TRUE si es satisfactorio.

ImqString userData( );
Devuelve los **datos de usuario** sin ninguna indicación de posibles errores.

Clase C++ ImqPutMessageOptions

Esta clase encapsula la estructura de datos MQPMO.

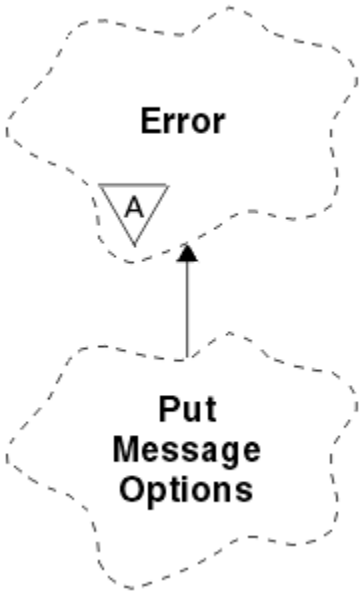


Figura 63. Clase ImqPutMessageOptions

- "Atributos de objetos" en la página 1385
- "Constructores" en la página 1386
- "Métodos de objeto (public)" en la página 1386
- "Datos de objeto (protegidos)" en la página 1387
- "códigos de razón" en la página 1387

Atributos de objetos

referencia de contexto
Una ImqQueue que proporciona un contexto para los mensajes. Inicialmente no hay ninguna referencia.

opciones
Las opciones de colocación de mensaje. El valor inicial es MQPMO_NONE. Son posibles los siguientes valores adicionales:

- MQPMO_SYNCPOINT
- MQPMO_NO_SYNCPOINT

- MQPMO_NEW_MSG_ID
- MQPMO_NEW_CORREL_ID
- MQPMO_LOGICAL_ORDER
- 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

campos de registro
Los distintivos que controlan la inclusión de registros de mensajes de colocación cuando se coloca un mensaje. El valor inicial es MQPMRF_NONE. Son posibles los siguientes valores adicionales:

- MQPMRF_MSG_ID
- ID MQPMRF_CORREL_ID
- MQPMRF_ID_grupo
- MQPMRF_FEEDBACK
- MQPMRF_ACCOUNTING_TOKEN

Los atributos del rastreador ImqMessagese toman del objeto para cualquier campo que se especifique. Los atributos de ImqMessageTracker se toman del objeto ImqMessage para cualquier campo que *no* se haya especificado.

Nombre del gestor de colas resuelto
Nombre de un gestor de colas de destino determinado durante una colocación. El valor inicial es nulo. Este atributo es de sólo lectura.

Nombre de cola resuelto
Nombre de una cola de destino determinada durante una colocación. El valor inicial es nulo. Este atributo es de sólo lectura.

participación de punto de sincronismo
TRUE cuando los mensajes se ponen bajo control de punto de sincronismo.

Constructores

ImqPutMessageOptions( );
El constructor predeterminado.

ImqPutMessageOptions(const ImqPutMessageOptions & *pmo*);
El constructor de copia.

Métodos de objeto (public)

void operator = (const ImqPutMessageOptions & *pmo*);
Copia datos de instancia de *pmo*, sustituyendo los datos de instancia existentes.

ImqQueue * contextReference() const ;
Devuelve la **referencia de contexto**.

void setContextReference(const ImqQueue & *cola*);
Establece la **referencia de contexto**.

void setContextReference(const ImqQueue * *queue* = 0);
Establece la **referencia de contexto**.

MQLONG opciones() const ;
Devuelve las **opciones**.

void setOptions(const MQLONG *opciones*);
Establece las **opciones**, incluido el valor de **participación de punto de sincronismo** .

MQLONG recordFields() const ;
Devuelve los **campos de registro**.

void setRecordFields(const MQLONG *campos*);
Establece los **campos de registro**.

ImqString resolvedQueueManagerName() const ;
Devuelve una copia del **nombre de gestor de colas resuelto**.

ImqString resolvedQueueNombre() const ;
Devuelve una copia del **nombre de cola resuelto**.

ImqBoolean syncPointParticipación() const ;
Devuelve el valor de **participación de punto de sincronismo** , que es TRUE si las **opciones** incluyen MQPMO_SYNCPOINT.

void setSyncPointParticipation(const ImqBoolean *sincronización*);
Establece el valor de **participación de punto de sincronismo** . Si *sync* es TRUE, las **opciones** se modifican para incluir MQPMO_SYNCPOINT y para excluir MQPMO_NO_SYNCPOINT. Si *sync* es FALSE, las **opciones** se modifican para incluir MQPMO_NO_SYNCPOINT y para excluir MQPMO_SYNCPOINT.

Datos de objeto (protegidos)

MQPMO *omqpmo*
Estructura de datos MQPMO.

códigos de razón

- MQRC_STORAGE_NOT_AVAILABLE

Clase C++ ImqQueue

Esta clase encapsula una cola de mensajes (un objeto WebSphere MQ de tipo MQOT_Q).

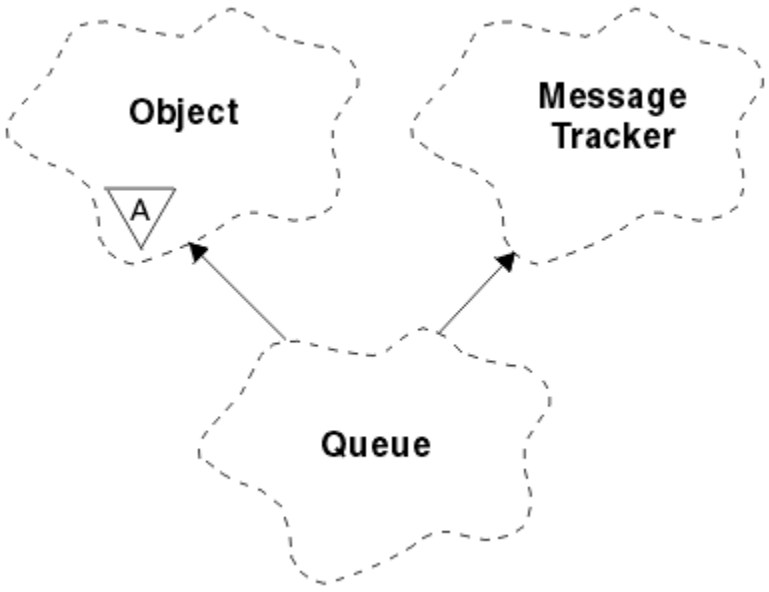


Figura 64. Clase ImqQueue

Esta clase está relacionada con las llamadas MQI listadas en Tabla 612 en la página 1324.

Atributos de objetos

Nombre de reposición en cola para restitución
Nombre de reposición en cola de restitución excesivo. Este atributo es de sólo lectura.

umbral de restituciones
El umbral de restituciones. Este atributo es de sólo lectura.

nombre de cola base
Nombre de la cola en la que se resuelve el alias. Este atributo es de sólo lectura.

Nombre de clúster
Nombre de clúster. Este atributo es de sólo lectura.

Nombre de la lista de nombres del clúster
Nombre de lista de nombres de clúster. Este atributo es de sólo lectura.

Rango de carga trabajo del clúster
El rango de la carga de trabajo del clúster. Este atributo es de sólo lectura.

Prioridad de carga de trabajo del clúster
La prioridad de la carga de trabajo del clúster. Este atributo es de sólo lectura.

Cola de uso de carga de trabajo de clúster
La carga de trabajo de clúster utiliza el valor de cola. Este atributo es de sólo lectura.

fecha de creación
Datos de creación de cola. Este atributo es de sólo lectura.

Hora de creación
Hora de creación de cola. Este atributo es de sólo lectura.

Profundidad actual
Número de mensajes en la cola. Este atributo es de sólo lectura.

enlace predeterminado
Enlace predeterminado. Este atributo es de sólo lectura.

Opción abierta de entrada predeterminada
Opción de apertura para entrada predeterminada. Este atributo es de sólo lectura.

Persistencia predeterminada
Persistencia de mensajes predeterminada. Este atributo es de sólo lectura.

Prioridad predeterminada
Prioridad de mensaje predeterminada. Este atributo es de sólo lectura.

Tipo de definición
El tipo de definición de la cola. Este atributo es de sólo lectura.

suceso de profundidad alta
Atributo de control para sucesos de profundidad de cola alta. Este atributo es de sólo lectura.

límite alto de profundidad
Límite alto para la profundidad de cola. Este atributo es de sólo lectura.

suceso de profundidad baja
Atributo de control para sucesos de profundidad de cola baja. Este atributo es de sólo lectura.

límite bajo de profundidad
Límite bajo para la profundidad de cola. Este atributo es de sólo lectura.

suceso de profundidad máxima
Atributo de control para sucesos máximos de profundidad de cola. Este atributo es de sólo lectura.

referencia de lista de distribución
Referencia opcional a una lista ImqDistributionque se puede utilizar para distribuir mensajes a más de una cola, incluida ésta. El valor inicial es nulo.

Nota: Cuando se abre un objeto ImqQueue , cualquier objeto de lista ImqDistributionabierto al que haga referencia se cierra automáticamente.

listas de distribución
Capacidad de una cola de transmisión para dar soporte a listas de distribución. Este atributo es de sólo lectura.

Nombre de la cola dinámica
Nombre de cola dinámica. El valor inicial es AMQ.* para todas las plataformas Windows, UNIXy Linux .

Copia en disco de restitución de obtención
Indica si se debe reforzar el recuento de restituciones. Este atributo es de sólo lectura.

Tipo de índice
Tipo de índice. Este atributo es de sólo lectura.

inhibir obtención
Si se permiten operaciones get. El valor inicial depende de la definición de cola. Este atributo sólo es válido para un alias o cola local.

inhibir colocación
Si se permiten las operaciones de colocación. El valor inicial depende de la definición de cola.

Nombre cola iniciación
Nombre de la cola de inicio. Este atributo es de sólo lectura.

Profundidad máxima
Número máximo de mensajes permitidos en la cola. Este atributo es de sólo lectura.

longitud máxima del mensaje
Longitud máxima para cualquier mensaje de esta cola, que puede ser menor que el máximo para cualquier cola gestionada por el gestor de colas asociado. Este atributo es de sólo lectura.

Secuencia de entrega de mensajes
Indica si la prioridad del mensaje es relevante. Este atributo es de sólo lectura.

siguiente cola distribuida
El siguiente objeto de esta clase, en ningún orden concreto, que tenga la misma **referencia de lista de distribución** que este objeto. El valor inicial es cero.

Si se suprime un objeto de una cadena, el objeto anterior y el siguiente objeto se actualizan para que sus enlaces de cola distribuida ya no apunten al objeto suprimido.

clase de mensaje no persistente
Nivel de fiabilidad para los mensajes no persistentes colocados en esta cola. Este atributo es de sólo lectura.

Recuento de entradas abiertas
Número de objetos ImqQueue que están abiertos para entrada. Este atributo es de sólo lectura.

Recuento de salidas abiertas
Número de objetos ImqQueue que están abiertos para salida. Este atributo es de sólo lectura.

cola distribuida anterior
Objeto anterior de esta clase, en ningún orden concreto, que tenga la misma **referencia de lista de distribución** que este objeto. El valor inicial es cero.

Si se suprime un objeto de una cadena, el objeto anterior y el siguiente objeto se actualizan para que sus enlaces de cola distribuida ya no apunten al objeto suprimido.

Nombre de proceso
Nombre de la definición de proceso. Este atributo es de sólo lectura.

Contabilidad de la cola
Nivel de información de contabilidad para colas. Este atributo es de sólo lectura.

nombre del gestor de colas
Nombre del gestor de colas (posiblemente remoto) donde reside la cola. No confunda el gestor de colas nombrado aquí con la **referencia de conexión** ImqObject , que hace referencia al gestor de colas (local) que proporciona una conexión. El valor inicial es nulo.

Supervisión de la cola
Nivel de recopilación de datos de supervisión para la cola. Este atributo es de sólo lectura.

estadísticas de cola
Nivel de datos de estadísticas para la cola. Este atributo es de sólo lectura.

Tipo de cola
El tipo de cola. Este atributo es de sólo lectura.

Nombre de gestor de colas remoto
Nombre del gestor de colas remoto. Este atributo es de sólo lectura.

Nombre de cola remota
Nombre de la cola remota tal como se conoce en el gestor de colas remoto. Este atributo es de sólo lectura.

Nombre del gestor de colas resuelto
Nombre de gestor de colas resuelto. Este atributo es de sólo lectura.

Nombre de cola resuelto
Nombre de cola resuelto. Este atributo es de sólo lectura.

Intervalo de retención
Intervalo de retención de cola. Este atributo es de sólo lectura.

Ámbito
Ámbito de la definición de cola. Este atributo es de sólo lectura.

intervalo de servicio
Intervalo de servicio. Este atributo es de sólo lectura.

suceso de intervalo de servicio
Atributo de control para sucesos de intervalo de servicio. Este atributo es de sólo lectura.

Posibilidad de compartición
Si la cola puede compartirse. Este atributo es de sólo lectura.

clase de almacenamiento
Clase de almacenamiento. Este atributo es de sólo lectura.

Nombre de cola de transmisión
Nombre de la cola de transmisión. Este atributo es de sólo lectura.

Activar control
Control de desencadenante. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

Datos desencadenantes
Datos del desencadenante. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

Profundidad de desencadenante
Profundidad del desencadenante. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

Prioridad de mensajes desencadenantes
Umbral de prioridad del mensaje para activaciones. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

tipo de activador
Tipo de desencadenante. El valor inicial depende de la definición de cola. Este atributo sólo es válido para una cola local.

Utilización
Uso. Este atributo es de sólo lectura.

Constructores

ImqQueue( );
El constructor predeterminado.

ImqQueue(const ImqQueue & ***cola*** **);**
El constructor de copia. El **estado abierto** de ImqObject será FALSE.

ImqQueue(const char * ***nombre*** **);**
Establece el **nombre** ImqObject .

Métodos de objeto (public)

void operator = (const ImqQueue & ***cola*** **);**
Realiza un cierre si es necesario y, a continuación, copia los datos de instancia de la cola de . El **estado abierto** de ImqObject será FALSE.

ImqBoolean backoutRequeueNombre(ImqString & ***nombre*** **);**
Proporciona una copia del **nombre de reposición en cola de restitución**. Devuelve TRUE si es satisfactorio.

ImqString backoutRequeueNombre();
Devuelve el **nombre de reposición en cola de restitución** sin ninguna indicación de posibles errores.

ImqBoolean backoutThreshold(MQLONG & ***umbral*** **);**
Proporciona una copia del **umbral de restitución**. Devuelve TRUE si es satisfactorio.

MQLONG backoutThreshold();
Devuelve el valor de **umbral de restitución** sin ninguna indicación de posibles errores.

ImqBoolean baseQueueNombre(ImqString & ***nombre*** **);**
Proporciona una copia del **nombre de cola base**. Devuelve TRUE si es satisfactorio.

ImqString baseQueueNombre();
Devuelve el **nombre de cola base** sin ninguna indicación de posibles errores.

ImqBoolean clusterName(ImqString & ***nombre*** **);**
Proporciona una copia del **nombre de clúster**. Devuelve TRUE si es satisfactorio.

ImqString clusterName( );
Devuelve el **nombre de clúster** sin ninguna indicación de posibles errores.

ImqBoolean clusterNamelistName (ImqString & ***nombre*** **);**
Proporciona una copia del **nombre de lista de nombres de clúster**. Devuelve TRUE si es satisfactorio.

ImqString clusterNamelistNombre ();
Devuelve el **nombre de lista de nombres de clúster** sin ninguna indicación de errores.

ImqBoolean clusterWorkLoadPriority (MQLONG & priority);
Proporciona una copia del valor de prioridad de carga de trabajo de clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkLoadPriority ();
Devuelve el valor de prioridad de carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkLoadRank (MQLONG & rango);
Proporciona una copia del valor de rango de carga de trabajo de clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkLoadRank ();
Devuelve el valor de rango de carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);
Proporciona una copia del valor de cola de uso de carga de trabajo de clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkLoadUseQ ();
Devuelve el valor de cola de uso de carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean creationDate(ImqString &*fecha*);
Proporciona una copia de la **fecha de creación**. Devuelve TRUE si es satisfactorio.

ImqString creationDate( );
Devuelve la **fecha de creación** sin ninguna indicación de posibles errores.

ImqBoolean creationTime(ImqString &*hora*);
Proporciona una copia de la **hora de creación**. Devuelve TRUE si es satisfactorio.

ImqString creationTime( );
Devuelve la **hora de creación** sin ninguna indicación de posibles errores.

ImqBoolean currentDepth(MQLONG &*profundidad*);
Proporciona una copia de la **profundidad actual**. Devuelve TRUE si es satisfactorio.

MQLONG currentDepth();
Devuelve la **profundidad actual** sin ninguna indicación de posibles errores.

ImqBoolean defaultInputOpenOption(MQLONG &*opción*);
Proporciona una copia de la **opción de apertura de entrada predeterminada**. Devuelve TRUE si es satisfactorio.

MQLONG defaultInputOpenOption();
Devuelve la **opción de apertura de entrada predeterminada** sin ninguna indicación de posibles errores.

ImqBoolean defaultPersistence(MQLONG &*persistencia*);
Proporciona una copia de la **persistencia predeterminada**. Devuelve TRUE si es satisfactorio.

MQLONG defaultPersistence();
Devuelve la **persistencia predeterminada** sin ninguna indicación de posibles errores.

ImqBoolean defaultPriority(MQLONG &*prioridad*);
Proporciona una copia de la **prioridad predeterminada**. Devuelve TRUE si es satisfactorio.

MQLONG defaultPriority();
Devuelve la **prioridad predeterminada** sin ninguna indicación de posibles errores.

ImqBoolean defaultBind(MQLONG &*bind*);
Proporciona una copia del **enlace predeterminado**. Devuelve TRUE si es satisfactorio.

MQLONG defaultBind();
Devuelve el **enlace predeterminado** sin ninguna indicación de posibles errores.

ImqBoolean definitionType(MQLONG &*tipo*);
Proporciona una copia del **tipo de definición**. Devuelve TRUE si es satisfactorio.

MQLONG definitionType();
Devuelve el **tipo de definición** sin ninguna indicación de posibles errores.

ImqBoolean depthHighEvent(MQLONG &*suceso*);
Proporciona una copia del estado de habilitación del **suceso de profundidad alta**. Devuelve TRUE si es satisfactorio.

MQLONG depthHighSuceso();
Devuelve el estado de habilitación del **suceso de profundidad alta** sin ninguna indicación de posibles errores.

ImqBoolean depthHighLimit(MQLONG &*límite*);
Proporciona una copia del **límite alto de profundidad**. Devuelve TRUE si es satisfactorio.

MQLONG depthHighLímite();
Devuelve el valor de **límite superior de profundidad** sin ninguna indicación de posibles errores.

ImqBoolean depthLowEvent(MQLONG &*suceso*);
Proporciona una copia del estado de habilitación del **suceso de profundidad baja**. Devuelve TRUE si es satisfactorio.

MQLONG depthLowSuceso();
Devuelve el estado de habilitación del **suceso de profundidad baja** sin ninguna indicación de posibles errores.

ImqBoolean depthLowLimit(MQLONG & *límite*);
Proporciona una copia del **límite bajo de profundidad**. Devuelve TRUE si es satisfactorio.

MQLONG depthLowLímite();
Devuelve el valor de **límite inferior de profundidad** sin ninguna indicación de posibles errores.

ImqBoolean depthMaximumEvent(MQLONG & *suceso*);
Proporciona una copia del estado de habilitación del **suceso de profundidad máxima**. Devuelve TRUE si es satisfactorio.

MQLONG depthMaximumSuceso();
Devuelve el estado de habilitación del **suceso máximo de profundidad** sin ninguna indicación de posibles errores.

ImqDistributionList * distributionListReference() const ;
Devuelve la **referencia de lista de distribución**.

void setDistributionListReference(ImqDistributionList & *lista*);
Establece la **referencia de lista de distribución**.

void setDistributionListReference(ImqDistributionList * *list* = 0);
Establece la **referencia de lista de distribución**.

ImqBoolean distributionLists(MQLONG & *soporte*);
Proporciona una copia del valor de **listas de distribución** . Devuelve TRUE si es satisfactorio.

MQLONG distributionLists();
Devuelve el valor de **listas de distribución** sin ninguna indicación de posibles errores.

ImqBoolean setDistributionLista(const MQLONG *soporte*);
Establece el valor de **listas de distribución** . Devuelve TRUE si es satisfactorio.

ImqString dynamicQueueNombre() const ;
Devuelve una copia del **nombre de cola dinámica**.

ImqBoolean setDynamicQueueName(const char * *nombre*);
Establece el **nombre de cola dinámica**. El **nombre de cola dinámica** solo se puede establecer mientras el **estado abierto** de ImqObject es FALSE. Devuelve TRUE si es satisfactorio.

ImqBoolean get(ImqMessage & *msg*, ImqGetMessageOptions & *opciones*);
Recupera un mensaje de la cola, utilizando las *opciones*especificadas. Invoca el método ImqObject **openFor** si es necesario para asegurarse de que las **opciones de apertura** ImqObject incluyen uno de los valores MQOO_INPUT_ * o el valor MQOO_BROWSE, en función de las *opciones*. Si el objeto *msg* tiene un **almacenamiento intermedio automático** ImqCache , el almacenamiento intermedio crece para acomodar cualquier mensaje recuperado. El método **clearMessage** se invoca en el objeto *msg* antes de la recuperación.

Este método devuelve TRUE si es satisfactorio.

Nota: El resultado de la invocación del método es FALSE si el **código de razón** ImqObject es MQRC_TRUNCATED_MSG_FAILED, aunque este **código de razón** se clasifique como aviso. Si se acepta un mensaje truncado, la **longitud de mensaje** ImqCache refleja la longitud truncada. En cualquier caso, la **longitud total de mensaje** ImqMessage indica el número de bytes que estaban disponibles.

ImqBoolean get(ImqMessage & *msg*);
En cuanto al método anterior, excepto que se utilizan las opciones de obtención de mensaje predeterminadas.

ImqBoolean get(ImqMessage & *msg*, ImqGetMessageOptions & *opciones*, const size_t *tamaño-búfer*);
En cuanto a los dos métodos anteriores, excepto que se indica una alteración temporal de *tamaño-búfer* . Si el objeto *msg* utiliza un almacenamiento intermedio automático de ImqCache , se invoca el método **resizeBuffer** en el objeto *msg* antes de la recuperación de mensajes, y el almacenamiento intermedio no crece más para acomodar ningún mensaje más grande.

ImqBoolean get(ImqMessage & *msg*, const size_t *tamaño-búfer*);
En cuanto al método anterior, excepto que se utilizan las opciones de obtención de mensaje predeterminadas.

ImqBoolean hardenGetBackout(MQLONG & *reforzar*);
Proporciona una copia del valor **harden get backout** . Devuelve TRUE si es satisfactorio.

MQLONG hardenGetBackout();
Devuelve el valor **harden get backout** sin ninguna indicación de posibles errores.

ImqBoolean indexType(MQLONG & *tipo*);
Proporciona una copia del **tipo de índice**. Devuelve TRUE si es satisfactorio.

MQLONG indexType();
Devuelve el **tipo de índice** sin ninguna indicación de posibles errores.

ImqBoolean inhibitGet(MQLONG & *inhibir*);
Proporciona una copia del valor **Impedir obtención** . Devuelve TRUE si es satisfactorio.

MQLONG inhibitGet();
Devuelve el valor **Impedir obtención** sin ninguna indicación de posibles errores.

ImqBoolean setInhibitGet(const MQLONG *inhibir*);
Establece el valor **Impedir obtención** . Devuelve TRUE si es satisfactorio.

ImqBoolean inhibitPut(MQLONG & *inhibir*);
Proporciona una copia del valor **Impedir colocación** . Devuelve TRUE si es satisfactorio.

MQLONG inhibitPut();
Devuelve el valor **Impedir colocación** sin ninguna indicación de posibles errores.

ImqBoolean setInhibitPut(const MQLONG *inhibir*);
Establece el valor **Impedir colocación** . Devuelve TRUE si es satisfactorio.

ImqBoolean initiationQueueNombre(ImqString & *nombre*);
Proporciona una copia del **nombre de cola de inicio**. Devuelve TRUE si es satisfactorio.

ImqString initiationQueueNombre();
Devuelve el **nombre de cola de inicio** sin ninguna indicación de posibles errores.

ImqBoolean maximumDepth(MQLONG & *profundidad*);
Proporciona una copia de la **profundidad máxima**. Devuelve TRUE si es satisfactorio.

MQLONG maximumDepth();
Devuelve la **profundidad máxima** sin ninguna indicación de posibles errores.

ImqBoolean maximumMessageLength(MQLONG & *longitud*);
Proporciona una copia de la **longitud máxima de mensaje**. Devuelve TRUE si es satisfactorio.

MQLONG maximumMessageLongitud();
Devuelve la **longitud máxima de mensaje** sin ninguna indicación de posibles errores.

ImqBoolean messageDeliverySecuencia(MQLONG & *secuencia*);
Proporciona una copia de la **secuencia de entrega de mensajes**. Devuelve TRUE si es satisfactorio.

MQLONG messageDeliverySecuencia();
Devuelve el valor de **secuencia de entrega de mensajes** sin ninguna indicación de posibles errores.

ImqQueue * nextDistributedCola() const ;
Devuelve la **siguiente cola distribuida**.

ImqBoolean nonPersistentMessageClass (MQLONG & monq);
Proporciona una copia del valor de clase de mensaje no persistente. Devuelve TRUE si es satisfactorio.

MQLONG nonPersistentMessageClass ();
Devuelve el valor de clase de mensaje no persistente sin ninguna indicación de posibles errores.

ImqBoolean openInputCount(MQLONG & *recuento*);
Proporciona una copia del **recuento de entradas abiertas**. Devuelve TRUE si es satisfactorio.

MQLONG openInputCount();
Devuelve el **recuento de entradas abiertas** sin ninguna indicación de posibles errores.

ImqBoolean openOutputCount(MQLONG & ***recuento*** **);**
Proporciona una copia del **recuento de salidas abiertas**. Devuelve TRUE si es satisfactorio.

MQLONG openOutputRecuento();
Devuelve el **recuento de salidas abiertas** sin ninguna indicación de posibles errores.

ImqQueue * previousDistributedQueue() const ;
Devuelve la **cola distribuida anterior**.

ImqBoolean processName(ImqString & ***nombre*** **);**
Proporciona una copia del **nombre de proceso**. Devuelve TRUE si es satisfactorio.

ImqString processName( );
Devuelve el **nombre de proceso** sin ninguna indicación de posibles errores.

ImqBoolean put(ImqMessage & ***msg*** **);**
Coloca un mensaje en la cola, utilizando las opciones de colocación de mensaje predeterminadas. Utiliza el método ImqObject **openFor** si es necesario para asegurarse de que las **opciones de apertura** de ImqObject incluyen MQOO_OUTPUT.

Este método devuelve TRUE si es satisfactorio.

ImqBoolean put(ImqMessage & ***msg*****, ImqPutMessageOptions &** ***pmo*** **);**
Coloca un mensaje en la cola, utilizando el *pmo*especificado. Utiliza el método ImqObject **openFor** según sea necesario para asegurarse de que las opciones de apertura de ImqObject incluyen MQOO_OUTPUT y (si las opciones de *pmo* incluyen alguno de los valores de MQPMO_PASS_IDENTITY_CONTEXT, MQPMO_PASS_ALL_CONTEXT, MQPMO_SET_IDENTITY_CONTEXT o MQPMO_SET_ALL_CONTEXT) correspondientes a MQOO_ * _CONTEXT.

Este método devuelve TRUE si es satisfactorio.

Nota: Si *pmo* incluye una **referencia de contexto**, el objeto referenciado se abre, si es necesario, para proporcionar un contexto.

ImqBoolean queueAccounting (MQLONG & acctq);
Proporciona una copia del valor de contabilidad de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueAccounting ();
Devuelve el valor de contabilidad de cola sin ninguna indicación de posibles errores.

ImqString queueManagerNombre() const ;
Devuelve el **nombre de gestor de colas**.

ImqBoolean setQueueManagerName(const char * ***nombre*** **);**
Establece el **nombre de gestor de colas**. El **nombre de gestor de colas** sólo se puede establecer mientras el **estado abierto** de ImqObject es FALSE. Este método devuelve TRUE si es satisfactorio.

ImqBoolean queueMonitoring (MQLONG & monq);
Proporciona una copia del valor de supervisión de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueMonitoring ();
Devuelve el valor de supervisión de cola sin ninguna indicación de posibles errores.

ImqBoolean queueStatistics (MQLONG & statq);
Proporciona una copia del valor de estadísticas de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueStatistics ();
Devuelve el valor de estadísticas de cola sin ninguna indicación de posibles errores.

ImqBoolean queueType(MQLONG & ***tipo*** **);**
Proporciona una copia del valor de **tipo de cola** . Devuelve TRUE si es satisfactorio.

MQLONG queueType();
Devuelve el **tipo de cola** sin ninguna indicación de posibles errores.

ImqBoolean remoteQueueManagerName(ImqString & ***nombre*** **);**
Proporciona una copia del **nombre de gestor de colas remoto**. Devuelve TRUE si es satisfactorio.

ImqString remoteQueueManagerName( );
Devuelve el **nombre de gestor de colas remoto** sin ninguna indicación de posibles errores.

ImqBoolean remoteQueueNombre(ImqString &*nombre*);
Proporciona una copia del **nombre de cola remota**. Devuelve TRUE si es satisfactorio.

ImqString remoteQueueNombre();
Devuelve el **nombre de cola remota** sin ninguna indicación de posibles errores.

ImqBoolean resolvedQueueManagerName(ImqString &*nombre*);
Proporciona una copia del **nombre de gestor de colas resuelto**. Devuelve TRUE si es satisfactorio.

Nota: Este método falla a menos que MQOO_RESOLVE_NAMES esté entre las **opciones de apertura** ImqObject .

ImqString resolvedQueueManagerName( ) ;
Devuelve el **nombre de gestor de colas resuelto**, sin ninguna indicación de posibles errores.

ImqBoolean resolvedQueueNombre (ImqString &*nombre*);
Proporciona una copia del **nombre de cola resuelto**. Devuelve TRUE si es satisfactorio.

Nota: Este método falla a menos que MQOO_RESOLVE_NAMES esté entre las **opciones de apertura** ImqObject .

ImqString resolvedQueueNombre ();
Devuelve el **nombre de cola resuelto**, sin ninguna indicación de posibles errores.

ImqBoolean retentionInterval(MQLONG &*intervalo*);
Proporciona una copia del **intervalo de retención**. Devuelve TRUE si es satisfactorio.

MQLONG retentionInterval();
Devuelve el **intervalo de retención** sin ninguna indicación de posibles errores.

ImqBoolean scope(MQLONG &*ámbito*);
Proporciona una copia del **ámbito**. Devuelve TRUE si es satisfactorio.

MQLONG Ámbito();
Devuelve el **ámbito** sin ninguna indicación de posibles errores.

ImqBoolean serviceInterval(MQLONG &*intervalo*);
Proporciona una copia del **intervalo de servicio**. Devuelve TRUE si es satisfactorio.

MQLONG serviceInterval();
Devuelve el **intervalo de servicio** sin ninguna indicación de posibles errores.

ImqBoolean serviceIntervalEvent(MQLONG &*suceso*);
Proporciona una copia del estado de habilitación del **suceso de intervalo de servicio**. Devuelve TRUE si es satisfactorio.

MQLONG serviceIntervalSuceso();
Devuelve el estado de habilitación del **suceso de intervalo de servicio** sin ninguna indicación de posibles errores.

ImqBoolean compartibilidad(MQLONG &*compartibilidad*);
Proporciona una copia del valor **shareability** . Devuelve TRUE si es satisfactorio.

MQLONG compartibilidad();
Devuelve el valor de **shareability** sin ninguna indicación de posibles errores.

ImqBoolean storageClass(ImqString &*clase*);
Proporciona una copia de la clase de almacenamiento . Devuelve TRUE si es satisfactorio.

ImqString storageClass( );
Devuelve la **clase de almacenamiento** sin ninguna indicación de posibles errores.

ImqBoolean transmissionQueueNombre(ImqString &*nombre*);
Proporciona una copia del **nombre de cola de transmisión**. Devuelve TRUE si es satisfactorio.

ImqString transmissionQueueNombre();
Devuelve el **nombre de cola de transmisión** sin ninguna indicación de posibles errores.

ImqBoolean triggerControl(MQLONG &*control*);
Proporciona una copia del valor de **control de desencadenante** . Devuelve TRUE si es satisfactorio.

MQLONG triggerControl();
Devuelve el valor de **control de desencadenante** sin ninguna indicación de posibles errores.

ImqBoolean setTriggerControl(const MQLONG *control*);
Establece el valor de **control de desencadenante** . Devuelve TRUE si es satisfactorio.

ImqBoolean triggerData(ImqString & *datos*);
Proporciona una copia de los **datos desencadenantes**. Devuelve TRUE si es satisfactorio.

ImqString triggerData( );
Devuelve una copia de los **datos de desencadenante** sin ninguna indicación de posibles errores.

ImqBoolean setTriggerData(const char * *datos*);
Establece los **datos de desencadenante**. Devuelve TRUE si es satisfactorio.

ImqBoolean triggerDepth(MQLONG & *profundidad*);
Proporciona una copia de la **profundidad de desencadenante**. Devuelve TRUE si es satisfactorio.

MQLONG triggerDepth();
Devuelve la **profundidad de desencadenante** sin ninguna indicación de posibles errores.

ImqBoolean setTriggerDepth(const MQLONG *profundidad*);
Establece la **profundidad de desencadenante**. Devuelve TRUE si es satisfactorio.

ImqBoolean triggerMessagePriority(MQLONG & *prioridad*);
Proporciona una copia de la **prioridad de mensaje desencadenante**. Devuelve TRUE si es satisfactorio.

MQLONG triggerMessagePriority();
Devuelve la **prioridad de mensaje desencadenante** sin ninguna indicación de posibles errores.

ImqBoolean setTriggerMessagePriority(const MQLONG *prioridad*);
Establece la **prioridad de mensaje desencadenante**. Devuelve TRUE si es satisfactorio.

ImqBoolean triggerType(MQLONG & *tipo*);
Proporciona una copia del **tipo de desencadenante**. Devuelve TRUE si es satisfactorio.

MQLONG triggerType();
Devuelve el **tipo de desencadenante** sin ninguna indicación de posibles errores.

ImqBoolean setTriggerTipo(const MQLONG *tipo*);
Establece el **tipo de desencadenante**. Devuelve TRUE si es satisfactorio.

ImqBoolean uso(MQLONG & *uso*);
Proporciona una copia del valor de **uso** . Devuelve TRUE si es satisfactorio.

MQLONG uso();
Devuelve el valor de **uso** sin ninguna indicación de posibles errores.

Métodos de objeto (protegidos)

void setNextDistributedQueue(ImqQueue * *queue* = 0);
Establece la **siguiente cola distribuida**.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de colas distribuidas.

void setPreviousDistributedQueue(ImqQueue * *queue* = 0);
Establece la **cola distribuida anterior**.

Atención: Utilice esta función sólo si está seguro de que no romperá la lista de colas distribuidas.

códigos de razón

- MQRC_ATTRIBUTE_LOCKED
- MQRC_CONTEXT_OBJECT_NOT_VALID
- MQRC_CONTEXT_OPEN_ERROR
- MQRC_CURSOR_NOT_VALID
- MQRC_NO_BUFFER

- MQRC_REOPEN_EXCL_INPUT_ERROR
- MQRC_REOPEN_INQUIRE_ERROR
- MQRC_REOPEN_TEMPORARY_Q_ERROR
- (códigos de razón de MQGET)
- (códigos de razón de MQPUT)

Clase C++ del gestor de ImqQueue

Esta clase encapsula un gestor de colas (un objeto WebSphere MQ de tipo MQOT_Q_MGR).

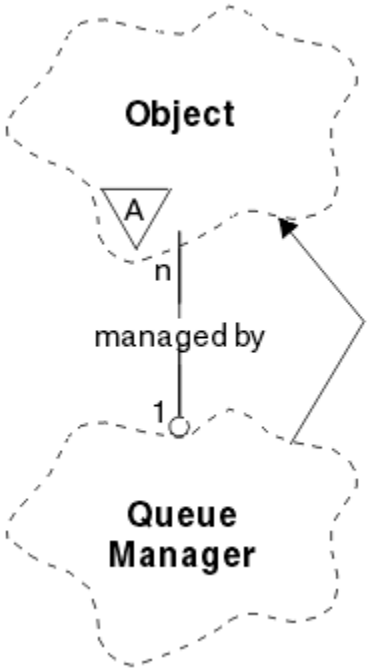


Figura 65. Clase de gestor ImqQueue

Esta clase está relacionada con las llamadas MQI listadas en “Referencia cruzada de ImqQueueManager” en la página 1327. No todos los métodos listados son aplicables a todas las plataformas; consulte ALTER QMGR para obtener más detalles.

Atributos de clase

comportamiento
Controla el comportamiento de la conexión y desconexión implícitas.

IMQ_EXPL_DISC_BACKOUT (0L)
Una llamada explícita al método **disconnect** implica restitución. Este atributo se excluye mutuamente con IMQ_EXPL_DISC_COMMIT.

IMQ_EXPL_DISC_COMMIT (1L)
Una llamada explícita al método **disconnect** implica commit (el valor predeterminado). Este atributo se excluye mutuamente con IMQ_EXPL_DISC_BACKOUT.

IMQ_IMPL_CONN (2L)
Se permite la conexión implícita (el valor predeterminado).

IMQ_IMPL_DISC_BACKOUT (0L)
Una llamada implícita al método **disconnect** , que puede producirse durante la destrucción de objetos, implica la restitución. Este atributo se excluye mutuamente con IMQ_IMPL_DISC_COMMIT.

IMQ_IMPL_DISC_COMMIT (4L)
Una llamada implícita al método **disconnect** , que puede producirse durante la destrucción de objetos, implica la confirmación (el valor predeterminado). Este atributo se excluye mutuamente con IMQ_IMPL_DISC_BACKOUT.

En WebSphere MQ V7.0 y superior, las aplicaciones C++ que utilizan una conexión implícita, deben especificar IMQ_IMPL_CONN junto con cualquier otra opción proporcionada en el método `setBehavior()` en un objeto de clase `ImqQueueManager`. Si la aplicación no utiliza el método `setBehavior()` para establecer explícitamente las opciones de comportamiento, por ejemplo,

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

este cambio no le afecta puesto que MQ_IMPL_CONN está habilitado de forma predeterminada.

Si la aplicación establece explícitamente las opciones de comportamiento, por ejemplo,

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

debe incluir IMQ_IMPL_CONN en el método `setBehavior()` de la forma siguiente, para permitir que la aplicación complete una conexión implícita:

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_CONN | IMQ_IMPL_DISC_COMMIT)
```

Atributos de objetos

alteración temporal de conexiones de contabilidad
Permite a las aplicaciones alterar temporalmente el valor de los valores values.This es de sólo lectura.

Intervalo contable
Tiempo antes de que se escriban los registros de contabilidad intermedios (en segundos). Este atributo es de sólo lectura.

registro de actividad
Controla la generación de los informes de actividades. Este atributo es de sólo lectura.

Adoptar nueva comprobación MCA
Los elementos comprobados para determinar si un MCA debe adoptarse cuando se detecta un nuevo canal de entrada que tiene el mismo nombre que un MCA que ya está activo. Este atributo es de sólo lectura.

Adoptar nuevo tipo MCA
Si una instancia huérfana de un MCA de un tipo de canal determinado debe reiniciarse automáticamente cuando se detecte una nueva solicitud de canal de entrada que coincida con los nuevos parámetros de comprobación de adopción de mca. Este atributo es de sólo lectura.

Tipo de autenticación
Indica el tipo de autenticación que se está realizando.

suceso de autorización
Controla los sucesos de autorización. Este atributo es de sólo lectura.

Opciones de inicio
Opciones que se aplican al método **begin** . El valor inicial es MQBO_NONE.

suceso de puente
Indica si se generan sucesos de puente IMS . Este atributo es de sólo lectura.

Definición automática de canal
Valor de definición automática de canal. Este atributo es de sólo lectura.

suceso de definición automática de canal
Valor de suceso de definición automática de canal. Este atributo es de sólo lectura.

Salida de definición automática de canal
Nombre de salida de definición automática de canal. Este atributo es de sólo lectura.

suceso de canal
Indica si se generan sucesos de canal. Este atributo es de sólo lectura.

Adaptadores del iniciador de canal
Número de subtareas de adaptador que se deben utilizar para procesar llamadas de WebSphere MQ . Este atributo es de sólo lectura.

Control del iniciador de canal
Indica si el iniciador de canal debe iniciarse automáticamente cuando se inicia el gestor de colas. Este atributo es de sólo lectura.

Asignadores de tareas del iniciador de canal
Número de asignadores a utilizar para el iniciador de canal. Este atributo es de sólo lectura.

inicio automático de rastreo de iniciador de canal
Indica si el rastreo de iniciador de canal debe iniciarse automáticamente o no. Este atributo es de sólo lectura.

Tamaño de tabla de rastreo del iniciador de canal
El tamaño del espacio de datos de rastreo del iniciador de canal (en MB). Este atributo es de sólo lectura.

Supervisión de canal
Controla la recopilación de los datos de supervisión para los canales. Este atributo es de sólo lectura.

referencia de canal
Una referencia a una definición de canal para utilizarla durante la conexión de cliente. Mientras está conectado, este atributo se puede establecer en nulo, pero no se puede cambiar a ningún otro valor. El valor inicial es nulo.

Estadísticas del canal
Controla la recopilación de los datos estadísticos para los canales. Este atributo es de sólo lectura.

juego de caracteres
Identificador de juego de caracteres codificado (CCSID). Este atributo es de sólo lectura.

Supervisión de clúster emisor
Controla la recopilación de datos de supervisión en línea para canales emisores de clúster definidos automáticamente. Este atributo es de sólo lectura.

Estadística de clúster emisor
Controla la recopilación de datos estadísticos para los canales emisores de clúster definidos automáticamente. Este atributo es de sólo lectura.

Datos de carga de trabajo de clúster
Datos de salida de carga de trabajo de clúster. Este atributo es de sólo lectura.

salida de carga de trabajo de clúster
Nombre de salida de carga de trabajo de clúster. Este atributo es de sólo lectura.

Longitud de la carga de trabajo de clúster
Longitud de carga de trabajo de clúster. Este atributo es de sólo lectura.

mru de carga de trabajo de clúster
Valor de canales utilizados más recientemente de carga de trabajo de clúster. Este atributo es de sólo lectura.

Cola de uso de carga de trabajo de clúster
La carga de trabajo de clúster utiliza el valor de cola. Este atributo es de sólo lectura.

suceso de mandato
Indica si se generan sucesos de mandato. Este atributo es de sólo lectura.

Nombre de cola de entrada de mandatos
Nombre de cola de entrada de mandatos del sistema. Este atributo es de sólo lectura.

Nivel de mandato
Nivel de mandatos soportado por el gestor de colas. Este atributo es de sólo lectura.

Control del servidor de mandatos
Indica si el servidor de mandatos se debe iniciar automáticamente cuando se inicia el gestor de colas. Este atributo es de sólo lectura.

Opciones de conexión
Opciones que se aplican al método **connect** . El valor inicial es MQCNO_NONE. Los siguientes valores adicionales pueden ser posibles, dependiendo de la plataforma:

- 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 de conexión
Identificador exclusivo que permite a MQ identificar de forma fiable una aplicación.

estado de conexión
TRUE cuando se conecta al gestor de colas. Este atributo es de sólo lectura.

etiqueta de conexión
Una etiqueta que se va a asociar con una conexión. Este atributo sólo se puede establecer cuando no está conectado. El valor inicial es nulo.

hardware de cifrado
Detalles de configuración para hardware criptográfico. Para conexiones de cliente MQI de MQ .

nombre de cola de mensajes no entregados
Nombre de la cola de mensajes no entregados. Este atributo es de sólo lectura.

nombre de cola de transmisión predeterminada
Nombre de cola de transmisión predeterminado. Este atributo es de sólo lectura.

listas de distribución
Capacidad del gestor de colas para dar soporte a listas de distribución.

grupo dns
El nombre del grupo al que se debe unir el escucha TCP que maneja las transmisiones de entrada para el grupo de compartición de colas cuando se utiliza el soporte de Workload Manager Dynamic Domain Name Services. Este atributo es de sólo lectura.

dns wlm
Indica si el escucha TCP que maneja las transmisiones de entrada para el grupo de compartición de colas debe registrarse con el Gestor de carga de trabajo para Dynamic Domain Name Services. Este atributo es de sólo lectura.

primer registro de autenticación
El primero de uno o más objetos de la clase ImqAuthenticationRecord, en ningún orden concreto, en el que la conexión de registro ImqAuthenticationhace referencia a este objeto. Para conexiones de cliente MQI de MQ .

primer objeto gestionado
El primero de uno o más objetos de la clase ImqObject, en ningún orden concreto, en el que la **referencia de conexión** ImqObject se dirige a este objeto. El valor inicial es cero.

suceso de inhibición
Controla los sucesos de inhibición. Este atributo es de sólo lectura.

Versión de dirección IP
Qué protocolo IP (IPv4 o IPv6) utilizar para una conexión de canal. Este atributo es de sólo lectura.

repositorio de claves
Ubicación del archivo de base de datos de claves en el que se almacenan las claves y los certificados. Para conexiones de cliente MQI de WebSphere MQ .

recuento de restablecimiento de clave
Número de bytes no cifrados enviados y recibidos en una conversación SSL antes de que se renegocie la clave secreta. Este atributo sólo se aplica a las conexiones de cliente que utilizan MQCONNX. Véase también recuento de restablecimiento de clave ssl.

Temporizador de escucha
El intervalo de tiempo (en segundos) entre los intentos de WebSphere MQ de reiniciar el escucha si se ha producido una anomalía de APPC o TCP/IP. Este atributo es de sólo lectura.

suceso local
Controla los sucesos locales. Este atributo es de sólo lectura.

LoggerEvent
Controla si se generan sucesos de registro de recuperación. Este atributo es de sólo lectura.

Nombre de grupo LU
El nombre de LU genérico que debe utilizar el escucha de LU 6.2 que maneja las transmisiones de entrada para el grupo de compartición de colas. Este atributo es de sólo lectura.

Nombre de LU
El nombre de la LU que se va a utilizar para las transmisiones de LU de salida 6.2 . Este atributo es de sólo lectura.

Sufijo de brazo lu62
El sufijo de SYS1.PARMLIB APPCPMxx, que nomina el LUADD para este iniciador de canal. Este atributo es de sólo lectura.

Canales lu62
El número máximo de canales que pueden ser actuales o clientes que se pueden conectar, que utilizan el protocolo de transmisión LU 6.2 . Este atributo es de sólo lectura.

máximo de canales activos
Número máximo de canales que pueden estar activos en cualquier momento. Este atributo es de sólo lectura.

Canales máximos
Número máximo de canales que pueden ser actuales (incluidos los canales de conexión de servidor con clientes conectados). Este atributo es de sólo lectura.

máximo de manejadores
Número máximo de descriptores de contexto. Este atributo es de sólo lectura.

longitud máxima del mensaje
Longitud máxima posible para cualquier mensaje en cualquier cola gestionada por este gestor de colas. Este atributo es de sólo lectura.

Prioridad máxima
Prioridad máxima de mensaje. Este atributo es de sólo lectura.

Máx. mensajes no confirmados
Número máximo de mensajes no confirmados dentro de una unidad o trabajo. Este atributo es de sólo lectura.

Contabilidad de MQI
Controla la recopilación de la información contable para los datos de la Interfaz de Colas de Mensajes (MQI). Este atributo es de sólo lectura.

Estadísticas MQI
Controla la recopilación de la información sobre la supervisión de estadísticas para el gestor de colas. Este atributo es de sólo lectura.

máximo de puerto de salida
El extremo superior del rango de números de puerto que se va a utilizar al enlazar canales de salida. Este atributo es de sólo lectura.

mínimo de puerto de salida
El extremo inferior del rango de números de puerto que se va a utilizar al enlazar canales de salida. Este atributo es de sólo lectura.

contraseña
contraseña asociada con el ID de usuario

suceso de rendimiento
Controla los sucesos de rendimiento. Este atributo es de sólo lectura.

plataforma
Plataforma en la que reside el gestor de colas. Este atributo es de sólo lectura.

Contabilidad de la cola
Controla la recopilación de la información contable para las colas. Este atributo es de sólo lectura.

Supervisión de la cola
Controla la recopilación de los datos de supervisión para las colas. Este atributo es de sólo lectura.

estadísticas de cola
Controla la recopilación de los datos estadísticos para las colas. Este atributo es de sólo lectura.

Tiempo de espera de recepción
Aproximadamente el tiempo que un canal de mensajes TCP/IP esperará a recibir datos, incluidos los latidos, de su socio, antes de volver al estado inactivo. Este atributo es de sólo lectura.

tiempo de espera de recepción mínimo
El tiempo mínimo que un canal TCP/IP esperará para recibir datos, incluidos los latidos, de su asociado, antes de volver al estado inactivo. Este atributo es de sólo lectura.

Tipo de tiempo de espera de recepción
Un calificador aplicado a **tiempo de espera de recepción**. Este atributo es de sólo lectura.

suceso remoto
Controla los sucesos remotos. Este atributo es de sólo lectura.

Nombre de depósito
Nombre de repositorio. Este atributo es de sólo lectura.

Lista nombres repositorio
Nombre de lista de nombres de repositorio. Este atributo es de sólo lectura.

nombre de gestor de colas compartido
Si los MQOPEN de una cola compartida en la que el nombre ObjectQMgres otro gestor de colas del grupo de compartición de colas deben resolverse en una apertura de la cola compartida en el gestor de colas local. Este atributo es de sólo lectura.

suceso ssl
Indica si se generan sucesos SSL. Este atributo es de sólo lectura.

Se necesita SSL FIPS
Indica si sólo se deben utilizar algoritmos certificados por FIPS si la criptografía se ejecuta en el software WebSphere MQ . Este atributo es de sólo lectura.

Cuenta restablecimiento clave SSL
Número de bytes no cifrados enviados y recibidos en una conversación SSL antes de que se renegocie la clave secreta. Este atributo es de sólo lectura.

suceso de inicio-detención
Controla los sucesos de inicio-detención. Este atributo es de sólo lectura.

Intervalo de estadísticas
Frecuencia con la que se graban los datos de supervisión de estadísticas en la cola de supervisión. Este atributo es de sólo lectura.

Disponibilidad de punto de sincronismo
Disponibilidad de participación de punto de sincronismo. Este atributo es de sólo lectura.

Nota: Las unidades de trabajo globales coordinadas por el gestor de colas no están soportadas en la plataforma IBM i .

canales tcp
Número máximo de canales que pueden ser actuales o clientes que se pueden conectar, que utilizan el protocolo de transmisión TCP/IP. Este atributo es de sólo lectura.

Mantener activo de TCP
Indica si se va a utilizar el recurso TCP KEEPALIVE para comprobar que el otro extremo de la conexión sigue estando disponible. Este atributo es de sólo lectura.

Nombre TCP
El nombre del sistema TCP/IP único o predeterminado que se va a utilizar, en función del valor de **tcp stack type**. Este atributo es de sólo lectura.

Tipo de pila TCP
Si el iniciador de canal sólo puede utilizar el espacio de direcciones TCP/IP especificado en **tcp name** o puede enlazarse a cualquier dirección TCP/IP seleccionada. Este atributo es de sólo lectura.

Registro de la ruta de rastreo
Controla el registro de la información de rastreo de ruta. Este atributo es de sólo lectura.

Activar intervalo
Intervalo de desencadenante. Este atributo es de sólo lectura.

ID de usuario
En plataformas UNIX and Linux , el ID de usuario real de la aplicación. En plataformas Windows, el ID de usuario de la aplicación.

Constructores

ImqQueueManager ();
El constructor predeterminado.

ImqQueueManager (const ImqQueueManager & *gestor*);
El constructor de copia. El **estado de conexión** será FALSE.

ImqQueueManager (const char * *nombre*);
Establece el ImqObject **nombre** en *nombre*.

Destructores

Cuando se destruye un objeto del gestor ImqQueue, se desconecta automáticamente.

Métodos de clase (public)

comportamiento MQLONG estático ();
Devuelve el **comportamiento**.

void setBehavior(const MQLONG *behavior* = 0);
Establece el **comportamiento**.

Métodos de objeto (public)

void operator = (const ImqQueueGestor & *mgr*);
Se desconecta si es necesario y copia los datos de instancia de *mgr*. El **estado de conexión** es FALSE.

ImqBoolean accountingConnOverride (MQLONG & statint);
Proporciona una copia del valor de alteración temporal de conexiones de contabilidad. Devuelve TRUE si es satisfactorio.

MQLONG accountingConnOverride ();
Devuelve el valor de alteración temporal de conexiones de contabilidad sin ninguna indicación de posibles errores.

ImqBoolean accountingInterval (MQLONG & statint);
Proporciona una copia del valor de intervalo de contabilidad. Devuelve TRUE si es satisfactorio.

MQLONG accountingInterval ();
Devuelve el valor de intervalo de contabilidad sin ninguna indicación de posibles errores.

ImqBoolean activityRecording (MQLONG & rec);
Proporciona una copia del valor de registro de actividad. Devuelve TRUE si es satisfactorio.

MQLONG activityRecording ();
Devuelve el valor de registro de actividad sin ninguna indicación de posibles errores.

ImqBoolean adoptNewMCACheck (MQLONG & check);
Proporciona una copia del valor de comprobación de adopción de MCA nuevo. Devuelve TRUE si es satisfactorio.

MQLONG adoptNewMCACheck ();
Devuelve el valor de comprobación de adopción de nuevo MCA sin ninguna indicación de posibles errores.

ImqBoolean adoptNewMCAType (MQLONG & type);
Proporciona una copia del nuevo tipo de MCA de adopción. Devuelve TRUE si es satisfactorio.

MQLONG adoptNewMCAType ();
Devuelve el nuevo tipo de MCA de adopción sin ninguna indicación de posibles errores.

QLONG authenticationType () Const.
Devuelve el tipo de autenticación.

void setAuthenticationType (const MQLONG type = MQCSP_AUTH_NONE);
Establece el tipo de autenticación.

ImqBoolean authorityEvent(MQLONG & *suceso*);
Proporciona una copia del estado de habilitación del **suceso de autorización**. Devuelve TRUE si es satisfactorio.

MQLONG authorityEvent();
Devuelve el estado de habilitación del **suceso de autorización** sin ninguna indicación de posibles errores.

ImqBoolean backout ();
Restituye los cambios no confirmados. Devuelve TRUE si es satisfactorio.

ImqBoolean begin ();
Inicia una unidad de trabajo. Las **opciones de inicio** afectan al comportamiento de este método. Devuelve TRUE si es satisfactorio, pero también devuelve TRUE incluso si la llamada MQBEGIN subyacente devuelve MQRC_NO_EXTERNAL_PARTICIPANTES o MQRC_PARTICIPANT_NOT_AVAILABLE (que están asociados con MQCC_WARNING).

MQLONG beginOptions() Const.
Devuelve las **opciones de inicio**.

void setBeginOptions (const MQLONG *opciones* = MQBO_NONE);
Establece las **opciones de inicio**.

ImqBoolean bridgeEvent (MQLONG & event);
Proporciona una copia del valor de suceso de puente. Devuelve TRUE si es satisfactorio.

MQLONG bridgeEvent ();
Devuelve el valor de suceso de puente sin ninguna indicación de posibles errores.

ImqBoolean channelAutoDefinition (MQLONG & *valor*);
Proporciona una copia del valor de **definición automática de canal** . Devuelve TRUE si es satisfactorio.

MQLONG channelAutoDefinition ();
Devuelve el valor de **definición automática de canal** sin ninguna indicación de posibles errores.

ImqBoolean channelAutoDefinitionEvent(MQLONG & *valor*);
Proporciona una copia del valor de **suceso de definición automática de canal** . Devuelve TRUE si es satisfactorio.

MQLONG channelAutoDefinitionEvent();
Devuelve el valor de **suceso de definición automática de canal** sin ninguna indicación de posibles errores.

ImqBoolean channelAutoDefinitionExit(ImqString & *nombre*);
Proporciona una copia del nombre de **salida de definición automática de canal** . Devuelve TRUE si es satisfactorio.

ImqString channelAutoDefinitionExit( );
Devuelve el nombre de **salida de definición automática de canal** sin ninguna indicación de posibles errores.

ImqBoolean channelEvent (MQLONG & event);
Proporciona una copia del valor de suceso de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelEvent();
Devuelve el valor de suceso de canal sin ninguna indicación de posibles errores.

MQLONG channelInitiatorAdapters ();
Devuelve el valor de adaptadores de iniciador de canal sin ninguna indicación de posibles errores.

ImqBoolean channelInitiatorAdapters (MQLONG & adapters);
Proporciona una copia del valor de adaptadores de iniciador de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelInitiatorControl ();
Devuelve el valor de inicio del iniciador de canal sin ninguna indicación de posibles errores.

ImqBoolean channelInitiatorControl (MQLONG & init);
Proporciona una copia del valor de inicio de control del iniciador de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelInitiatorDispatchers ();
Devuelve el valor de los asignadores de iniciador de canal sin ninguna indicación de posibles errores.

ImqBoolean channelInitiatorDispatchers (MQLONG & dispatchers);
Proporciona una copia del valor de los asignadores del iniciador de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelInitiatorTraceAutoInicio ();
Devuelve el valor de inicio automático del rastreo del iniciador de canal sin ninguna indicación de posibles errores.

ImqBoolean channelInitiatorTraceAutoInicio (MQLONG & auto);
Proporciona una copia del valor de inicio automático de rastreo de iniciador de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelInitiatorTraceTableTamaño ();
Devuelve el valor de tamaño de tabla de rastreo del iniciador de canal sin ninguna indicación de posibles errores.

ImqBoolean channelInitiatorTraceTableTamaño (MQLONG & size);
Proporciona una copia del valor de tamaño de tabla de rastreo del iniciador de canal. Devuelve TRUE si es satisfactorio.

ImqBoolean channelMonitoring (MQLONG & monchl);
Proporciona una copia del valor de supervisión de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelMonitoring ();
Devuelve el valor de supervisión de canal sin ninguna indicación de posibles errores.

ImqBoolean channelReference(ImqChannel * &*pchannel*);
Proporciona una copia de la **referencia de canal**. Si la **referencia de canal** no es válida, establece *pchannel* en nulo. Este método devuelve TRUE si es satisfactorio.

ImqChannel * channelReference( );
Devuelve la **referencia de canal** sin ninguna indicación de posibles errores.

ImqBoolean setChannelReference (ImqChannel &*canal*);
Establece la **referencia de canal**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setChannelReference (ImqChannel **canal* = 0);
Establece o restablece la **referencia de canal**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean channelStatistics (MQLONG & statchl);
Proporciona una copia del valor de estadísticas de canal. Devuelve TRUE si es satisfactorio.

MQLONG channelStatistics ();
Devuelve el valor de estadísticas de canal sin ninguna indicación de posibles errores.

ImqBoolean characterSet(MQLONG &*ccsid*);
Proporciona una copia del **juego de caracteres**. Devuelve TRUE si es satisfactorio.

MQLONG characterSet();
Devuelve una copia del **juego de caracteres**, sin ninguna indicación de posibles errores.

MQLONG clientSslKeyResetCount () Const.
Devuelve el valor de recuento de restablecimiento de clave SSL utilizado en las conexiones de cliente.

void setClientSslKeyResetCount(const MQLONG count);
Establece el recuento de restablecimiento de clave SSL utilizado en las conexiones de cliente.

ImqBoolean clusterSenderMonitoring (MQLONG & monacls);
Proporciona una copia del valor predeterminado de supervisión del remitente del clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterSenderMonitoring ();
Devuelve el valor predeterminado de supervisión del remitente del clúster sin ninguna indicación de posibles errores.

ImqBoolean clusterSenderStatistics (MQLONG & statacls);
Proporciona una copia del valor de estadísticas del remitente del clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterSenderStatistics ();
Devuelve el valor de estadísticas de clúster emisor sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkloadDatos (ImqString &*datos*);
Proporciona una copia de los **datos de salida de carga de trabajo de clúster**. Devuelve TRUE si es satisfactorio.

ImqString clusterWorkloadData ();
Devuelve los **datos de salida de carga de trabajo de clúster** sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkloadExit (ImqString &*nombre*);
Proporciona una copia del **nombre de salida de carga de trabajo de clúster**. Devuelve TRUE si es satisfactorio.

ImqString clusterWorkloadExit ();
Devuelve el **nombre de salida de carga de trabajo de clúster** sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkloadLongitud (MQLONG &*longitud*);
Proporciona una copia de la **longitud de carga de trabajo de clúster**. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkloadLongitud ();
Devuelve la **longitud de carga de trabajo de clúster** sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkLoadMRU (MQLONG & mru);
Proporciona una copia del valor de canales utilizados más recientemente de la carga de trabajo del clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkLoadMRU ();
Devuelve el valor de canales utilizados más recientemente de la carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);
Proporciona una copia del valor de cola de uso de carga de trabajo de clúster. Devuelve TRUE si es satisfactorio.

MQLONG clusterWorkLoadUseQ ();
Devuelve el valor de cola de uso de carga de trabajo de clúster sin ninguna indicación de posibles errores.

ImqBoolean commandEvent (MQLONG & event);
Proporciona una copia del valor de suceso de mandato. Devuelve TRUE si es satisfactorio.

MQLONG commandEvent ();
Devuelve el valor de suceso de mandato sin ninguna indicación de posibles errores.

ImqBoolean commandInputQueueName(ImqString & ***nombre*** **);**
Proporciona una copia del **nombre de cola de entrada de mandatos**. Devuelve TRUE si es satisfactorio.

ImqString commandInputQueueName( );
Devuelve el **nombre de cola de entrada de mandatos** sin ninguna indicación de posibles errores.

ImqBoolean commandLevel(MQLONG & ***nivel*** **);**
Proporciona una copia del **nivel de mandatos**. Devuelve TRUE si es satisfactorio.

MQLONG commandLevel();
Devuelve el **nivel de mandatos** sin ninguna indicación de posibles errores.

MQLONG commandServerControl ();
Devuelve el valor de inicio del servidor de mandatos sin ninguna indicación de posibles errores.

ImqBoolean commandServerControl (MQLONG & server);
Proporciona una copia del valor de inicio de control del servidor de mandatos. Devuelve TRUE si es satisfactorio.

ImqBoolean commit ();
Confirma los cambios no confirmados. Devuelve TRUE si es satisfactorio.

ImqBoolean connect ();
Se conecta al gestor de colas con el ImqObject **nombre**especificado, el valor predeterminado es el gestor de colas local. Si desea conectarse a un gestor de colas específico, utilice el método ImqObject **setName** antes de la conexión. Si hay una **referencia de canal**, se utiliza para pasar información sobre la definición de canal a MQCONNX en un MQCD. El ChannelType en MQCD se establece en MQCHT_CLNTCONN. La información de **referencia de canal** , que sólo es significativa para las conexiones de cliente, se ignora para las conexiones de servidor. Las **opciones de conexión** afectan al comportamiento de este método. Este método establece el **estado de conexión** en TRUE si es satisfactorio. Devuelve el nuevo estado de conexión.

Si hay un primer registro de autenticación, la cadena de registros de autenticación se utiliza para autenticar certificados digitales para canales de cliente seguros.

Puede conectar más de un objeto de gestor ImqQueueal mismo gestor de colas. Todos utilizan el mismo descriptor de conexión MQHCONN y comparten la funcionalidad de UOW para la conexión asociada con la hebra. El primer gestor ImqQueueque se conecta obtiene el descriptor de contexto MQHCONN. El último gestor de ImqQueueen desconectarse realiza el MQDISC.

Para un programa multihebra, se recomienda que se utilice un objeto de gestor ImqQueueindependiente para cada hebra.

ImqBinary connectionId () Const.
Devuelve el ID de conexión.

ImqBinary connectionTag () Const.
Devuelve la **etiqueta de conexión**.

ImqBoolean setConnectionTag (const MQBYTE128 *etiqueta* = 0);
Establece la **etiqueta de conexión**. Si la *etiqueta* es cero, borra la **etiqueta de conexión**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setConnectionTag (const ImqBinary & *etiqueta*);
Establece la **etiqueta de conexión**. La **longitud de datos** del *código* debe ser cero (para borrar el **código de conexión**) o MQ_CONN_TAG_LENGTH. Este método devuelve TRUE si es satisfactorio.

MQLONG connectOptions() Const.
Devuelve las **opciones de conexión**.

void setConnectOptions (const MQLONG *opciones* = MQCNO_NONE);
Establece las **opciones de conexión**.

ImqBoolean connectionStatus() Const.
Devuelve el **estado de conexión**.

ImqString cryptographicHardware ( );
Devuelve el **hardware criptográfico**.

ImqBoolean setCryptographicHardware (const char * *hardware* = 0);
Establece el **hardware criptográfico**. Este método devuelve TRUE si es satisfactorio.

ImqBoolean deadLetterQueueName(ImqString & *nombre*);
Proporciona una copia del **nombre de cola de mensajes no entregados**. Devuelve TRUE si es satisfactorio.

ImqString deadLetterQueueName( );
Devuelve una copia del **nombre de cola de mensajes no entregados**, sin ninguna indicación de posibles errores.

ImqBoolean defaultTransmissionQueueName(ImqString & *nombre*);
Proporciona una copia del **nombre de cola de transmisión predeterminado**. Devuelve TRUE si es satisfactorio.

ImqString defaultTransmissionQueueName( );
Devuelve el **nombre de cola de transmisión predeterminado** sin ninguna indicación de posibles errores.

ImqBoolean disconnect ();
Se desconecta del gestor de colas y establece el **estado de conexión** en FALSE. Cierra todos los objetos ImqProcess e ImqQueue asociados con este objeto, y interrumpe su **referencia de conexión** antes de la desconexión. Si hay más de un objeto de gestor ImqQueueconectado al mismo gestor de colas, sólo el último en desconectarse realiza una desconexión física; otros realizan una desconexión lógica. Los cambios no confirmados sólo se confirman en la desconexión física.

Este método devuelve TRUE si es satisfactorio. Si se llama cuando no hay ninguna conexión existente, el código de retorno también es verdadero.

ImqBoolean distributionLists(MQLONG & *soporte*);
Proporciona una copia del valor de **listas de distribución** . Devuelve TRUE si es satisfactorio.

MQLONG distributionLists();
Devuelve el valor de **listas de distribución** sin ninguna indicación de posibles errores.

ImqBoolean dnsGroup (ImqString & group);
Proporciona una copia del nombre de grupo DNS. Devuelve TRUE si es satisfactorio.

ImqString dnsGroup ( );
Devuelve el nombre del grupo DNS sin ninguna indicación de posibles errores.

ImqBoolean dnsWlm (MQLONG & wlm);
Proporciona una copia del valor WLM de DNS. Devuelve TRUE si es satisfactorio.

MQLONG dnsWlm ();
Devuelve el valor WLM de DNS sin ninguna indicación de posibles errores.

ImqAuthenticationRecord * firstAuthenticationRecord () Const.
Devuelve el **primer registro de autenticación**.

void setFirstAuthenticationRecord (const ImqAuthenticationRecord * *air* = 0);
Establece el **primer registro de autenticación**.

ImqObject * firstManagedObject () Const.
Devuelve el **primer objeto gestionado**.

ImqBoolean inhibitEvent(MQLONG & *suceso*);
Proporciona una copia del estado de habilitación del **suceso de inhibición**. Devuelve TRUE si es satisfactorio.

MQLONG inhibitEvent();
Devuelve el estado de habilitación del **suceso de inhibición** sin ninguna indicación de posibles errores.

ImqBoolean ipAddressVersion (MQLONG & version);
Proporciona una copia del valor de versión de dirección IP. Devuelve TRUE si es satisfactorio.

MQLONG ipAddressVersión ();
Devuelve el valor de versión de dirección IP sin ninguna indicación de posibles errores.

ImqBoolean keepAlive (MQLONG & keepalive);
Proporciona una copia del valor de mantener activo. Devuelve TRUE si es satisfactorio.

MQLONG keepAlive ();
Devuelve el valor de mantener activo sin ninguna indicación de posibles errores.

ImqString keyRepository ( );
Devuelve el **repositorio de claves**.

ImqBoolean setKeyRepository (const char * *repositorio* = 0);
Establece el **repositorio de claves**. Devuelve TRUE si es satisfactorio.

ImqBoolean listenerTimer (MQLONG & timer);
Proporciona una copia del valor de temporizador de escucha. Devuelve TRUE si es satisfactorio.

MQLONG listenerTimer ();
Devuelve el valor de temporizador de escucha sin ninguna indicación de posibles errores.

ImqBoolean localEvent(MQLONG & *suceso*);
Proporciona una copia del estado de habilitación del **suceso local**. Devuelve TRUE si es satisfactorio.

MQLONG localEvent();
Devuelve el estado de habilitación del **suceso local** sin ninguna indicación de posibles errores.

ImqBoolean loggerEvent (MQLONG & count);
Proporciona una copia del valor de suceso de registrador. Devuelve TRUE si es satisfactorio.

MQLONG loggerEvent ();
Devuelve el valor de suceso de registrador sin ninguna indicación de posibles errores.

ImqBoolean luGroupNombre (ImqString & name);.
Proporciona una copia del nombre de grupo de LU. Devuelve TRUE si es satisfactorio

ImqString luGroupNombre ();
Devuelve el nombre del grupo de LU sin ninguna indicación de posibles errores.

ImqBoolean lu62ARMSuffix (ImqString & suffix);
Proporciona una copia del sufijo ARM LU62 . Devuelve TRUE si es satisfactorio.

ImqString lu62ARMSuffix ( );
Devuelve el sufijo ARM LU62 sin ninguna indicación de posibles errores

ImqBoolean luName (ImqString & name);
Proporciona una copia del nombre de LU. Devuelve TRUE si es satisfactorio.

ImqString luName ( );
Devuelve el nombre de LU sin ninguna indicación de posibles errores.

ImqBoolean maximumActiveChannels (MQLONG & canales);
Proporciona una copia del valor máximo de canales activos. Devuelve TRUE si es satisfactorio.

MQLONG maximumActiveChannels ();
Devuelve el valor máximo de canales activos sin ninguna indicación de posibles errores.

ImqBoolean maximumCurrentChannels (MQLONG & channels);
Proporciona una copia del valor máximo de canales actuales. Devuelve TRUE si es satisfactorio.

MQLONG maximumCurrentChannels ();
Devuelve el valor máximo de canales actuales sin ninguna indicación de posibles errores.

ImqBoolean maximumHandles(MQLONG & ***número*** **);**
Proporciona una copia de los **descriptores de contexto máximos**. Devuelve TRUE si es satisfactorio.

MQLONG maximumHandles();
Devuelve los **manejadores máximos** sin ninguna indicación de posibles errores.

ImqBoolean maximumLu62Channels (MQLONG & canales);
Proporciona una copia del valor máximo de canales LU62 . Devuelve TRUE si es satisfactorio.

MQLONG maximumLu62Channels ();.
Devuelve el valor máximo de canales LU62 sin ninguna indicación de posibles errores

ImqBoolean maximumMessageLength (MQLONG & ***longitud*** **);**
Proporciona una copia de la **longitud máxima de mensaje**. Devuelve TRUE si es satisfactorio.

MQLONG maximumMessageLength ();
Devuelve la **longitud máxima de mensaje** sin ninguna indicación de posibles errores.

ImqBoolean maximumPriority(MQLONG & ***prioridad*** **);**
Proporciona una copia de la **prioridad máxima**. Devuelve TRUE si es satisfactorio.

MQLONG maximumPriority();
Devuelve una copia de la **prioridad máxima**, sin ninguna indicación de posibles errores.

ImqBoolean maximumTcpCanales (MQLONG & canales);
Proporciona una copia del valor máximo de canales TCP. Devuelve TRUE si es satisfactorio.

MQLONG maximumTcpCanales ();
Devuelve el valor máximo de canales TCP sin ninguna indicación de posibles errores.

ImqBoolean maximumUncommittedMessages (MQLONG & ***número*** **);**
Proporciona una copia del **número máximo de mensajes no confirmados**. Devuelve TRUE si es satisfactorio.

MQLONG maximumUncommittedMensajes ();
Devuelve el **número máximo de mensajes no confirmados** sin ninguna indicación de posibles errores.

ImqBoolean mqiAccounting (MQLONG & statint);
Proporciona una copia del valor de contabilidad MQI. Devuelve TRUE si es satisfactorio.

MQLONG mqiAccounting ();
Devuelve el valor de contabilidad MQI sin ninguna indicación de posibles errores.

ImqBoolean mqiStatistics (MQLONG & statmqi);
Proporciona una copia del valor de estadísticas de MQI. Devuelve TRUE si es satisfactorio.

MQLONG mqiStatistics ();
Devuelve el valor de estadísticas MQI sin ninguna indicación de posibles errores.

ImqBoolean outboundPortMax (MQLONG & max);
Proporciona una copia del valor máximo de puerto de salida. Devuelve TRUE si es satisfactorio.

MQLONG outboundPortMax ();
Devuelve el valor de puerto de salida máximo sin ninguna indicación de posibles errores.

ImqBoolean outboundPortMín (MQLONG & min);
Proporciona una copia del valor de puerto de salida mínimo. Devuelve TRUE si es satisfactorio.

MQLONG outboundPortMin ();
Devuelve el valor de puerto de salida mínimo sin ninguna indicación de posibles errores.

Contraseña de ImqBinary () Const.
Devuelve la contraseña utilizada en las conexiones de cliente.

ImqBoolean setPassword (const ImqString & password);
Establece la contraseña utilizada en las conexiones de cliente.

ImqBoolean setPassword (const char * = 0 contraseña);
Establece la contraseña utilizada en las conexiones de cliente.

ImqBoolean setPassword (const ImqBinary & password);
Establece la contraseña utilizada en las conexiones de cliente.

ImqBoolean performanceEvent(MQLONG & ***suceso*** **);**
Proporciona una copia del estado de habilitación del **suceso de rendimiento**. Devuelve TRUE si es satisfactorio.

MQLONG performanceEvent();
Devuelve el estado de habilitación del **suceso de rendimiento** sin ninguna indicación de posibles errores.

Plataforma ImqBoolean (MQLONG & ***plataforma*** **);**
Proporciona una copia de la plataforma . Devuelve TRUE si es satisfactorio.

plataforma MQLONG ();
Devuelve la **plataforma** sin ninguna indicación de posibles errores.

ImqBoolean queueAccounting (MQLONG & acctq);
Proporciona una copia del valor de contabilidad de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueAccounting ();
Devuelve el valor de contabilidad de cola sin ninguna indicación de posibles errores.

ImqBoolean queueMonitoring (MQLONG & monq);
Proporciona una copia del valor de supervisión de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueMonitoring ();
Devuelve el valor de supervisión de cola sin ninguna indicación de posibles errores.

ImqBoolean queueStatistics (MQLONG & statq);
Proporciona una copia del valor de estadísticas de cola. Devuelve TRUE si es satisfactorio.

MQLONG queueStatistics ();
Devuelve el valor de estadísticas de cola sin ninguna indicación de posibles errores.

ImqBoolean receiveTimeout (MQLONG & timeout);
Proporciona una copia del valor de tiempo de espera de recepción. Devuelve TRUE si es satisfactorio.

MQLONG receiveTimeout ();
Devuelve el valor de tiempo de espera de recepción sin ninguna indicación de posibles errores.

ImqBoolean receiveTimeoutMin (MQLONG & min);
Proporciona una copia del valor mínimo de tiempo de espera de recepción. Devuelve TRUE si es satisfactorio.

MQLONG receiveTimeoutMin ();
Devuelve el valor de tiempo de espera de recepción mínimo sin ninguna indicación de posibles errores.

ImqBoolean receiveTimeoutTipo (MQLONG & type);
Proporciona una copia del tipo de tiempo de espera de recepción. Devuelve TRUE si es satisfactorio.

MQLONG receiveTimeoutTipo ();
Devuelve el tipo de tiempo de espera de recepción sin ninguna indicación de posibles errores.

ImqBoolean remoteEvent(MQLONG & ***suceso*** **);**
Proporciona una copia del estado de habilitación del **suceso remoto**. Devuelve TRUE si es satisfactorio.

MQLONG remoteEvent();
Devuelve el estado de habilitación del **suceso remoto** sin ninguna indicación de posibles errores.

ImqBoolean repositoryName(ImqString & ***nombre*** **);**
Proporciona una copia del **nombre de repositorio**. Devuelve TRUE si es satisfactorio.

ImqString repositoryName( );
Devuelve el **nombre de repositorio** sin ninguna indicación de posibles errores.

ImqBoolean repositoryNamelistName (ImqString & ***nombre*** **);**
Proporciona una copia del **nombre de lista de nombres de repositorio**. Devuelve TRUE si es satisfactorio.

ImqString repositoryNamelistNombre ();
Devuelve una copia del **nombre de lista de nombres de repositorio** sin ninguna indicación de posibles errores.

ImqBoolean sharedQueueQueueManagerNombre (MQLONG & name);
Proporciona una copia del valor del nombre del gestor de colas compartido. Devuelve TRUE si es satisfactorio.

MQLONG sharedQueueQueueManagerNombre ();
Devuelve el valor del nombre del gestor de colas compartido sin ninguna indicación de posibles errores.

ImqBoolean sslEvent (MQLONG & event);
Proporciona una copia del valor de suceso SSL. Devuelve TRUE si es satisfactorio.

MQLONG sslEvent ();
Devuelve el valor de suceso SSL sin ninguna indicación de posibles errores.

ImqBoolean sslFips (MQLONG & sslfips);
Proporciona una copia del valor SSL FIPS. Devuelve TRUE si es satisfactorio.

MQLONG sslFips ();
Devuelve el valor SSL FIPS sin ninguna indicación de posibles errores.

ImqBoolean sslKeyResetCount (MQLONG & count);
Proporciona una copia del valor de recuento de restablecimiento de clave SSL. Devuelve TRUE si es satisfactorio.

MQLONG sslKeyResetCount ();
Devuelve el valor de recuento de restablecimiento de clave SSL sin ninguna indicación de posibles errores.

ImqBoolean startStopEvent (MQLONG & ***suceso*** **);**
Proporciona una copia del estado de habilitación del **suceso de inicio-detención**. Devuelve TRUE si es satisfactorio.

MQLONG startStopSuceso ();
Devuelve el estado de habilitación del **suceso de inicio-detención** sin ninguna indicación de posibles errores.

ImqBoolean statisticsInterval (MQLONG & statint);
Proporciona una copia del valor de intervalo de estadísticas. Devuelve TRUE si es satisfactorio.

MQLONG statisticsInterval ();
Devuelve el valor de intervalo de estadísticas sin ninguna indicación de posibles errores.

ImqBoolean syncPointAvailability (MQLONG & ***sincronización*** **);**
Proporciona una copia del valor de **disponibilidad de punto de sincronismo** . Devuelve TRUE si es satisfactorio.

MQLONG syncPointDisponibilidad ();
Devuelve una copia del valor de **disponibilidad de punto de sincronismo** , sin ninguna indicación de posibles errores.

ImqBoolean tcpName (ImqString & name);
Proporciona una copia del nombre del sistema TCP. Devuelve TRUE si es satisfactorio.

ImqString tcpName ( );
Devuelve el nombre del sistema TCP sin ninguna indicación de posibles errores.

ImqBoolean tcpStackTipo (MQLONG & type);
Proporciona una copia del tipo de pila TCP. Devuelve TRUE si es satisfactorio.

MQLONG tcpStackTipo ();
Devuelve el tipo de pila TCP sin ninguna indicación de posibles errores.

ImqBoolean traceRouteRecording (MQLONG & routerec);
Proporciona una copia del valor de registro de ruta de rastreo. Devuelve TRUE si es satisfactorio.

MQLONG traceRouteRecording ();
Devuelve el valor de registro de ruta de rastreo sin ninguna indicación de posibles errores.

ImqBoolean triggerInterval(MQLONG & ***intervalo*** **);**
Proporciona una copia del **intervalo de desencadenante**. Devuelve TRUE si es satisfactorio.

MQLONG triggerInterval();
Devuelve el **intervalo de desencadenante** sin ninguna indicación de posibles errores.

ImqBinary userId () Const.
Devuelve el ID de usuario utilizado en las conexiones de cliente.

ImqBoolean setUserId (const ImqString & id);
Establece el ID de usuario utilizado en las conexiones de cliente.

ImqBoolean setUserId (const char * = 0 id);
Establece el ID de usuario utilizado en las conexiones de cliente.

ImqBoolean setUserId (const ImqBinary & id);
Establece el ID de usuario utilizado en las conexiones de cliente.

Métodos de objeto (protegidos)

void setFirstManagedObject(const ImqObject * ***objeto*** **= 0);**
Establece el **primer objeto gestionado**.

Datos de objeto (protegidos)

MQHCONN ***ohconn***
El descriptor de conexión de WebSphere MQ (significativo sólo mientras el **estado de conexión** es TRUE).

códigos de razón

- MQRC_ATTRIBUTE_LOCKED
- ERROR_ENTORNO_MQRC
- MQRC_FUNCTION_NOT_SUPPORTED
- ERROR_REFERENCIA_MQRC
- (códigos de razón para MQBACK)
- (códigos de razón para MQBEGIN)
- (códigos de razón para MQCMIT)
- (códigos de razón para MQCONNX)
- (códigos de razón para MQDISC)
- (códigos de razón para MQCONN)

Clase C++ de cabecera ImqReference

Esta clase encapsula características de la estructura de datos MQRMH.

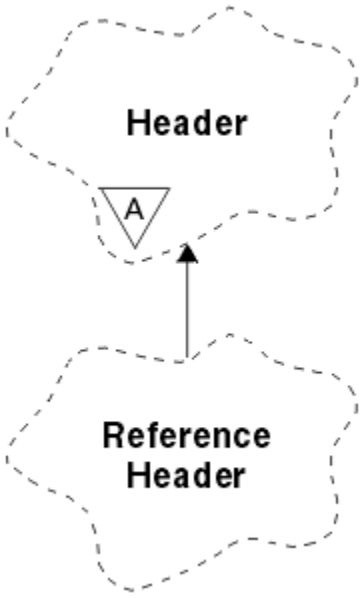


Figura 66. Clase de cabecera ImqReference

Esta clase está relacionada con las llamadas MQI listadas en “ImqReferenceReferencia cruzada de cabecera” en la página 1332.

Atributos de objetos

entorno de destino
Entorno para el destino. El valor inicial es una serie nula.

nombre de destino
Nombre del destino de datos. El valor inicial es una serie nula.

ID de instancia
Identificador de instancia. Un valor binario (MQBYTE24) de longitud MQ_OBJECT_INSTANCE_ID_LENGTH. El valor inicial es MQOII_NONE.

longitud lógica
Longitud lógica, o prevista, de los datos de mensaje que siguen a esta cabecera. El valor inicial es cero.

desplazamiento lógico
Desplazamiento lógico para los datos de mensaje siguientes, que se interpretarán en el contexto de los datos en su conjunto, en el destino final. El valor inicial es cero.

desplazamiento lógico 2
Extensión de orden superior al **desplazamiento lógico**. El valor inicial es cero.

Tipo de referencia
Tipo de referencia. El valor inicial es una serie nula.

Entorno de origen
Entorno para el origen. El valor inicial es una serie nula.

Nombre del origen
Nombre del origen de datos. El valor inicial es una serie nula.

Constructores

Cabecera ImqReference();
El constructor predeterminado.

Cabecera ImqReference(const ImqReferenceHeader & *header*);
El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut(ImqMessage & *msg*);
Inserta una estructura de datos MQRMH en el almacenamiento intermedio de mensajes al principio, moviendo los datos de mensaje existentes más adelante y establece el formato *msg* en MQFMT_REF_MSG_HEADER.

Consulte la descripción del método de clase ImqHeader en "Clase C++ ImqHeader" en la página 1360 para obtener más detalles.

virtual ImqBoolean pasteIn(ImqMessage & *msg*);
Lee una estructura de datos MQRMH del almacenamiento intermedio de mensajes.

Para ser satisfactorio, el **formato** ImqMessage debe ser MQFMT_REF_MSG_HEADER.

Consulte la descripción del método de clase ImqHeader en "Clase C++ ImqHeader" en la página 1360 para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqReferenceHeader & *cabecera*);
Copia los datos de instancia de la cabecera de , sustituyendo los datos de instancia existentes.

ImqString destinationEnvironment() const ;
Devuelve una copia del **entorno de destino**.

void setDestinationEnvironment(const char * *entorno* = 0);
Establece el **entorno de destino**.

ImqString destinationName() const ;
Devuelve una copia del **nombre de destino**.

void setDestinationName(const char * *name* = 0);
Establece el **nombre de destino**.

ImqBinary instanceId() const ;
Devuelve una copia del **ID de instancia**.

ImqBoolean setInstanceId(const ImqBinary & *id*);
Establece el **id de instancia**. La **longitud de datos** de *señal* debe ser 0 o MQ_OBJECT_INSTANCE_ID_LENGTH. Este método devuelve TRUE si es satisfactorio.

void setInstanceId(const MQBYTE24 *id* = 0);
Establece el **id de instancia**. *id* puede ser cero, que es lo mismo que especificar MQOII_NONE. Si *id* es distinto de cero, debe direccionar MQ_OBJECT_INSTANCE_ID_LENGTH bytes de datos binarios. Cuando se utilizan valores predefinidos como MQOII_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma, por ejemplo (MQBYTE *) MQOII_NONE.

MQLONG logicalLength() const ;
Devuelve la **longitud lógica**.

void setLogicalLength(const MQLONG *longitud*);
Establece la **longitud lógica**.

MQLONG logicalOffset() const ;
Devuelve el **desplazamiento lógico**.

void setLogicalOffset(const MQLONG *desplazamiento*);
Establece el **desplazamiento lógico**.

MQLONG conjuntologicalOff2() const ;
Devuelve el **desplazamiento lógico 2**.

void setLogicalOffset2(const MQLONG *desplazamiento*);
Establece el **desplazamiento lógico 2**.

ImqString referenceType() const ;
Devuelve una copia del **tipo de referencia**.

void setReferenceType(const char * *name* = 0);
Establece el **tipo de referencia**.

ImqString sourceEnvironment() const ;
Devuelve una copia del **entorno de origen**.

void setSourceEnvironment(const char * *entorno* = 0);
Establece el **entorno de origen**.

ImqString sourceName() const ;
Devuelve una copia del **nombre de origen**.

void setSourceName(const char * *name* = 0);
Establece el **nombre de origen**.

Datos de objeto (protegidos)

MQRMH *omqrmh*
Estructura de datos MQRMH.

códigos de razón

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_STRUC_LENGTH_ERROR
- MQRC_STRUC_ID_ERROR
- MQRC_INSUFICIENT_DATA
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR

Clase C++ ImqString

Esta clase proporciona almacenamiento de serie de caracteres y manipulación para series terminadas en nulo.

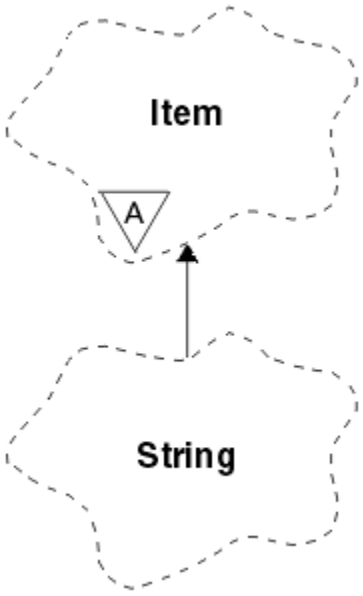


Figura 67. Clase ImqString

Utilice un ImqString en lugar de un **char *** en la mayoría de las situaciones en las que un parámetro llama a un **char ***.

- "Atributos de objetos" en la página 1418
- "Constructores" en la página 1418
- "Métodos de clase (public)" en la página 1419
- "Métodos ImqItem sobrecargados" en la página 1419
- "Métodos de objeto (public)" en la página 1419
- "Métodos de objeto (protegidos)" en la página 1422
- "códigos de razón" en la página 1422

Atributos de objetos

caracteres
Caracteres del **almacenamiento** que preceden a un nulo final.

longitud
Número de bytes en los **caracteres**. Si no hay ningún **almacenamiento**, la **longitud** es cero. El valor inicial es cero.

almacenamiento
Una matriz volátil de bytes de tamaño arbitrario. Un nulo final siempre debe estar presente en el **almacenamiento** después de los **caracteres**, para que se pueda detectar el final de los **caracteres** . Los métodos aseguran que esta situación se mantiene, pero aseguran, cuando se establecen los bytes en la matriz directamente, que existe un nulo final después de la modificación. Inicialmente, no hay ningún atributo **storage** .

Constructores

ImqString( );
El constructor predeterminado.

ImqString(const ImqString & *serie*);
El constructor de copia.

ImqString(const char *c*);
Los **caracteres** comprenden *c*.

ImqString(const char * *texto*);
Los **caracteres** se copian de *texto*.

ImqString(const void * *almacenamiento intermedio*, const size_t *longitud*);
Copia *longitud* bytes a partir de *almacenamiento intermedio* y los asigna a los **caracteres**. La sustitución se realiza para cualquier carácter nulo copiado. El carácter de sustitución es un punto (.). No se da ninguna consideración especial a ningún otro carácter no imprimible o no visualizable copiado.

Métodos de clase (public)

static ImqBoolean copy (char * *buffer-destino*, const size_t *length*, const char * *búfer-origen*, const char *pad* = 0);
Copia hasta *longitud* bytes de *búfer de origen* a *búfer de destino*. Si el número de caracteres en *búfer de origen* es insuficiente, rellena el espacio restante en *búfer de destino* con *relleno* caracteres. *búfer de origen* puede ser cero. *buffer-destino* puede ser cero si *longitud* también es cero. Los códigos de error se pierden. Este método devuelve TRUE si es satisfactorio.

static ImqBoolean copy (char * *buffer-destino*, const size_t *length*, const char * *búfer-origen*, ImqError & *objeto-error*, const char *pad* = 0);
Copia hasta *longitud* bytes de *búfer de origen* a *búfer de destino*. Si el número de caracteres en *búfer de origen* es insuficiente, rellena el espacio restante en *búfer de destino* con *relleno* caracteres. *búfer de origen* puede ser cero. *buffer-destino* puede ser cero si *longitud* también es cero. Los códigos de error se establecen en *objeto-error*. Este método devuelve TRUE si es satisfactorio.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut(ImqMessage & *msg*);
Copia los **caracteres** en el almacenamiento intermedio de mensajes, sustituyendo cualquier contenido existente. Establece el formato *msg* en MQFMT_STRING.

Consulte la descripción del método de clase padre para obtener más detalles.

virtual ImqBoolean pasteIn(ImqMessage & *msg*);
Establece los **caracteres** transfiriendo los datos restantes del almacenamiento intermedio de mensajes, sustituyendo los **caracteres**existentes.

Para ser satisfactorio, la **codificación** del objeto *msg* debe ser MQENC_NATIVE. Recuperar mensajes con MQGMO_CONVERT a MQENC_NATIVE.

Para ser satisfactorio, el **formato** ImqMessage debe ser MQFMT_STRING.

Consulte la descripción del método de clase padre para obtener más detalles.

Métodos de objeto (public)

char & operator [] (const size_t *desplazamiento*) Const.
Hace referencia al carácter en desplazamiento *desplazamiento* en el **almacenamiento**. Asegúrese de que el byte relevante existe y es direccionable.

Operador ImqString () (const size_t *desplazamiento*, const size_t *longitud* = 1) Const.
Devuelve una subserie copiando bytes de los **caracteres** a partir de *desplazamiento*. Si *longitud* es cero, devuelve el resto de los **caracteres**. Si la combinación de *desplazamiento* y *longitud* no produce una referencia dentro de los **caracteres**, devuelve una ImqStringvacía.

void operator = (const ImqString & *serie*);
Copia datos de instancia de *serie*, sustituyendo los datos de instancia existentes.

ImqString operador + (const char *c*) Const.
Devuelve el resultado de añadir *c* a los **caracteres**.

ImqString operator + (const char * *texto*) Const.
Devuelve el resultado de añadir *texto* a los **caracteres**. Esto también se puede invertir. Por ejemplo:

```
strOne + "string two" ;
"string one" + strTwo ;
```

Nota: Aunque la mayoría de compiladores aceptan **strOne + "string two";** Microsoft Visual C++ requiere **strOne + (char *) "string two";**

Operador ImqString + (const ImqString & *string1*) Const.
Devuelve el resultado de añadir *string1* a los **caracteres**.

ImqString operator + (const double *número*) Const.
Devuelve el resultado de añadir *número* a los **caracteres** después de la conversión a texto.

ImqString operator + (const long *número*) Const.
Devuelve el resultado de añadir *número* a los **caracteres** después de la conversión a texto.

void operator + = (const char *c*);
Añade *c* a los **caracteres**.

void operator + = (const char * *texto*);
Añade *texto* a los **caracteres**.

void operator + = (const ImqString & *serie*);
Añade *serie* a los **caracteres**.

void operator + = (const double *número*);
Añade *número* a los **caracteres** después de la conversión a texto.

void operator + = (const long *número*);
Añade *número* a los **caracteres** después de la conversión a texto.

operador char * () Const.
Devuelve la dirección del primer byte en el **almacenamiento**. Este valor puede ser cero y es volátil. Utilice este método sólo para fines de sólo lectura.

Operador ImqBoolean < (const ImqString & *serie*) Const.
Compara los **caracteres** con los de *serie* utilizando el método **compare** . El resultado es TRUE si es menor que y FALSE si es mayor que o igual a.

Operador ImqBoolean > (const ImqString & *serie*) Const.
Compara los **caracteres** con los de *serie* utilizando el método **compare** . El resultado es TRUE si es mayor que y FALSE si es menor o igual a.

Operador ImqBoolean < = (const ImqString & *serie*) Const.
Compara los **caracteres** con los de *serie* utilizando el método **compare** . El resultado es TRUE si es menor o igual que y FALSE si es mayor que.

Operador ImqBoolean > = (const ImqString & *serie*) Const.
Compara los **caracteres** con los de *serie* utilizando el método **compare** . El resultado es TRUE si es mayor o igual que y FALSE si es menor que.

Operador ImqBoolean == (const ImqString & *serie*) Const.
Compara los **caracteres** con los de *serie* utilizando el método **compare** . Devuelve TRUE o FALSE.

Operador ImqBoolean ! = (const ImqString & *serie*) Const.
Compara los **caracteres** con los de *serie* utilizando el método **compare** . Devuelve TRUE o FALSE.

comparación corta (const ImqString & *serie*) Const.
Compara los **caracteres** con los de la *serie*. El resultado es cero si los **caracteres** son iguales, negativo si es menor que y positivo si es mayor que. La comparación es sensible a las mayúsculas y minúsculas. Una ImqString nula se considera menor que una ImqStringno nula.

ImqBoolean copyOut(char * *buffer*, const size_t *length*, const char *pad* = 0);
Copia hasta *longitud* bytes de los **caracteres** en el *almacenamiento intermedio*. Si el número de **caracteres** es insuficiente, rellena el espacio restante en *almacenamiento intermedio* con *relleno* caracteres. *buffer* puede ser cero si *length* también es cero. Devuelve TRUE si es satisfactorio.

size_t copyOut(long &*número*) Const.

Establece *número* a partir de los **caracteres** después de la conversión a partir del texto y devuelve el número de caracteres implicados en la conversión. Si es cero, no se ha realizado ninguna conversión y no se ha establecido *número* . Una secuencia de caracteres convertible debe empezar con los valores siguientes:

```
<blank(s)>
<+|->
digit(s)
```

size_t copyOut(ImqString &*señal*, const char *c* = '') Const.

Si los **caracteres** contienen uno o más caracteres que son diferentes de *c*, identifica una señal como la primera secuencia contigua de dichos caracteres. En este caso, *token* se establece en esa secuencia y el valor devuelto es la suma del número de caracteres iniciales *c* y el número de bytes de la secuencia. De lo contrario, devuelve cero y no establece *token*.

size_t cutOut(long &*número*);

Establece *número* como para el método **copy** , pero también elimina de **caracteres** el número de bytes indicado por el valor de retorno. Por ejemplo, la serie que se muestra en el ejemplo siguiente se puede cortar en tres números utilizando **cutOut**(*número*) tres veces:

```
strNumbers = "-1 0     +55 "

while ( strNumbers.cutOut( number ) );
number becomes -1, then 0, then 55
leaving strNumbers == " "
```

size_t cutOut(ImqString &*señal*, const char *c* = " '

Establece *token* como para el método **copyOut** y elimina de **caracteres** los caracteres *strToken* y también los caracteres *c* que preceden a los caracteres *token* . Si *c* no está en blanco, elimina los caracteres *c* que son correctos directamente los caracteres *token* . Devuelve el número de caracteres eliminados. Por ejemplo, la serie que se muestra en el ejemplo siguiente se puede cortar en tres señales utilizando **cutOut**(*token*) tres veces:

```
strText = "  Program Version 1.1  "

while ( strText.cutOut( token ) );

// token becomes "Program", then "Version",
// then "1.1" leaving strText == "  "
```

El ejemplo siguiente muestra cómo analizar un nombre de vía de acceso de 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 &*serie*);

Busca una coincidencia exacta para *serie* en cualquier lugar dentro de los **caracteres**. Si no se encuentra ninguna coincidencia, devuelve FALSE. De lo contrario, devuelve TRUE. Si *serie* es nula, devuelve TRUE.

ImqBoolean find (const ImqString &*serie*, size_t &*desplazamiento*);

Busca una coincidencia exacta para *serie* en algún lugar dentro de los **caracteres** del desplazamiento *desplazamiento* en adelante. Si *serie* es nula, devuelve TRUE sin actualizar *desplazamiento*. Si no se encuentra ninguna coincidencia, devuelve FALSE (es posible que se haya aumentado el valor de

desplazamiento). Si se encuentra una coincidencia, devuelve TRUE y actualiza *desplazamiento* al desplazamiento de *serie* dentro de los **caracteres**.

size_t longitud () Const.
Devuelve la **longitud**.

ImqBoolean pasteIn(const double *número*, const char * *format* = "%f");
Añade *número* a los **caracteres** después de la conversión a texto. Devuelve TRUE si es satisfactorio.

La especificación *formato* se utiliza para formatear la conversión de coma flotante. Si se especifica, debe ser uno adecuado para su uso con **printf** y números de coma flotante, por ejemplo **%.3f**.

ImqBoolean pasteIn(const long *número*);
Añade *número* a los **caracteres** después de la conversión a texto. Devuelve TRUE si es satisfactorio.

ImqBoolean pasteIn(const void * *buffer*, const size_t *longitud*);
Añade *longitud* bytes de *almacenamiento intermedio* a los **caracteres**y añade un nulo final. Sustituye los caracteres nulos copiados. El carácter de sustitución es un punto (.). No se da ninguna consideración especial a ningún otro carácter no imprimible o no visualizable copiado. Este método devuelve TRUE si es satisfactorio.

ImqBoolean set (const char * *almacenamiento intermedio*, const size_t *longitud*);
Establece los **caracteres** de un campo de caracteres de longitud fija, que puede contener un valor nulo. Añade un valor nulo a los caracteres del campo de longitud fija si es necesario. Este método devuelve TRUE si es satisfactorio.

ImqBoolean setStorage(const size_t *longitud*);
Asigna (o reasigna) el **almacenamiento**. Conserva los **caracteres**originales, incluidos los nulos finales, si todavía hay espacio para ellos, pero no inicializa ningún almacenamiento adicional.

Este método devuelve TRUE si es satisfactorio.

size_t storage () Const.
Devuelve el número de bytes en el **almacenamiento**.

size_t stripLeading(const char *c* = '');
Elimina los caracteres iniciales *c* de los **caracteres** y devuelve el número eliminado.

size_t stripTrailing(const char *c* = '');
Elimina los caracteres de cola *c* de los **caracteres** y devuelve el número eliminado.

ImqString upperCase() Const.
Devuelve una copia en mayúsculas de los **caracteres**.

Métodos de objeto (protegidos)

ImqBoolean assign(const ImqString & *serie*);
Equivalente al método **operator =** equivalente, pero no virtual. Devuelve TRUE si es satisfactorio.

códigos de razón

- MQRC_DATA_TRUNCADO
- MQRC_NULL_POINTER
- MQRC_STORAGE_NOT_AVAILABLE
- MQRC_BUFFER_ERROR
- MQRC_INCONSISTENT_FORMAT

Clase C++ ImqTrigger

Esta clase encapsula la estructura de datos MQTM (mensaje desencadenante).

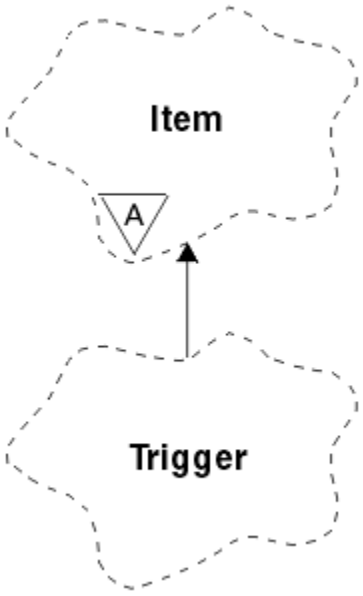


Figura 68. Clase ImqTrigger

Los objetos de esta clase suelen ser utilizados por un programa supervisor desencadenante. La tarea de un programa supervisor desencadenante es esperar a estos mensajes concretos y actuar sobre ellos para asegurarse de que otras aplicaciones de WebSphere MQ se inician cuando los mensajes los esperan.

Consulte el programa de ejemplo IMQSTRG para ver un ejemplo de uso.

- "Atributos de objetos" en la página 1423
- "Constructores" en la página 1424
- "Métodos ImqItem sobrecargados" en la página 1424
- "Métodos de objeto (public)" en la página 1424
- "Datos de objeto (protegidos)" en la página 1425
- "códigos de razón" en la página 1425

Atributos de objetos

ID de aplicación
Identidad de la aplicación que ha enviado el mensaje. El valor inicial es una serie nula.

Tipo de aplicación
Tipo de aplicación que ha enviado el mensaje. El valor inicial es cero. Son posibles los siguientes valores adicionales:

- MQAT_AIX
- MQAT_CICS
- MQAT_DOS
- MQAT_IMS
- MQAT_MVS
- MQAT_NOTES_AGENT
- MQAT_OS2
- MQAT_OS390
- MQAT_OS400
- MQAT_UNIX
- MQAT_WINDOWS

- MQAT_WINDOWS_NT
- MQAT_USER_FIRST
- MQAT_USER_LAST

Datos de entorno
Datos de entorno para el proceso. El valor inicial es una serie nula.

Nombre de proceso
Nombre del proceso. El valor inicial es una serie nula.

Nombre de cola
Nombre de la cola que se va a iniciar. El valor inicial es una serie nula.

Datos desencadenantes
Desencadenar datos para el proceso. El valor inicial es una serie nula.

datos de usuario
Datos de usuario para el proceso. El valor inicial es una serie nula.

Constructores

ImqTrigger( );
El constructor predeterminado.

ImqTrigger(const ImqTrigger & *desencadenante*);
El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut(ImqMessage & *msg*);
Escribe una estructura de datos MQTM en el almacenamiento intermedio de mensajes, sustituyendo cualquier contenido existente. Establece el formato *msg* en MQFMT_TRIGGER.

Consulte la descripción del método de clase ImqItem en "Clase C++ ImqItem" en la página 1364 para obtener más detalles.

virtual ImqBoolean pasteIn(ImqMessage & *msg*);
Lee una estructura de datos MQTM del almacenamiento intermedio de mensajes.

Para ser satisfactorio, el **formato** ImqMessage debe ser MQFMT_TRIGGER.

Consulte la descripción del método de clase ImqItem en "Clase C++ ImqItem" en la página 1364 para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqTrigger & *desencadenante*);
Copia datos de instancia de *desencadenante*, sustituyendo los datos de instancia existentes.

ImqString applicationId() const ;
Devuelve una copia del **ID de aplicación**.

void setApplicationId(const char * *id*);
Establece el **ID de aplicación**.

MQLONG applicationType() const ;
Devuelve el **tipo de aplicación**.

void setApplicationType(const MQLONG *tipo*);
Establece el **tipo de aplicación**.

ImqBoolean copyOut(MQTMC2 * *ptmc2*);
Encapsula la estructura de datos MQTM, que es la que se recibe en las colas de inicio. Rellena una estructura de datos MQTMC2 equivalente proporcionada por el llamante y establece el campo QMgrName (que no está presente en la estructura de datos MQTM) en todos los espacios en blanco.

La estructura de datos MQTMC2 se utiliza tradicionalmente como parámetro para las aplicaciones iniciadas por un supervisor desencadenante. Este método devuelve TRUE si es satisfactorio.

ImqString environmentData() const ;
Devuelve una copia de los **datos de entorno**.

void setEnvironmentData(const char * *datos*);
Establece los **datos de entorno**.

ImqString processName() const ;
Devuelve una copia del **nombre de proceso**.

void setProcessName(const char * *nombre*);
Establece el **nombre de proceso**, rellenado con espacios en blanco en 48 caracteres.

ImqString queueName() const ;
Devuelve una copia del **nombre de cola**.

void setQueueName(const char * *nombre*);
Establece el **nombre de cola**, rellenado con espacios en blanco en 48 caracteres.

ImqString triggerData() const ;
Devuelve una copia de los **datos de desencadenante**.

void setTriggerData(const char * *datos*);
Establece los **datos de desencadenante**.

ImqString userData() const ;
Devuelve una copia de los **datos de usuario**.

void setUserData(const char * *datos*);
Establece los **datos de usuario**.

Datos de objeto (protegidos)

MQTM *omqtm*
Estructura de datos MQTM.

códigos de razón

- MQRC_NULL_POINTER
- MQRC_INCONSISTENT_FORMAT
- MQRC_ENCODING_ERROR
- MQRC_STRUC_ID_ERROR

Clase C++ de cabecera ImqWork

Esta clase encapsula características específicas de la estructura de datos MQWIH.

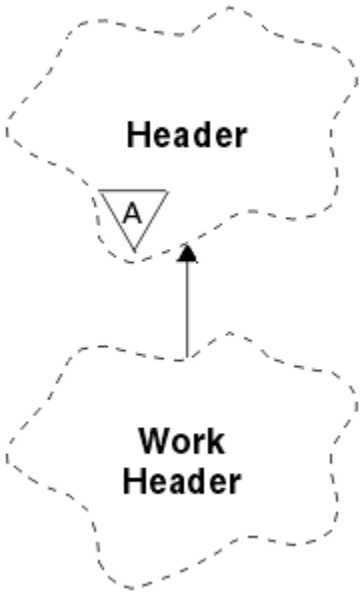


Figura 69. Clase de cabecera ImqWork

Los objetos de esta clase los utilizan las aplicaciones que colocan mensajes en la cola gestionada por el gestor de carga de trabajo de z/OS .

- "Atributos de objetos" en la página 1426
- "Constructores" en la página 1426
- "Métodos ImqItem sobrecargados" en la página 1426
- "Métodos de objeto (public)" en la página 1427
- "Datos de objeto (protegidos)" en la página 1427
- "códigos de razón" en la página 1427

Atributos de objetos

token de mensaje
Señal de mensaje para z/OS Workload Manager, de longitud MQ_MSG_TOKEN_LENGTH. El valor inicial es MQMTOK_NONE.

nombre de servicio
Nombre de 32 caracteres de un proceso. El nombre está inicialmente en blanco.

paso de servicio
El nombre de 8 caracteres de un paso dentro del proceso. El nombre está inicialmente en blanco.

Constructores

Cabecera ImqWork();
El constructor predeterminado.

ImqWorkHeader (const ImqWorkHeader & ***cabecera*** **);**
El constructor de copia.

Métodos ImqItem sobrecargados

virtual ImqBoolean copyOut(ImqMessage & ***msg*** **);**
Inserta una estructura de datos MQWIH al principio del almacenamiento intermedio de mensajes, moviendo los datos de mensaje existentes más adelante y establece el formato *msg* en MQFMT_WORK_INFO_HEADER.

Consulte la descripción del método de clase padre para obtener más detalles.

virtual ImqBoolean pasteIn(ImqMessage & *msg*);
Lee una estructura de datos MQWIH del almacenamiento intermedio de mensajes.

Para que la codificación del objeto *msg* sea correcta, debe ser MQENC_NATIVE. Recuperar mensajes con MQGMO_CONVERT a MQENC_NATIVE.

El formato ImqMessage debe ser MQFMT_WORK_INFO_HEADER.

Consulte la descripción del método de clase padre para obtener más detalles.

Métodos de objeto (public)

void operator = (const ImqWorkHeader & *cabecera*);
Copia los datos de instancia de la cabecera de , sustituyendo los datos de instancia existentes.

ImqBinary messageToken () Const.
Devuelve la **señal de mensaje**.

ImqBoolean setMessageToken (const ImqBinary & *señal*);
Establece la **señal de mensaje**. La longitud de datos de *señal* debe ser cero o MQ_MSG_TOKEN_LENGTH. Devuelve TRUE si es satisfactorio.

void setMessageToken (const MQBYTE16 *señal* = 0);
Establece la **señal de mensaje**. *token* puede ser cero, que es lo mismo que especificar MQMTOK_NONE. Si *token* es distinto de cero, debe direccionar MQ_MSG_TOKEN_LENGTH bytes de datos binarios.

Cuando se utilizan valores predefinidos como MQMTOK_NONE, es posible que tenga que realizar una conversión para garantizar una coincidencia de firma; por ejemplo, (MQBYTE *) MQMTOK_NONE.

ImqString serviceName () Const.
Devuelve el **nombre de servicio**, incluidos los espacios en blanco finales.

void setServiceName (const char * *nombre*);
Establece el **nombre de servicio**.

ImqString serviceStep () Const.
Devuelve el **paso de servicio**, incluidos los espacios en blanco finales.

void setServiceStep (const char * *paso*);
Establece el **paso de servicio**.

Datos de objeto (protegidos)

MQWIH omqwih
Estructura de datos MQWIH.

códigos de razón

- MQRC_BINARY_DATA_LENGTH_ERROR

Las clases IBM WebSphere MQ para bibliotecas Java

La ubicación de las clases IBM WebSphere MQ para bibliotecas Java varía según la plataforma. Especifique esta ubicación cuando inicie una aplicación.

Para especificar la ubicación de las bibliotecas JNI (Java Native Interface), inicie la aplicación utilizando un mandato **java** con el formato siguiente:

```
java -Djava.library.path=library_path application_name
```

donde *vía_acceso_biblioteca* es la vía de acceso a las clases de WebSphere MQ para bibliotecas Java, que incluyen las bibliotecas JNI. Tabla 615 en la página 1428 muestra la ubicación de las clases de WebSphere MQ para bibliotecas Java para cada plataforma.

Tabla 615. La ubicación de las clases de WebSphere MQ para bibliotecas Java para cada plataforma	
Plataforma	**Directorio que contiene las clases de WebSphere MQ para bibliotecas Java**
AIX	*MQ_INSTALLATION_PATH*/java/lib (bibliotecas de 32 bits) *MQ_INSTALLATION_PATH*/java/lib64 (bibliotecas de 64 bits)
HP-UX Linux (POWER, x86-64 y zSeries s390x plataformas) Solaris (plataformasx86-64 y SPARC)	*MQ_INSTALLATION_PATH*/java/lib (bibliotecas de 32 bits) *MQ_INSTALLATION_PATH*/java/lib64 (bibliotecas de 64 bits)
Linux (plataformax86)	*MQ_INSTALLATION_PATH*/java/lib
Windows	*MQ_INSTALLATION_PATH*\Java\lib (bibliotecas de 32 bits) *MQ_INSTALLATION_PATH*\Java\lib64 (bibliotecas de 64 bits)
MQ_INSTALLATION_PATH representa el directorio de alto nivel en el que está instalado WebSphere MQ.	

Nota:

1. En AIX, HP-UX, Linux (Power platform) o Solaris, utilice las bibliotecas de 32 bits o las bibliotecas de 64 bits. Utilice las bibliotecas de 64 bits sólo si está ejecutando la aplicación en una máquina virtual Java (JVM) de 64 bits en una plataforma de 64 bits. De lo contrario, utilice las bibliotecas de 32 bits.
2. En Windows, puede utilizar la variable de entorno PATH para especificar la ubicación de las bibliotecas de WebSphere MQ para Java en lugar de especificar su ubicación en el mandato **java** .
3. Para utilizar WebSphere MQ classes for Java en modalidad de enlaces en IBM i, asegúrese de que la biblioteca QMQMJAVA esté en la lista de bibliotecas.

Tareas relacionadas

Utilización de clases de WebSphere MQ para Java

Propiedades de objetos IBM WebSphere MQ classes for JMS

Todos los objetos de IBM WebSphere MQ classes for JMS tienen propiedades. Las distintas propiedades se aplican a distintos tipos de objeto. Las distintas propiedades tienen distintos valores permitidos, y los valores de propiedades simbólicas difieren entre la herramienta de administración y el código de programa.

IBM WebSphere MQ classes for JMS proporciona recursos para establecer y consultar las propiedades de los objetos utilizando la herramienta de administración JMS de WebSphere MQ , WebSphere MQ Explorer, o en una aplicación. Muchas de las propiedades sólo son relevantes para un subconjunto específico de los tipos de objeto.

Para obtener información sobre cómo utilizar la herramienta de administración JMS de WebSphere MQ , consulte Utilización de la herramienta de administración JMS de WebSphere MQ .

Tabla 616 en la página 1429 proporciona una breve descripción de cada propiedad y muestra para cada propiedad a qué tipos de objeto se aplica. Los tipos de objeto se identifican utilizando palabras clave; consulte Tipos de objeto JMS para obtener una explicación de estos.

Los números se refieren a las notas al final de la tabla. Consulte también el tema "Dependencias entre propiedades de WebSphere MQ clases para objetos JMS" en la página 1480.

Una propiedad consta de un par nombre-valor en el formato:

```
PROPERTY_NAME(property_value)
```

Los temas de esta sección listan, para cada propiedad, el nombre de la propiedad y una breve descripción, y muestra los valores de propiedad válidos utilizados en la herramienta de administración. y el método set que se utiliza para establecer el valor de la propiedad en una aplicación. Los temas también muestran los valores de propiedad válidos para cada propiedad y la correlación entre los valores de propiedad simbólica utilizados en la herramienta y sus equivalentes programables.

Los nombres de propiedad no distinguen entre mayúsculas y minúsculas y están restringidos al conjunto de nombres reconocidos que se muestran en estos temas.

Tabla 616. Nombres de propiedad y tipos de objeto aplicables

Propiedad	**Formato abreviado**	**Tipo de objeto**							
		CF	**QCF**	**TCF**	**Q**	**T**	**XACF**	**XAQCF**	**XATCF**
"APPLICATIONNAME" en la página 1432	APPNAME	Y	Y	Y			Y	Y	Y
"ASYNCEXCEPTION" en la página 1433	AEX	Y	Y	Y			Y	Y	Y
"BROKERCCDURSUBQ" en la página 1434[1]	CCDSUB					Y			
"BROKERCCSUBQ" en la página 1434[1]	CCSUB	Y		Y			Y		Y
"BROKERCONQ" en la página 1435[1]	BCON	Y		Y			Y		Y
"BROKERDURSUBQ" en la página 1435[1]	BDSUB					Y			
"BROKERPUBQ" en la página 1436[1]	BPUB	Y		Y		Y	Y		Y
"BROKERPUBQMGR" en la página 1436[1]	BPQM					Y			
"BROKERQMGR" en la página 1437[1]	BQM	Y		Y			Y		Y
"BROKERSUBQ" en la página 1437[1]	BSUB	Y		Y			Y		Y
"BROKERVER" en la página 1438[1]	BVER	Y[2]		Y[2]		Y	Y		Y
"CCDTURL" en la página 1438[3]	CCDT	Y	Y	Y			Y	Y	Y
"CCSID" en la página 1439	CCS	Y	Y	Y	Y	Y	Y	Y	Y
"CHANNEL" en la página 1439[3]	CHAN	Y	Y	Y			Y	Y	Y
"CLEANUP" en la página 1440[1]	CL	Y		Y			Y		Y
"CLEANUPINT" en la página 1440[1]	CLINT	Y		Y			Y		Y
"ConnectionNameList" en la página 1441	CNLISTA	Y	Y	Y					
"CLIENTRECONNECTOPTIONS" en la página 1441	CROPT	Y	Y	Y					
"CLIENTRECONNECTTIMEOUT" en la página 1442	CRT	Y	Y	Y					

Tabla 616. Nombres de propiedad y tipos de objeto aplicables (continuación)

Propiedad	Formato abreviado	Tipo de objeto							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
"CLIENTID" en la página 1442	CID	Y[2]	Y	Y[2]			Y	Y	Y
"CLONESUPP" en la página 1443	CLS	Y		Y			Y		Y
"COMPHDR" en la página 1443	HC	Y		Y			Y		Y
"COMPMSG" en la página 1444	MC	Y	Y	Y			Y	Y	Y
"CONNOPT" en la página 1444	CNOPT	Y	Y	Y			Y	Y	Y
"CONNTAG" en la página 1445	CNTAG	Y	Y	Y			Y	Y	Y
"description" en la página 1446	DESC	Y[2]	Y	Y[2]	Y	Y	Y	Y	Y
"DIRECTAUTH" en la página 1446	DAUTH	Y[2]		Y[2]					
"ENCODING" en la página 1447	ENC				Y	Y			
"EXPIRY" en la página 1448	EXP				Y	Y			
"FAILIFQUIESCE" en la página 1448	FIQ	Y	Y	Y	Y	Y	Y	Y	Y
"HOSTNAME" en la página 1449	HOST	Y[2]	Y	Y[2]			Y	Y	Y
"LOCALADDRESS" en la página 1449	LA	Y[2]	Y	Y[2]			Y	Y	Y
"ESTILO de nombre de mapa" en la página 1450	MNST	Y	Y	Y			Y	Y	Y
"MAXBUFFSIZE" en la página 1451	MBSZ	Y[2]		Y[2]					
"MDREAD" en la página 1451	MDR				Y	Y			
"MDWRITE" en la página 1452	MDW				Y	Y			
"MDMSGCTX" en la página 1452	MDCTX				Y	Y			
"MSGBATCHSZ" en la página 1453[1]	MBS	Y	Y	Y			Y	Y	Y
"MSGBODY" en la página 1453	MBODY				Y	Y			
"MSGRETENTION" en la página 1454	MRET	Y	Y				Y	Y	
"MSGSELECTION" en la página 1454[1]	MSEL	Y		Y			Y		Y
"MULTICAST" en la página 1455	MCAST	Y[2]		Y[2]		Y			
"OPTIMISTICPUBLICATION" en la página 1456[1]	OPTPUB	Y		Y					
"OUTCOMENOTIFICATION" en la página 1456[1]	NOTIFY	Y		Y					
"PERSISTENCE" en la página 1457	PER				Y	Y			
"POLLINGINT" en la página 1457[1]	PINT	Y	Y	Y			Y	Y	Y
"PORT" en la página 1458	PORT	Y[2]	Y	Y[2]			Y	Y	Y
"PRIORITY" en la página 1458	PRI				Y	Y			
"PROCESSDURATION" en la página 1459[1]	PROCDUR	Y		Y					

Tabla 616. Nombres de propiedad y tipos de objeto aplicables (continuación)

Propiedad	Formato abreviado	Tipo de objeto							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
"PROVIDERVERSION" en la página 1459	PVER	Y	Y	Y			Y	Y	Y
"PROXYHOSTNAME" en la página 1461	PHOST	Y[2]		Y[2]					
"PROXYPORT" en la página 1461	PPORT	Y[2]		Y[2]					
"PUBACKINT" en la página 1462[1]	PAI	Y		Y			Y		Y
"PUTASYNCALLOWED" en la página 1462	PAALD				Y	Y			
"QMANAGER" en la página 1463	QMGR	Y	Y	Y	Y		Y	Y	Y
"COLA" en la página 1463	QU				Y				
"READAHEADALLOWED" en la página 1464	RALD				Y	Y			
"READAHEADCLOSEPOLICY" en la página 1464	RACP				Y	Y			
"RECEIVECCSID" en la página 1465	RCCS				Y	Y			
"RECEIVECONVERSION" en la página 1465	RCNV				Y	Y			
"RECEIVEISOLATION" en la página 1466[1]	RCVISOL	Y		Y					
"RECEXIT" en la página 1466	RCX	Y	Y	Y			Y	Y	Y
"RECEXITINIT" en la página 1467	RCXI	Y	Y	Y			Y	Y	Y
"REPLYTOSTYLE" en la página 1467	RTOST				Y	Y			
"RESCANINT" en la página 1468[1]	RINT	Y	Y				Y	Y	
"SECEXIT" en la página 1468	SCX	Y	Y	Y			Y	Y	Y
"SECEXITINIT" en la página 1469	SCXI	Y	Y	Y			Y	Y	Y
"SENDCHECKCOUNT" en la página 1469	SCC	Y	Y	Y			Y	Y	Y
"SENDEXIT" en la página 1470	SDX	Y	Y	Y			Y	Y	Y
"SENDEXITINIT" en la página 1470	SDXI	Y	Y	Y			Y	Y	Y
"SHARECONVALLOWED" en la página 1471	SCALD	Y	Y	Y			Y	Y	Y
"SPARSESUBS" en la página 1471[1]	SSUBS	Y		Y					
"SSLCIPHERSUITE" en la página 1472	SCPHS	Y	Y	Y			Y	Y	Y
"SSLCRL" en la página 1472	SCRL	Y	Y	Y			Y	Y	Y
"SSLFIPSREQUIRED" en la página 1473	SFIPS	Y	Y	Y			Y	Y	Y
"SSLPEERNAME" en la página 1473	SPEER	Y	Y	Y			Y	Y	Y

Tabla 616. Nombres de propiedad y tipos de objeto aplicables (continuación)

Propiedad	Formato abreviado	Tipo de objeto							
		CF	QCF	TCF	Q	T	XACF	XAQCF	XATCF
"SSLRESETCOUNT" en la página 1474	SRC	Y	Y	Y			Y	Y	Y
"STATREFRESHINT" en la página 1474[1]	SRI	Y		Y			Y		Y
"SUBSTORE" en la página 1475[1]	SS	Y		Y			Y		Y
"SYNCPOINTALLGETS" en la página 1475	SPAG	Y	Y	Y			Y	Y	Y
"TARGCLIENT" en la página 1476	TC				Y	Y			
"TARGCLIENTMATCHING" en la página 1476	TCM	Y	Y				Y	Y	
"TEMPMODEL" en la página 1477	TM	Y	Y				Y	Y	
"TEMPQPREFIX" en la página 1477	TQP	Y	Y				Y	Y	
"TEMPTOPICPREFIX" en la página 1478	TTP	Y		Y			Y		Y
"Tema" en la página 1478	SUPERIOR					Y			
"TRANSPORT" en la página 1479	TRAN	Y[2]	Y	Y[2]			Y	Y	Y
"WILDCARDFORMAT" en la página 1479	WCFMT	Y		Y			Y		Y

Nota:

1. Esta propiedad se puede utilizar con la versión 7.0 de WebSphere MQ classes for JMS, pero no tiene ningún efecto para una aplicación conectada a un gestor de colas de la versión 7.0 a menos que la propiedad PROVIDERVERSION de la fábrica de conexiones se establezca en un número de versión inferior a 7.
2. Solo se admiten las propiedades BROKERVER, CLIENTID, DESCRIPTION, DIRECTAUTH, HOSTNAME, LOCALADDRESS, MAXBUFFSIZE, MULTICAST, PORT, PROXYHOSTNAME, PROXYPORT y TRANSPORT para un objeto de fábrica ConnectionFactory o TopicConnectioncuando se utiliza una conexión en tiempo real con un intermediario.
3. Las propiedades CCDTURL y CHANNEL de un objeto no deben establecerse al mismo tiempo.

APPLICATIONNAME

Una aplicación puede definir un nombre que identifique su conexión al gestor de colas. Este nombre de aplicación se muestra mediante el mandato **DISPLAY CONN MQSC/PCF** (donde el campo se denomina **APPLTAG**) o en la pantalla **Conexiones de aplicación** de IBM WebSphere MQ Explorer (donde el campo se denomina **App name**).

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: APPLICATIONNAME

Nombre abreviado de la herramienta de administración JMS: APPNAME

Acceso programático

Setters/getters

- MQConnectionFactory.setAppNombre ()
- MQConnectionFactory.getAppNombre ()

Valores

Cualquier serie válida que no tenga más de 28 caracteres. Los nombres más largos se ajustan para ajustarse eliminando los nombres de paquete iniciales, si es necesario. Por ejemplo, si la clase invocadora es `com.example.MainApp`, se utiliza el nombre completo, pero si la clase invocadora es `com.example.dictionaryAndThesaurus.multilingual.mainApp`, se utiliza el nombre `multilingual.mainApp`, porque es la combinación más larga formada por el nombre de clase y el nombre de paquete situado más a la derecha que se ajusta a la longitud disponible.

Si el nombre de clase propiamente dicho tiene más de 28 caracteres de longitud, se trunca para que quepa. Por ejemplo, `com.example.mainApplicationForSecondTestCase` pasa a ser `mainApplicationForSecondTest`.

ASYNCEXCEPTION

Esta propiedad determina si WebSphere MQ classes for JMS informa a un ExceptionListener sólo cuando se interrumpe una conexión o cuando se produce una excepción de forma asíncrona en una llamada de API JMS. Esto se aplica a todas las conexiones creadas a partir de esta ConnectionFactory que tienen un ExceptionListener registrado.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: ASYNCEXCEPTION

Nombre abreviado de la herramienta de administración JMS: AEX

Acceso programático

Setters/Getters

- MQConnectionFactory.setAsyncExcepciones ()
- MQConnectionFactory.getAsyncExcepciones ()

Valores

ASÍNCRONOS_EXCEPCIONES_TODOS

Cualquier excepción detectada de forma asíncrona, fuera del ámbito de una llamada a API síncrona, y todas las excepciones de interrupción de conexión se envían al ExceptionListener.

Entorno	Valor
Herramienta de administración JMS	ALL
Programático	WMQCONSTANTS.ASYNC_EXCEPTIONS_ALL = -1
WebSphere MQ Explorer	Todo

ASYNC_EXCEPTIONS_CONNECTIONBROKEN

Solo las excepciones que indican una conexión interrumpida se envían al ExceptionListener. Cualquier otra excepción que se produce durante el proceso asíncrono no se notifica al ExceptionListener y, por lo tanto, la aplicación no está informada de estas excepciones. V 7.5.0.8 Este es el valor predeterminado de IBM WebSphere MQ Version 7.5.0, Fixpack 8 (consulte JMS: Exception listener changes in Version 7.5).

Entorno	Valor
Herramienta de administración JMS	CONNECTIONBROKEN
Programático	WMQCONSTANTS.ASYNC_EXCEPTIONS_CONNECTIONBROKEN = 1
WebSphere MQ Explorer	Conexión interrumpida

Se define la siguiente constante adicional: V 7.5.0.8

- Desde Version 7.5.0, Fix Pack 8: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_CONNECTIONBROKEN
- Antes de Version 7.5.0, Fix Pack 8: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_ALL

Conceptos relacionados

Excepciones en WebSphere MQ Classes for JMS

BROKERCCDURSUBQ

El nombre de la cola de la que se recuperan los mensajes de suscripción duradera para un ConnectionConsumer.

Objetos aplicables

Tema

Nombre largo de la herramienta de administración JMS: BROKERCCDURSUBQ

Nombre abreviado de la herramienta de administración JMS: CCDSUB

Acceso programático

Setters/getters

- MQTopic.setBrokerCCDurSubQueue()
- MQTopic.getBrokerCCDurSubQueue()

Valores

SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE
Este es el valor predeterminado.

Cualquier serie válida

BROKERCCSUBQ

El nombre de la cola de la que se recuperan los mensajes de suscripción no duradera para un ConnectionConsumer.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: BROKERCCSUBQ

Nombre abreviado de la herramienta de administración JMS: CCSUB

Acceso programático

Setters/getters

- MQConnectionFactory.setBrokerCCSubQueue()
- MQConnectionFactory.getBrokerCCSubQueue()

Valores

SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE
Este es el valor predeterminado.

Cualquier serie válida

BROKERCONQ

El nombre de cola de control del intermediario.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: BROKERCONQ

Nombre abreviado de la herramienta de administración JMS: BCON

Acceso programático

Setters/getters

- MQConnectionFactory.setBrokerControlQueue()
- MQConnectionFactory.getBrokerControlQueue()

Valores

SYSTEM.BROKER.CONTROL.QUEUE
Este es el valor predeterminado.

Cualquier serie válida

BROKERDURSUBQ

Cuando las clases de WebSphere MQ para JMS se están utilizando en la modalidad de migración de proveedor de mensajería de WebSphere MQ , esta propiedad especifica el nombre de la cola de la que se recuperan los mensajes de suscripción duradera.

Objetos aplicables

Tema

Nombre largo de la herramienta de administración JMS: BROKERDURSUBQ

Nombre abreviado de la herramienta de administración JMS: BDSUB

Acceso programático

Setters/getters

- MQTopic.setBrokerDurSubQueue()
- MQTopic.getBrokerDurSubQueue()

Valores

SYSTEM.JMS.D.SUBSCRIBER.QUEUE
Éste es el valor predeterminado.

Cualquier serie válida
Empezando por SYSTEM.JMS.D

Conceptos relacionados

Reglas para seleccionar la modalidad de proveedor de mensajería de WebSphere MQ

BROKERPUBQ

El nombre de la cola donde se envían los mensajes publicados (el valor de la corriente de datos).

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, Topic, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: BROKERPUBQ

Nombre abreviado de la herramienta de administración JMS: BPUB

Acceso programático

Setters/getters

- MQConnectionFactory.setBrokerPubQueue
- MQConnectionFactory.getBrokerPubQueue

Valores

SYSTEM.BROKER.DEFAULT.STREAM
Este es el valor predeterminado.

Cualquier serie válida

BROKERPUBQMGR

Nombre del gestor de colas al que pertenece la cola a la que se envían los mensajes publicados en el tema.

Objetos aplicables

Tema

Nombre largo de la herramienta de administración JMS: BROKERPUBQMGR

Nombre abreviado de la herramienta de administración JMS: BPQM

Acceso programático

Setters/getters

- MQTopic.setBrokerPubQueueManager()
- MQTopic.getBrokerPubQueueManager()

Valores

null
Este es el valor predeterminado.

Cualquier serie válida

BROKERQMGR

Nombre del gestor de colas en el que se ejecuta el intermediario.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: BROKERQMGR

Nombre abreviado de la herramienta de administración JMS: BQM

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setBrokerQueueManager()
- MQConnectionFactory.getBrokerQueueManager()

Valores

null
Este es el valor predeterminado.

Cualquier serie válida

BROKERSUBQ

Cuando las clases de WebSphere MQ para JMS se están utilizando en la modalidad de migración de proveedor de mensajería de WebSphere MQ , esta propiedad especifica el nombre de la cola de la que se recuperan los mensajes de suscripción no duradera.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: BROKERSUBQ

Nombre abreviado de la herramienta de administración JMS: BSUB

Acceso programático

Setters/getters

- MQConnectionFactory.setBrokerSubQueue()
- MQConnectionFactory.getBrokerSubQueue()

Valores

SYSTEM.JMS.ND.SUBSCRIBER.QUEUE
Éste es el valor predeterminado.

Cualquier serie válida
Empezando por SYSTEM.JMS.ND

Conceptos relacionados

Reglas para seleccionar la modalidad de proveedor de mensajería de WebSphere MQ

BROKERVER

La versión del intermediario que se está utilizando.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, Topic, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: BROKERVER

Nombre abreviado de la herramienta de administración JMS: BVER

Acceso programático

Setters/getters

- MQConnectionFactory.setBrokerVersión ()
- MQConnectionFactory.getBrokerVersión ()

Valores

V1
Para utilizar un intermediario de publicación/suscripción de WebSphere MQ , o para utilizar un intermediario de WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker o WebSphere Business Integration Message Broker en modalidad de compatibilidad. Este es el valor predeterminado si TRANSPORT se establece en BIND o CLIENT.

V2
Para utilizar un intermediario de WebSphere MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker o WebSphere Business Integration Message Broker en modalidad nativa. Este es el valor predeterminado si TRANSPORT se establece en DIRECT o DIRECTHTTP.

no especificada
Después de que el intermediario haya migrado de V6 a V7, establezca esta propiedad para que las cabeceras RFH2 ya no se utilicen. Después de la migración, esta propiedad ya no es relevante.

CCDTURL

Un localizador uniforme de recursos (URL) que identifica el nombre y la ubicación del archivo que contiene la tabla de definición de canal de cliente y especifica cómo se puede acceder al archivo.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: CCDTURL

Nombre abreviado de la herramienta de administración JMS: CCDT

Acceso programático

Setters/getters

- MQConnectionFactory.setCCDTURL()
- MQConnectionFactory.getCCDTURL()

Valores

null
Este es el valor predeterminado.

Un localizador universal de recursos (URL)

CCSID

El ID de juego de caracteres codificado que se va a utilizar para una conexión o destino.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: CCSID

Nombre abreviado de la herramienta de administración JMS: CCS

Acceso programático

Setters/getters

- MQConnectionFactory.setCCSID()
- MQConnectionFactory.getCCSID()

Valores

819
Este es el valor predeterminado para una fábrica de conexiones.

1208
Es el valor predeterminado para un destino.

Cualquier entero positivo

CHANNEL

El nombre del canal de conexión de cliente que se está utilizando.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: CHANNEL

Nombre abreviado de la herramienta de administración JMS: CHAN

Acceso programático

Setters/getters

- MQConnectionFactory.setChannel()
- MQConnectionFactory.getChannel()

Valores

SYSTEM.DEF.SVRCONN
Este es el valor predeterminado.

Cualquier serie válida

CLEANUP

Nivel de limpieza para almacenes de suscripciones BROKER o MIGRATE.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: CLEANUP

Nombre abreviado de la herramienta de administración JMS: CL

Acceso programático

Setters/getters

- MQConnectionFactory.setCleanupNivel ()
- MQConnectionFactory.getCleanupNivel ()

Valores

SEGURA
Utilice una limpieza segura. Este es el valor predeterminado.

ASPROP
Utilice una limpieza segura, fuerte o nula de acuerdo con una propiedad establecida en la línea de mandatos de Java.

NINGUNO
No utilizar limpieza.

STRONG
Utilice una limpieza fuerte.

CLEANUPINT

El intervalo, en milisegundos, entre ejecuciones en segundo plano del programa de utilidad de limpieza de publicación/suscripción.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: CLEANUPINT

Nombre abreviado de la herramienta de administración JMS: CLINT

Acceso programático

Setters/getters

- MQConnectionFactory.setCleanupIntervalo ()
- MQConnectionFactory.getCleanupIntervalo ()

Valores

3600000
Este es el valor predeterminado.

Cualquier entero positivo

ConnectionNameList

Lista de nombres de conexión TCP/IP. La lista se intenta en orden, una vez por cada reintento de reconexión.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración JMS: CONNECTIONNAMELIST

Nombre abreviado de la herramienta de administración JMS: CNLIST

Acceso programático

Setters/getters

- MQConnectionFactory.setconnectionNameList ()
- MQConnectionFactory.getconnectionNameLista ()

Valores

Lista separada por comas de HOSTNAME (PORT). HOSTNAME puede ser un nombre DNS o una dirección IP.

PORT toma el valor predeterminado de 1414.

CLIENTRECONNECTOPTIONS

Opciones que rigen la reconexión.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración JMS: CLIENTRECONNECTOPTIONS

Nombre abreviado de la herramienta de administración JMS: CROPT

Acceso programático

Métodos setter/getters

- MQConnectionFactory.setClientReconnectOptions()
- MQConnectionFactory.getClientReconnectOptions()

Valores

QMGR

La aplicación se puede volver a conectar pero solamente al gestor de colas al que se ha conectado originalmente.

Utilice este valor si una aplicación se puede reconectar, pero existe una afinidad entre las clases de WebSphere MQ para la aplicación JMS y el gestor de colas con el que estableció por primera vez una conexión.

Elija este valor si desea que una aplicación se vuelva a conectar automáticamente a la instancia en espera de un gestor de colas de alta disponibilidad.

Para utilizar este valor mediante programación, utilice la constante WMQConstants.WMQ_CLIENT_RECONNECT_Q_MGR.

CUALQUIERA

La aplicación se puede reconectar a cualquier gestor de colas.

Utilice la opción de reconexión sólo si no hay ninguna afinidad entre las clases de WebSphere MQ para la aplicación JMS y el gestor de colas con el que estableció inicialmente una conexión.

Para utilizar este valor de un programa, utilice la constante WMQConstants.WMQ_CLIENT_RECONNECT.

DISABLED

La aplicación no se reconectará.

Para utilizar este valor mediante programación, utilice la constante WMQConstants.WMQ_CLIENT_RECONNECT_DISABLED.

ASDEF

Si la aplicación se volverá a conectar automáticamente depende del valor del atributo de canal DefReconnectde WebSphere MQ .

Para utilizar este valor de un programa, utilice la constante WMQConstants.WMQ_CLIENT_RECONNECT_AS_DEF.

CLIENTRECONNECTTIMEOUT

Tiempo antes de que cesen los reintentos de reconexión.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory

Nombre largo de la herramienta de administración JMS: CLIENTRECONNECTTIMEOUT

Nombre abreviado de la herramienta de administración JMS: CRT

Acceso programático

Setters/getters

- MQConnectionFactory.setClientReconnectTimeout()
- MQConnectionFactory.setClientReconnectTimeout()

Valores

Intervalo en segundos. Valor predeterminado 1800 (30 minutos).

CLIENTID

El identificador de cliente se utiliza para identificar de forma exclusiva la conexión de aplicación para las suscripciones duraderas.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: CLIENTID

Nombre abreviado de la herramienta de administración JMS: CID

Acceso programático

Setters/getters

- MQConnectionFactory.setClientId ()
- MQConnectionFactory.getClientId ()

Valores

null
Este es el valor predeterminado.

Cualquier serie válida

CLONESUPP

Determina si dos o más instancias del mismo suscriptor de tema duradero se pueden ejecutar de forma simultánea.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: CLONESUPP

Nombre abreviado de la herramienta de administración JMS: CLS

Acceso programático

Setters/getters

- MQConnectionFactory.setCloneSoporte ()
- MQConnectionFactory.getCloneSoporte ()

Valores

DISABLED
Sólo se puede ejecutar una instancia de un suscriptor de tema duradero a la vez. Este es el valor predeterminado.

ENABLED
Dos o más instancias del mismo suscriptor de tema duradero pueden ejecutarse simultáneamente, pero cada instancia debe ejecutarse en una máquina virtual Java (JVM) independiente.

COMPHDR

Lista de las técnicas que se pueden utilizar para comprimir datos de cabecera en una conexión.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: COMPHDR

Nombre abreviado de la herramienta de administración JMS: HC

Acceso programático

Setters/getters

- MQConnectionFactory.setHdrCompList()
- MQConnectionFactory.getHdrCompList()

Valores

NINGUNO
Este es el valor predeterminado.

SISTEMA
Se realiza la compresión de cabecera de mensaje RLE.

COMPMSG

Lista de las técnicas que se pueden utilizar para comprimir datos de mensaje en una conexión.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: COMPMSG

Nombre abreviado de la herramienta de administración JMS: MC

Acceso programático

Setters/getters

- MQConnectionFactory.setMsgCompList()
- MQConnectionFactory.getMsgCompList()

Valores

NINGUNO
Este es el valor predeterminado.

Una lista de uno o varios de los valores siguientes separados por caracteres en blanco:
RLE ZLIBFAST ZLIBHIGH

CONNOPT

Controla cómo las clases de WebSphere MQ para aplicaciones JMS que utilizan el transporte de enlaces se conectan al gestor de colas.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: CONNOPT

Nombre abreviado de la herramienta de administración JMS: CNOPT

Acceso programático

Setters/getters

- MQConnectionFactory.setMQConnectionOpciones ()
- MQConnectionFactory.getMQConnectionOpciones ()

Valores

ESTÁNDAR
La naturaleza del enlace entre la aplicación y el gestor de colas depende del valor del atributo *DefaultBindTipo* del gestor de colas. El valor STANDARD se correlaciona con WebSphere MQ *ConnectOption* MQCNO_STANDARD_BINDING

SHARED
La aplicación y el agente del gestor de colas local se ejecutan en unidades de ejecución independientes pero comparten algunos recursos. Este valor se correlaciona con WebSphere MQ *ConnectOption* MQCNO_SHARED_BINDING.

Aislado
La aplicación y el agente del gestor de colas local se ejecutan en unidades de ejecución independientes y no comparten recursos. El valor AISLADO se correlaciona con WebSphere MQ *ConnectOption* MQCNO_ISOLATED_BINDING.

FASTPATH
La aplicación y el agente del gestor de colas local se ejecutan en la misma unidad de ejecución. Este valor se correlaciona con WebSphere MQ *ConnectOption* MQCNO_FASTPATH_BINDING.

SERIALQM
La aplicación solicita el uso exclusivo del código de conexión dentro del ámbito del gestor de colas. Este valor se correlaciona con WebSphere MQ *ConnectOption* MQCNO_SERIALIZE_CONN_TAG_Q_MGR.

SERIALQSG
La aplicación solicita el uso exclusivo del código de conexión dentro del ámbito del grupo de compartición de colas al que pertenece el gestor de colas. El valor SERIALQSG se correlaciona con WebSphere MQ *ConnectOption* MQCNO_SERIALIZE_CONN_TAG_QSG.

RESTRICTQM
La aplicación solicita el uso compartido del código de conexión, pero hay restricciones en el uso compartido del código de conexión dentro del ámbito del gestor de colas. Este valor se correlaciona con WebSphere MQ *ConnectOption* MQCNO_RESTRICT_CONN_TAG_Q_MGR.

RESTRICTQSG
La aplicación solicita el uso compartido del código de conexión, pero existen restricciones sobre el uso compartido del código de conexión dentro del ámbito del grupo de compartición de colas al que pertenece el gestor de colas. Este valor se correlaciona con WebSphere MQ *ConnectOption* MQCNO_RESTRICT_CONN_TAG_QSG.

Para obtener más información sobre las opciones de conexión de WebSphere MQ , consulte Conexión a un gestor de colas utilizando la llamada MQCONNX

CONNTAG

Código que el gestor de colas asocia con los recursos actualizados por la aplicación dentro de una unidad de trabajo mientras la aplicación está conectada al gestor de colas.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: CONNTAG

Nombre abreviado de la herramienta de administración JMS: CNTAG

Acceso programático

Setters/getters

- MQConnectionFactory.setConnEtiqueta ()
- MQConnectionFactory.getConnEtiqueta ()

Valores

Una matriz de bytes de 128 elementos, donde cada elemento es 0
Este es el valor predeterminado.

Cualquier serie
El valor se trunca si tiene más de 128 bytes.

description

Una descripción del objeto almacenado.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: DESCRIPTION

Nombre abreviado de la herramienta de administración JMS: DESC

Acceso programático

Setters/getters

- MQConnectionFactory.setDescription()
- MQConnectionFactory.getDescription()

Valores

null
Este es el valor predeterminado.

Cualquier serie válida
`

DIRECTAUTH

Si se utiliza la autenticación SSL en una conexión en tiempo real con un intermediario.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración JMS: DIRECTAUTH

Nombre abreviado de la herramienta de administración JMS: DAUTH

Acceso programático

Setters/getters

- MQConnectionFactory.setDirectAuth ()
- MQConnectionFactory.getDirectAuth ()

Valores

BÁSICO
Sin autenticación, autenticación de nombre de usuario o autenticación de contraseña. Este es el valor predeterminado.

CERTIFICATE
Autenticación de certificado de clave pública.

ENCODING

Cómo se representan los datos numéricos en el cuerpo de un mensaje cuando se envía el mensaje a este destino. La propiedad especifica la representación de enteros binarios, enteros decimales empaquetados y números de coma flotante.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración JMS: ENCODING

Nombre abreviado de la herramienta de administración JMS: ENC

Acceso programático

Setters/getters

- MQDestination.setEncoding()
- MQDestination.getEncoding()

Valores

La propiedad de codificación

Los valores válidos que puede tomar la propiedad ENCODING se construyen a partir de las tres subpropiedades:

Codificación de enteros
Normal o invertido

Codificación decimal
Normal o invertido

codificación de coma flotante
IEEE normal, IEEE invertido o z/OS

La propiedad ENCODING se expresa como una serie de tres caracteres con la sintaxis siguiente:

```
{N|R}{N|R}{N|R|3}
```

En esta serie:

- N indica normal
- R indica invertido
- 3 denota z/OS
- El primer carácter representa *codificación de enteros*
- El segundo carácter representa la *codificación decimal*
- El tercer carácter representa la *codificación de coma flotante*

Esto proporciona un conjunto de doce valores posibles para la propiedad ENCODING .

Hay un valor adicional, la serie NATIVE, que establece los valores de codificación adecuados para la plataforma Java.

Los ejemplos siguientes muestran combinaciones válidas para ENCODING:

```
ENCODING(NNR)
ENCODING(NATIVE)
ENCODING(RR3)
```

EXPIRY

La hora después de la cual caducan los mensajes en un destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración JMS: EXPIRE

Nombre abreviado de la herramienta de administración JMS: EXP

Acceso programático

Setters/getters

- MQDestination.setExpiry()
- MQDestination.getExpiry()

Valores

APP
La caducidad puede ser definida por la aplicación JMS. Este es el valor predeterminado.

UNLIM
No se produce ninguna caducidad.

0
No se produce ninguna caducidad.

Cualquier entero positivo que represente la caducidad en milisegundos.

FAILIFQUIESCE

Esta propiedad determina si las llamadas a determinados métodos fallan si el gestor de colas está en un estado de inmovilización, o si una aplicación se está conectando a un gestor de colas utilizando el transporte CLIENT y el canal que la aplicación está utilizando se ha puesto en un estado de inmovilización, por ejemplo, utilizando el mandato **STOP CHANNEL** o **STOP CHANNEL MODE(QUIESCE)** MQSC.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, Topic, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: FAILIFQUIESCE

Nombre abreviado de la herramienta de administración JMS: FIQ

Acceso programático

Setters/getters

- MQConnectionFactory.setFailIfQuiesce()
- MQConnectionFactory.getFailIfQuiesce()

Valores

SÍ
Las llamadas a determinados métodos fallan si el gestor de colas está en un estado de desactivación temporal o si el canal que se utiliza para conectarse a un gestor de colas está desactivado temporalmente. Si una aplicación detecta cualquiera de estas condiciones, la aplicación puede

completar su tarea inmediata y cerrar la conexión, permitiendo que se detenga el gestor de colas o la instancia de canal. Este es el valor predeterminado.

No
Ninguna llamada de método falla porque el gestor de colas, o el canal que se está utilizando para conectarse a un gestor de colas, está en un estado de desactivación temporal. Si especifica este valor, una aplicación no puede detectar que el gestor de colas o el canal se está desactivando temporalmente. La aplicación puede continuar realizando operaciones en el gestor de colas y, por lo tanto, impedir que se detenga el gestor de colas.

HOSTNAME

Para una conexión con un gestor de colas, el nombre de host o la dirección IP del sistema en el que se ejecuta el gestor de colas o, para una conexión en tiempo real con un intermediario, el nombre de host o la dirección IP del sistema en el que se ejecuta el intermediario.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: HOSTNAME

Nombre abreviado de la herramienta de administración JMS: HOST

Acceso programático

Setters/getters

- MQConnectionFactory.setHostNombre ()
- MQConnectionFactory.getHostNombre ()

Valores

localhost
Este es el valor predeterminado.

Cualquier serie válida

LOCALADDRESS

Para una conexión con un gestor de colas, esta propiedad especifica la interfaz de red local que se va a utilizar, o el puerto local o el rango de puertos locales que se va a utilizar. Para una conexión en tiempo real con un intermediario, esta propiedad sólo es relevante cuando se utiliza la multidifusión y especifica la interfaz de red local que se va a utilizar.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: LOCALADDRESS

Nombre abreviado de la herramienta de administración JMS: LA

Acceso programático

Setters/getters

- MQConnectionFactory.setLocalDirección ()
- MQConnectionFactory.getLocalDirección ()

Valores

"" (serie vacía)
Este es el valor predeterminado.

Una serie con el formato [ip-addr] [(low-port [, high port])]
A continuación, se detallan algunos ejemplos:

192.0.2.0

El canal se enlaza a la dirección 192.0.2.0 localmente.

192.0.2.0(1000)

El canal se enlaza a la dirección 192.0.2.0 localmente y utiliza el puerto 1000.

192.0.2.0(1000,2000)

El canal se enlaza a la dirección 192.0.2.0 localmente y utiliza un puerto en el rango de 1000 a 2000.

(1000)

El canal se enlaza al puerto 1000 localmente.

(1000,2000)

El canal se enlaza a un puerto en el rango de 1000 a 2000 localmente.

Puede especificar un nombre de host en lugar de una dirección IP. Para una conexión en tiempo real con un intermediario, esta propiedad sólo es relevante cuando se utiliza la multidifusión, y el valor de la propiedad no debe contener un número de puerto o un rango de números de puerto. Los únicos valores válidos de la propiedad en este caso son null, una dirección IP o un nombre de host.

ESTILO de nombre de mapa

Permite utilizar el estilo de compatibilidad para los nombres de elemento MapMessage .

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: MAPNAMESTYLE

Nombre abreviado de la herramienta de administración JMS: MNST

Acceso programático

Setters/getters

- MQConnectionFactory.setMapNameStyle()
- MQConnectionFactory.getMapNameStyle()

Valores

ESTÁNDAR
Se va a utilizar el formato de denominación de elementos com.ibm.jms.JMSMapMessage estándar. Este es el valor predeterminado y permite que se utilicen identificadores Java no legales como nombre de elemento.

Compatible
Se va a utilizar el formato de denominación de elementos com.ibm.jms.JMSMapMessage más antiguo. Sólo se pueden utilizar identificadores Java legales como nombre de elemento. Esto sólo es necesario si se envían mensajes de correlación a una aplicación que utiliza una versión de IBM WebSphere MQ classes for JMS anterior a la versión 5.3.

MAXBUFFSIZE

Número máximo de mensajes recibidos que se pueden almacenar en un almacenamiento intermedio de mensajes interno mientras se espera a que la aplicación los procese. Esta propiedad sólo se aplica cuando TRANSPORT tiene el valor DIRECT o DIRECTHTTP.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración JMS: MAXBUFFSIZE

Nombre abreviado de la herramienta de administración JMS: MBSZ

Acceso programático

Setters/getters

- MQConnectionFactory.setMaxBufferSize()
- MQConnectionFactory.getMaxBufferSize()

Valores

1000
Este es el valor predeterminado.

Cualquier entero positivo

MDREAD

Esta propiedad determina si una aplicación JMS puede extraer los valores de los campos MQMD.

Objetos aplicables

Nombre largo de la herramienta de administración JMS: MDREAD

Nombre abreviado de la herramienta de administración JMS: MDR

Acceso programático

Setters/getters

- MQDestination.setMQMDReadEnabled()
- MQDestination.getMQMDReadEnabled()

Valores

No
Al enviar mensajes, las propiedades JMS_IBM_MQMD* en un mensaje enviado no se actualizan para reflejar los valores de campo actualizados en el MQMD. Al recibir mensajes, ninguna de las propiedades JMS_IBM_MQMD* está disponible para un mensaje recibido, incluso si el emisor ha establecido todas o algunas de ellas. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice False.

Sí
Al enviar mensajes, todas las propiedades JMS_IBM_MQMD* de un mensaje enviado se actualizan para reflejar los valores de campo actualizados en MQMD, incluidas las propiedades que el remitente no ha establecido explícitamente. Al recibir mensajes, todas las propiedades JMS_IBM_MQMD* están disponibles en un mensaje recibido, incluidas las propiedades que el remitente no ha establecido explícitamente.

Para los programas, utilice True.

MDWRITE

Esta propiedad determina si una aplicación JMS puede establecer los valores de los campos MQMD.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración JMS: MDWRITE

Nombre abreviado de la herramienta de administración JMS: MDR

Acceso programático

Setters/getters

- MQDestination.setMQMDWriteEnabled()
- MQDestination.getMQMDWriteEnabled()

Valores

No

Todas las propiedades JMS_IBM_MQMD* se ignoran y sus valores no se copian en la estructura MQMD subyacente. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice False.

SÍ

Se procesan las propiedades JMS_IBM_MQMD*. Sus valores se copian en la estructura MQMD subyacente.

Para los programas, utilice True.

MDMSGCTX

Qué nivel de contexto de mensaje debe establecer la aplicación JMS. La aplicación debe estar en ejecución con la autoridad de contexto adecuada para que esta propiedad tenga efecto.

Objetos aplicables

Nombre largo de la herramienta de administración JMS: MDMSGCTX

Nombre abreviado de la herramienta de administración JMS: MDCTX

Acceso programático

Setters/getters

- MQDestination.setMQMDMessageContext()
- MQDestination.getMQMDMessageContext()

Valores

DEFAULT

La llamada de API MQOPEN y la estructura MQPMO no especifican opciones de contexto de mensaje explícitas. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice WMQ_MDCTX_DEFAULT.

SET_IDENTITY_CONTEXT
La llamada de la API MQOPEN especifica la opción de contexto de mensaje MQOO_SET_IDENTITY_CONTEXT y la estructura MQPMO especificaMQPMO_SET_IDENTITY_CONTEXT.

Para los programas, utilice WMQ_MDCTX_SET_IDENTITY_CONTEXT.

SET_ALL_CONTEXT
La llamada de la API MQOPEN especifica la opción de contexto de mensaje MQOO_SET_ALL_CONTEXT y la estructura MQPMO especifica MQPMO_SET_ALL_CONTEXT.

Para programas, utilice WMQ_MDCTX_SET_ALL_CONTEXT.

MSGBATCHSZ

Número máximo de mensajes que se deben tomar de una cola en un paquete cuando se utiliza la entrega de mensajes asíncrona.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: MAXBUFFSIZE

Nombre abreviado de la herramienta de administración JMS: MBSZ

Acceso programático

Setters/getters

- MQConnectionFactory.setMsgBatchSize()
- MQConnectionFactory.getMsgBatchSize()

Valores

10
Este es el valor predeterminado.

Cualquier entero positivo

MSGBODY

Determina si una aplicación JMS accede a la MQRFH2 de un mensaje IBM WebSphere MQ como parte de la carga útil del mensaje.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración JMS: WMQ_MESSAGE_BODY

Nombre abreviado de la herramienta de administración JMS: MBODY

Acceso programático

Setters/getters

- MQConnectionFactory.setMessageBodyStyle()
- MQConnectionFactory.getMessageBodyStyle()

Valores

SIN ESPECIFICAR
En el envío, determina si IBM WebSphere MQ classes for JMS genera e incluye una cabecera MQRFH2, dependiendo del valor de WMQ_TARGET_CLIENT. Al recibir, actúa como valor JMS.

JMS
Al enviar, IBM WebSphere MQ classes for JMS genera automáticamente una cabecera MQRFH2 y la incluye en el mensaje WebSphere MQ .

Al recibir, IBM WebSphere MQ classes for JMS establezca las propiedades del mensaje JMS de acuerdo con los valores de MQRFH2 (si está presente); no presenta MQRFH2 como parte del cuerpo del mensaje JMS.

MQ
En el envío, IBM WebSphere MQ classes for JMS no genera una cabecera MQRFH2.

Al recibir, IBM WebSphere MQ classes for JMS presenta MQRFH2 como parte del cuerpo del mensaje JMS.

MSGRETENTION

Indica si el consumidor de conexión mantiene los mensajes no entregados en la cola de entrada.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory,

Nombre largo de la herramienta de administración JMS: MSGRETENTION

Nombre abreviado de la herramienta de administración JMS: MRET

Acceso programático

Setters/getters

- MQConnectionFactory.setMessageRetención ()
- MQConnectionFactory.getMessageRetención ()

Valores

Sí
Los mensajes no entregados permanecen en la cola de entrada. Este es el valor predeterminado.

No
Los mensajes no entregados se tratan de acuerdo con sus opciones de disposición.

MSGSELECTION

Determina si la selección de mensajes la realizan las clases WebSphere MQ para JMS o el intermediario. Si TRANSPORT tiene el valor DIRECT, el intermediario siempre realiza la selección de mensajes y se ignora el valor de MSGSELECTION. La selección de mensajes por parte del intermediario no está soportada cuando BROKERVER tiene el valor V1.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: MSGSELECTION

Nombre abreviado de la herramienta de administración JMS: MSEL

Acceso programático

Setters/getters

- MQConnectionFactory.setMessageSelection ()
- MQConnectionFactory.getMessageSelección ()

Valores

CLIENT
La selección de mensajes la realizan las clases de WebSphere MQ para JMS. Este es el valor predeterminado.

BROKER
La selección de mensajes ha sido realizada por el intermediario.

MULTICAST

Para habilitar la multidifusión en una conexión en tiempo real con un intermediario y, si está habilitada, para especificar la forma precisa en la que se utiliza la multidifusión para entregar mensajes del intermediario a un consumidor de mensajes. La propiedad no tiene ningún efecto sobre cómo un productor de mensajes envía mensajes a un intermediario.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, Tema

Nombre largo de la herramienta de administración JMS: MULTICAST

Nombre abreviado de la herramienta de administración JMS: MCAST

Acceso programático

Setters/getters

- MQConnectionFactory.setMulticast()
- MQConnectionFactory.getMulticast()

Valores

DISABLED
Los mensajes no se entregan a un consumidor de mensajes utilizando el transporte de multidifusión. Este es el valor predeterminado para los objetos ConnectionFactory y TopicConnectionFactory.

ASCF
Los mensajes se entregan a un consumidor de mensajes de acuerdo con el valor de multidifusión para la fábrica de conexiones asociada al consumidor de mensajes. El valor de multidifusión para la fábrica de conexiones se anota en el momento en que se crea el consumidor de mensajes. Este valor sólo es válido para los objetos de tema y es el valor predeterminado para los objetos de tema.

ENABLED
Si el tema está configurado para multidifusión en el intermediario, los mensajes se entregan a un consumidor de mensajes utilizando el transporte de multidifusión. Se utiliza una calidad de servicio fiable si el tema está configurado para la multidifusión fiable.

PROVEEDOR
Si el tema está configurado para multidifusión fiable en el intermediario, los mensajes se entregan al consumidor de mensajes utilizando el transporte de multidifusión con una calidad de servicio fiable. Si el tema no está configurado para la multidifusión fiable, no puede crear un consumidor de mensajes para el tema.

NOTR
Si el tema está configurado para multidifusión en el intermediario, los mensajes se entregan al consumidor de mensajes utilizando el transporte de multidifusión. No se utiliza una calidad de servicio fiable, aunque el tema se haya configurado para la multidifusión fiable.

OPTIMISTICPUBLICATION

Esta propiedad determina si WebSphere MQ classes for JMS devuelve el control inmediatamente a un publicador que ha publicado un mensaje, o si devuelve el control sólo después de que haya completado todo el proceso asociado a la llamada y pueda informar del resultado al publicador.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración JMS: OPTIMITICPUBLICATION

Nombre abreviado de la herramienta de administración JMS: OPTPUB

Acceso programático

Setters/getters

- MQConnectionFactory.setOptimisticPublicación ()
- MQConnectionFactory.getOptimisticPublicación ()

Valores

No
Cuando un publicador publica un mensaje, las clases WebSphere MQ para JMS no devuelven el control al publicador hasta que ha completado todo el proceso asociado a la llamada y puede notificar el resultado al publicador. Este es el valor predeterminado.

SÍ
Cuando un publicador publica un mensaje, WebSphere MQ classes for JMS devuelve el control al publicador inmediatamente, antes de que haya completado todo el proceso asociado con la llamada y pueda informar del resultado al publicador. WebSphere MQ classes for JMS notifica el resultado sólo cuando el publicador confirma el mensaje.

OUTCOMENOTIFICATION

Esta propiedad determina si WebSphere MQ classes for JMS devuelve el control inmediatamente a un suscriptor que acaba de reconocer o confirmar un mensaje, o si devuelve el control sólo después de que haya completado todo el proceso asociado a la llamada y pueda notificar el resultado al suscriptor.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración JMS: OUTCOMENOTIFICACIÓN

Nombre abreviado de la herramienta de administración JMS: NOTIFY

Acceso programático

Setters/getters

- MQConnectionFactory.setOutcomeNotificación ()
- MQConnectionFactory.getOutcomeNotificación ()

Valores

SÍ
Cuando un suscriptor reconoce o confirma un mensaje, las clases WebSphere MQ para JMS no devuelven el control al suscriptor hasta que haya completado todo el proceso asociado con la llamada y pueda notificar el resultado al suscriptor. Este es el valor predeterminado.

No
Cuando un suscriptor reconoce o confirma un mensaje, WebSphere MQ classes for JMS devuelve el control al suscriptor inmediatamente, antes de que haya completado todo el proceso asociado a la llamada y pueda notificar el resultado al suscriptor.

PERSISTENCE

Persistencia de los mensajes enviados a un destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración JMS: PERSISTENCE

Nombre abreviado de la herramienta de administración JMS: PER

Acceso programático

Setters/getters

- MQDestination.setPersistence()
- MQDestination.getPersistence()

Valores

APP
La persistencia está definida por la aplicación JMS. Este es el valor predeterminado.

qdef
La persistencia toma el valor predeterminado de la cola.

pers
Los mensajes son persistentes.

no
Los mensajes no son persistentes.

SUPERIOR
Consulte Mensajes persistentes JMS para obtener más información sobre el uso de este valor.

POLLINGINT

Si cada escucha de mensajes dentro de una sesión no tiene ningún mensaje adecuado en su cola, este es el intervalo máximo, en milisegundos, que transcurre antes de que cada escucha de mensajes intente de nuevo obtener un mensaje de su cola. Si con frecuencia sucede esto de que no haya disponible ningún mensaje adecuado para ninguno de los escuchas de mensajes de una sesión, considere aumentar el valor de esta propiedad. Esta propiedad sólo tiene relevancia si TRANSPORT tiene el valor BIND o CLIENT.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: POLLINGINT

Nombre abreviado de la herramienta de administración JMS: PINT

Acceso programático

Setters/getters

- MQConnectionFactory.setPollingIntervalo ()
- MQConnectionFactory.getPollingIntervalo ()

Valores

5.000
Este es el valor predeterminado.

Cualquier entero positivo

PORT

Para una conexión con un gestor de colas, el número del puerto en el que el gestor de colas está a la escucha o, para una conexión en tiempo real con un intermediario, el número del puerto en el que el intermediario está a la escucha de conexiones en tiempo real.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: PORT

Nombre abreviado de la herramienta de administración JMS: PORT

Acceso programático

Setters/getters

- MQConnectionFactory.setPort()
- MQConnectionFactory.getPort()

Valores

1414
Este es el valor predeterminado si TRANSPORT se establece en CLIENT.

1506
Este es el valor predeterminado si TRANSPORT se establece en DIRECT o DIRECTHTTP.

Cualquier entero positivo

PRIORITY

Prioridad de los mensajes enviados a un destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración JMS: PRIORITY

Nombre abreviado de la herramienta de administración JMS: PRI

Acceso programático

Setters/getters

- MQDestination.setPriority()

- MQDestination.getPriority()

Valores

APP
La prioridad la define la aplicación JMS. Este es el valor predeterminado.

qdef
La prioridad toma el valor predeterminado de la cola.

Cualquier entero en el rango 0-9
De menor a mayor.

PROCESSDURATION

Esta propiedad determina si un suscriptor garantiza procesar rápidamente cualquier mensaje que reciba antes de devolver el control a las clases WebSphere MQ para JMS.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración JMS: PROCESSDURATION

Nombre abreviado de la herramienta de administración JMS: PROCDUR

Acceso programático

Setters/getters

- MQConnectionFactory.setProcessDuración ()
- MQConnectionFactory.getProcessDuración ()

Valores

DESCONOCIDO
Un suscriptor no puede dar ninguna garantía sobre la rapidez con la que puede procesar cualquier mensaje que reciba. Este es el valor predeterminado.

SHORT
Un suscriptor garantiza procesar rápidamente cualquier mensaje que reciba antes de devolver el control a las clases WebSphere MQ para JMS.

PROVIDERVERSION

Esta propiedad diferencia entre las dos modalidades de operación de mensajería de WebSphere MQ : WebSphere MQ modalidad normal de proveedor de mensajería y WebSphere MQ modalidad de migración de proveedor de mensajería.

La modalidad normal del proveedor de mensajería de WebSphere MQ utiliza todas las características de los gestores de colas de WebSphere MQ Versión 7.0 para implementar JMS. Esta modalidad sólo se utiliza para conectarse a un gestor de colas de WebSphere MQ y puede conectarse a gestores de colas de WebSphere MQ Versión 7.0 en modalidad de cliente o de enlaces. Esta modalidad se optimiza para utilizar la nueva función WebSphere MQ Versión 7.0 . Si no utiliza WebSphere MQ Real-Time Transport, la modalidad de operación utilizada se determina principalmente mediante la propiedad PROVIDERVERSION de la fábrica de conexiones.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnection Factory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: PROVIDERVERSION

Nombre abreviado de la herramienta de administración JMS: PVER

Acceso programático

Setters/getters

- MQConnectionFactory.setProviderVersión ()
- MQConnectionFactory.getProviderVersión ()

Valores

Puede establecer **PROVIDERVERSION** en los valores posibles: 7, 6 o *unspecified*. Sin embargo, **PROVIDERVERSION** puede ser una cadena con uno de estos formatos:

- V.R.M.F
- V.R.M
- V.R
- V

donde V, R, M y F son valores enteros mayores que o iguales a cero.

7

Utiliza la modalidad normal del proveedor de mensajería de WebSphere MQ.

Si establece PROVIDERVERSION en 7 sólo está disponible la modalidad de operación normal del proveedor de mensajería de WebSphere MQ . Si el gestor de colas al que está conectado como resultado de los otros valores de la fábrica de conexiones no es un gestor de colas de la versión 7.0 , el método createConnection() falla con una excepción.

La modalidad normal del proveedor de mensajería de WebSphere MQ utiliza la característica de compartición de conversaciones y el número de conversaciones que se pueden compartir está controlado por la propiedad SHARECNV () en el canal de conexión de servidor. Si esta propiedad se establece en 0, no puede utilizar la modalidad normal del proveedor de mensajería de WebSphere MQ y el método createConnection() falla con una excepción.

6

Utiliza la modalidad de migración del proveedor de mensajería de WebSphere MQ.

Las clases de WebSphere MQ para JMS utilizan características y algoritmos suministrados por WebSphere MQ versión 6.0. Si desea conectarse a WebSphere Event Broker o WebSphere Message Broker utilizando WebSphere MQ Enterprise Transport, debe utilizar esta modalidad. Puede conectarse a un gestor de colas de WebSphere MQ Versión 7.0 utilizando esta modalidad, pero no se utiliza ninguna de las características nuevas de un gestor de colas de la Versión 7.0 , por ejemplo, lectura anticipada o modalidad continua.

no especificada

Este es el valor predeterminado y el texto real es "unspecified".

Una fábrica de conexiones que se ha creado con una versión anterior de WebSphere MQ Classes for JMS en JNDI adopta este valor cuando la fábrica de conexiones se utiliza con la nueva versión de WebSphere MQ Classes for JMS. El siguiente algoritmo se utiliza para determinar qué modalidad de operación se utiliza. Este algoritmo se utiliza cuando se llama al método createConnection() y utiliza otros aspectos de la fábrica de conexiones para determinar si es necesaria la modalidad normal del proveedor de mensajería de WebSphere MQ o la modalidad de migración del proveedor de mensajería de WebSphere MQ .

- En primer lugar, se realiza un intento de utilizar la modalidad normal del proveedor de mensajería de WebSphere MQ.
- Si el gestor de colas conectado no es WebSphere MQ Versión 7.0, la conexión se cierra y se utiliza en su lugar la modalidad de migración del proveedor de mensajería de WebSphere MQ .

- Si la propiedad SHARECNV () del canal de conexión de servidor se establece en 0, la conexión se cierra y se utiliza en su lugar la modalidad de migración del proveedor de mensajería de WebSphere MQ .
- Si BROKERVER se establece en 1 o en el nuevo valor predeterminado "unspecified", se sigue utilizando la modalidad normal de proveedor de mensajería de WebSphere MQ y, por lo tanto, las operaciones de publicación/suscripción utilizan las nuevas características de WebSphere MQ V7.0 . Si WebSphere Event Broker o WebSphere Message Broker se utilizan en modalidad de compatibilidad (y desea utilizar la función de publicación/suscripción de la versión 6.0 en lugar de la función de publicación/suscripción de WebSphere MQ Versión 7), establezca PROVIDERVERSION en 6 asegúrese de que se utilice la modalidad de migración del proveedor de mensajería de WebSphere MQ .

PROXYHOSTNAME

Nombre de host o dirección IP del sistema en el que se ejecuta el servidor proxy cuando se utiliza una conexión en tiempo real con un intermediario a través de un servidor proxy.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración JMS: PROXYHOSTNAME

Nombre abreviado de la herramienta de administración JMS: PHOST

Acceso programático

Setters/getters

- MQConnectionFactory.setProxyHostName()
- MQConnectionFactory.getProxyHostName()

Valores

null
Nombre de host del servidor proxy. Este es el valor predeterminado.

PROXYPORT

Número del puerto en el que el servidor proxy está a la escucha cuando se utiliza una conexión en tiempo real con un intermediario a través de un servidor proxy.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración JMS: PROXYPORT

Nombre abreviado de la herramienta de administración JMS: PPORT

Acceso programático

Setters/getters

MQConnectionFactory.setProxyPuerto ()

MQConnectionFactory.getProxyPuerto ()

Valores

443
Número de puerto del servidor proxy. Este es el valor predeterminado.

PUBACKINT

El número de mensajes publicados por un publicador antes de que las clases WebSphere MQ para JMS soliciten un acuse de recibo del intermediario.

Cuando se reduce el valor de esta propiedad, las clases WebSphere MQ para confirmaciones de solicitudes JMS con más frecuencia, por lo tanto, el rendimiento del publicador disminuye. Cuando se genera el valor, las clases WebSphere MQ para JMS tardan más tiempo en generar una excepción si el intermediario falla. Esta propiedad sólo tiene relevancia si TRANSPORT tiene el valor BIND o CLIENT.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: PROXYPORT

Nombre abreviado de la herramienta de administración JMS: PPORT

Acceso programático

Setters/getters

MQConnectionFactory.setPubAckInterval()

MQConnectionFactory.getPubAckInterval()

Valores

25
Cualquier entero positivo puede ser el valor predeterminado.

PUTASYNCALLOWED

Esta propiedad determina si se permite a los productores de mensajes utilizar operaciones de transferencia asíncrona para enviar mensajes a este destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración JMS: PUTASYNCALLOWED

Nombre abreviado de la herramienta de administración JMS: PAALD

Acceso programático

Setters/getters

MQDestination.setPutAsyncAllowed()

MQDestination.getPutAsyncAllowed()

Valores

AS_DEST
Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de cola o tema. Este es el valor predeterminado.

AS_Q_DEF
Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de cola.

AS_TOPIC_DEF
Determine si las operaciones de transferencia asíncrona están permitidas consultando la definición de tema.

No
Las operaciones de transferencia asíncrona no están permitidas.

SÍ
Las operaciones de transferencia asíncrona están permitidas.

QMANAGER

Nombre del gestor de colas al que se va a conectar.

Sin embargo, si la aplicación utiliza una tabla de definiciones de canal de cliente para conectarse a un gestor de colas, consulte Utilización de una tabla de definiciones de canal de cliente con clases de WebSphere MQ para JMS.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, XAConnectionFactory, XAQueueConnection Factory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: QMANAGER

Nombre abreviado de la herramienta de administración JMS: QMGR

Acceso programático

Setters/getters

- MQConnectionFactory.setQueueGestor ()
- MQConnectionFactory.getQueueGestor ()

Valores

"" (serie vacía)
Cualquier serie puede ser el valor predeterminado.

COLA

El nombre del destino de cola JMS. Coincide con el nombre de la cola utilizada por el gestor de colas.

Objetos aplicables

Cola

Nombre largo de la herramienta de administración JMS: QUEUE

Nombre abreviado de la herramienta de administración JMS: QU

Valores

Cualquier serie
Cualquier nombre de cola de IBM WebSphere MQ válido.

Conceptos relacionados
Reglas de denominación de objetos de IBM WebSphere MQ

READAHEADALLOWED

Esta propiedad determina si los consumidores de mensajes y los navegadores de colas pueden utilizar la lectura anticipada para obtener mensajes no persistentes de este destino en un almacenamiento intermedio interno antes de recibirlos.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración JMS: READAHEADALLOWED

Nombre abreviado de la herramienta de administración JMS: RAALD

Acceso programático

Setters/getters

- MQDestination.setReadAheadAllowed()
- MQDestination.getReadAheadAllowed()

Valores

AS_DEST
Determine si se permite la lectura anticipada consultando la definición de cola o tema. Este es el valor predeterminado en las herramientas administrativas.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_DEST en programas.

AS_Q_DEF
Determine si se permite la lectura anticipada consultando la definición de cola.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF en programas.

AS_TOPIC_DEF
Determine si se permite la lectura anticipada consultando la definición de tema.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_TOPIC_DEF en los programas.

No
No se permite la lectura anticipada.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_DISABLED en los programas.

SÍ
La lectura anticipada está permitida.

Utilice WMQConstants.WMQ_READ_AHEAD_ALLOWED_ENABLED en los programas.

READAHEADCLOSEPOLICY

Para los mensajes que se entregan a un escucha de mensajes asíncrono, qué sucede con los mensajes del almacenamiento intermedio de lectura anticipada interno cuando se cierra el consumidor de mensajes.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración JMS: READAHEADCLOSEPOLICY

Nombre abreviado de la herramienta de administración JMS: RACP

Acceso programático

Setters/getters

- MQDestination.setReadAheadClosePolicy()
- MQDestination.getReadAheadClosePolicy()

Valores

DELIVER_ALL
Todos los mensajes del almacenamiento intermedio de lectura anticipada interno se entregan al escucha de mensajes de la aplicación antes de devolverlos. Este es el valor predeterminado en las herramientas administrativas.

Utilice WMQConstants.WMQ_READ_AHEAD_DELIVERALL en programas.

ACTUAL_ENTREGA
Solo se completa la invocación del escucha de mensajes actual antes de la devolución, posiblemente dejando mensajes en el almacenamiento intermedio de lectura anticipada interno que, después, se descartan.

Utilice WMQConstants.WMQ_READ_AHEAD_DELIVERCURRENT en programas.

RECEIVECCSID

La propiedad de destino que define el CCSID de destino para la conversión de mensajes del gestor de colas. El valor se ignora a menos que RECEPECONVERSION se establezca en `WMQ_RECEIVE_CONVERSION_QMGR`

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración JMS: RECEIVECCSID

Nombre abreviado de la herramienta de administración JMS: RCCS

Acceso programático

Métodos setter/Getters

- `MQDestination.setReceiveCCSID`
- `MQDestination.getReceiveCCSID`

Valores

WMQConstants.WMQ_RECEIVE_CCSID_JVM_DEFAULT
0 - Utilizar `Charset.defaultCharset` de JVM

1208
UTF-8

ccsid
Identificador de juego de caracteres codificado soportado.

RECEIVECONVERSION

Propiedad de destino que determina si el gestor de colas va a realizar la conversión de datos.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración JMS: RECEPECONVERSION

Nombre abreviado de la herramienta de administración JMS: RCNV

Acceso programático

Métodos setter/Getters

- `MQDestination.setReceiveConversion`
- `MQDestination.getReceiveConversion`

Valores

`WMQConstants.WMQ_RECEIVE_CONVERSION_CLIENT_MSG`
1 -Sólo realizar la conversión de datos en el cliente JMS. El valor predeterminado de hasta V7.0, y de, e incluyendo, 7.0.1.5.

`WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR`
2 -Realizar la conversión de datos en el gestor de colas antes de enviar un mensaje al cliente. El valor predeterminado (y solo) de V7.0 a V7.0.1.4 inclusive, excepto si se aplica el APAR IC72897 .

RECEIVEISOLATION

Esta propiedad determina si un suscriptor puede recibir mensajes que no se han confirmado en la cola de suscriptores.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración JMS: RECEIVEISOLATION

Nombre abreviado de la herramienta de administración JMS: RCVISOL

Valores

CONFIRMADO
Un suscriptor sólo recibe los mensajes de la cola de suscriptores que se han confirmado. Este es el valor predeterminado en las herramientas administrativas.

Utilice WMQConstants.WMQ_RCVISOL_COMMITTED en los programas.

SIN CONFIRMAR
Un suscriptor puede recibir mensajes que no se han confirmado en la cola de suscriptor.

Utilice WMQConstants.WMQ_RCVISOL_UNCOMMITTED en programas.

RECEXIT

Identifica una salida de recepción de canal o una secuencia de salidas de recepción que se van a ejecutar sucesivamente.

Es posible que sea necesaria una configuración adicional para que IBM WebSphere MQ classes for JMS localice las salidas de recepción. Para obtener más información, consulte Configuración de las clases de IBM WebSphere MQ para JMS para utilizar salidas de canal.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: RECEXIT

Nombre abreviado de la herramienta de administración JMS: RCX

Acceso programático

Setters/getters

- MQConnectionFactory.setReceiveSalir ()
- MQConnectionFactory.getReceiveSalir ()

Valores

null
Una serie que consta de uno o más elementos separados por comas, donde cada elemento es:

- El nombre de una clase que implementa la interfaz WMQReceiveExit (para una salida de recepción de canal escrita en Java).
- Una serie con el formato *libraryName(entryPointNombre)* (para una salida de recepción de canal no escrita en Java).

Este es el valor predeterminado.

RECEXITINIT

Los datos de usuario que se pasan a las salidas de recepción de canal cuando se llaman.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: RECEXITINIT

Nombre abreviado de la herramienta de administración JMS: RCXI

Acceso programático

Setters/getters

- MQConnectionFactory.setReceiveExitInit()
- MQConnectionFactory.getReceiveExitInit()

Valores

null
Una cadena que comprende uno o más elementos de datos de usuario separados por comas. Este es el valor predeterminado.

REPLYTOSTYLE

Determina cómo se construye el campo JMSReplyTo en un mensaje recibido.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: REPLYTOSTYLE

Nombre abreviado de la herramienta de administración JMS: RTOST

Acceso programático

Setters/getters

- MQConnectionFactory.setReplyToStyle()
- MQConnectionFactory.getReplyToStyle()

Valores

DEFAULT
Equivale a MQMD.

RFH2
Utilice el valor proporcionado en la cabecera RFH2 . Si se ha establecido un valor JMSReplyTo en la aplicación emisora, utilice ese valor.

MQMD
Utilice el valor proporcionado por MQMD. Este comportamiento es equivalente al comportamiento predeterminado de WebSphere MQ Versión 6.0.2.4 y 6.0.2.5.

Si el valor JMSReplyTo establecido por la aplicación emisora no contiene un nombre de gestor de colas, el gestor de colas receptor inserta su propio nombre en el MQMD. Si establece este parámetro en MQMD, la cola de respuesta que utiliza se encuentra en el gestor de colas receptor. Si establece este parámetro en RFH2, la cola de respuesta que utiliza se encuentra en el gestor de colas especificado en la RFH2 del mensaje enviado, tal como lo estableció originalmente la aplicación emisora.

Si el valor JMSReplyTo establecido por la aplicación emisora contiene un nombre de gestor de colas, el valor de este parámetro no es importante porque tanto MQMD como RFH2 contienen el mismo valor.

RESCANINT

Cuando un consumidor de mensajes del dominio punto a punto utiliza un selector de mensajes para seleccionar qué mensajes desea recibir, las clases WebSphere MQ para JMS buscan en la cola WebSphere MQ los mensajes adecuados en la secuencia determinada por el atributo MsgDeliverySequence de la cola.

Después de que WebSphere MQ classes for JMS encuentre un mensaje adecuado y lo entregue al consumidor, WebSphere MQ classes for JMS reanuda la búsqueda del siguiente mensaje adecuado desde su posición actual en la cola. WebSphere MQ classes for JMS continúa buscando en la cola de esta forma hasta que alcanza el final de la cola, o hasta que el intervalo de tiempo en milisegundos, determinado por el valor de esta propiedad, ha caducado. En cada caso, WebSphere MQ classes for JMS vuelve al principio de la cola para continuar la búsqueda y comienza un nuevo intervalo de tiempo.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nombre largo de la herramienta de administración JMS: RESCANINT

Nombre abreviado de la herramienta de administración JMS: RINT

Acceso programático

Setters/getters

- MQConnectionFactory.setRescanIntervalo ()
- MQConnectionFactory.getRescanIntervalo ()

Valores

5.000
Cualquier entero positivo puede ser el valor predeterminado.

SECEXIT

Identifica una salida de seguridad de canal.

Es posible que sea necesaria una configuración adicional para que IBM WebSphere MQ classes for JMS localice las salidas de seguridad. Para obtener más información, consulte Configuración de las clases de IBM WebSphere MQ para JMS para utilizar salidas de canal.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: SECEXIT

Nombre abreviado de la herramienta de administración JMS: SXC

Acceso programático

Setters/getters

- MQConnectionFactory.setSecuritySalir ()
- MQConnectionFactory.getSecurityExit ()

Valores

null
El nombre de una clase que implementa la interfaz WMQSecurityExit (para una salida de seguridad de canal escrita en Java).

Una serie con el formato *libraryName(entryPointNombre)* (para una salida de seguridad de canal no escrita en Java).

SECEXITINIT

Los datos de usuario que se pasan a una salida de seguridad de canal cuando se invoca.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: SECEXITINIT

Nombre abreviado de la herramienta de administración JMS: SCXI

Acceso programático

Setters/getters

- MQConnectionFactory.setSecurityExitInit()
- MQConnectionFactory.getSecurityExitInit()

Valores

null
Cualquier serie puede ser el valor predeterminado.

SENDCHECKCOUNT

El número de llamadas de envío que se permiten entre la comprobación de errores de colocación asíncrona, dentro de una única sesión JMS sin transacción.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: SENDCHECKCOUNT

Nombre abreviado de la herramienta de administración JMS: SCC

Acceso programático

Setters/getters

- MQConnectionFactory.setSendCheckCount()
- MQConnectionFactory.getSendCheckCount()

Valores

null
Cualquier serie puede ser el valor predeterminado.

SENDEXIT

Identifica una salida de envío de canal o una secuencia de salidas de envío que se van a ejecutar sucesivamente.

Es posible que sea necesaria una configuración adicional para que IBM WebSphere MQ classes for JMS localice las salidas de envío. Para obtener más información, consulte Configuración de las clases de IBM WebSphere MQ para JMS para utilizar salidas de canal.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: SENDEXIT

Nombre abreviado de la herramienta de administración JMS: SDX

Acceso programático

Setters/getters

- MQConnectionFactory.setSendSalir ()
- MQConnectionFactory.getSendSalir ()

Valores

null
Cualquier serie que comprenda uno o más elementos separados por comas, donde cada elemento sea:

- El nombre de una clase que implementa la interfaz WMQSendExit (para una salida de envío de canal escrita en Java).
- Una serie con el formato *libraryName(entryPointNombre)* (para una salida de emisión de canal no escrita en Java).
-

Este es el valor predeterminado.

SENDEXITINIT

Los datos de usuario que se pasan a las salidas de emisión de canal cuando se invocan.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: SENDEXITINIT

Nombre abreviado de la herramienta de administración JMS: SDXI

Acceso programático

Setters/getters

- MQConnectionFactory.setSendExitInit()
- MQConnectionFactory.getSendExitInit()

Valores

null
Cualquier serie que comprenda uno o más elementos de datos de usuario separados por comas puede ser el valor predeterminado.

SHARECONVALLOWED

Esta propiedad determina si una conexión de cliente puede compartir su socket con otras conexiones JMS de nivel superior del mismo proceso con el mismo gestor de colas, si las definiciones de canal coinciden.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: SHARECONVALLOWED

Nombre abreviado de la herramienta de administración JMS: SCALD

Acceso programático

Setters/getters

- MQConnectionFactory.setShareConvAllowed()
- MQConnectionFactory.getShareConvAllowed()

Valores

SÍ
Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice WMQConstants.WMQ_SHARE_CONV_ALLOWED_YES.

No
Este valor es para herramientas administrativas.

Para los programas, utilice WMQConstants.WMQ_SHARE_CONV_ALLOWED_NO.

SPARSESUBS

Controla la política de recuperación de mensajes de un objeto TopicSubscriber.

Objetos aplicables

ConnectionFactory, Fábrica TopicConnection

Nombre largo de la herramienta de administración JMS: SPARSESUBS

Nombre abreviado de la herramienta de administración JMS: SSUBS

Acceso programático

Setters/getters

- MQConnectionFactory.setSparseSuscripciones ()
- MQConnectionFactory.getSparseSuscripciones ()

Valores

No
Las suscripciones reciben mensajes coincidentes frecuentes. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice false.

SÍ
Las suscripciones reciben mensajes coincidentes poco frecuentes. Este valor exige que la cola de suscripciones se pueda abrir para examinarla.

Para los programas, utilice true.

SSLCIPHERSUITE

La CipherSuite que se debe utilizar para una conexión SSL.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: SSLCIPHERSUITE

Nombre abreviado de la herramienta de administración JMS: SCPHS

Acceso programático

Setters/getters

- MQConnectionFactory.setSSLCipherSuite ()
- MQConnectionFactory.getSSLCipherSuite ()

Valores

null
Este es el valor predeterminado. Para obtener más información, consulte Propiedades SSL de objetos JMS.

SSLCRL

Servidores CRL para comprobar la revocación de certificados SSL.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: SSLCRL

Nombre abreviado de la herramienta de administración JMS: SCRL

Acceso programático

Setters/getters

- MQConnectionFactory.setSSLCertAlmacenes ()
- MQConnectionFactory.getSSLCertAlmacenes ()

Valores

null
Lista separada por espacios de URL de LDAP. Este es el valor predeterminado. Para obtener más información, consulte Propiedades SSL de objetos JMS.

SSLFIPSREQUIRED

Esta propiedad determina si una conexión SSL debe utilizar una CipherSuite soportada por el proveedor IBM Java JSSE FIPS (IBMJSSEFIPS).

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: SSLFIPSREQUIRED

Nombre abreviado de la herramienta de administración JMS: SFIPS

Acceso programático

Setters/getters

- MQConnectionFactory.setSSLFipsObligatorio ()
- MQConnectionFactory.getSSLFipsObligatorio ()

Valores

No
Una conexión SSL puede utilizar cualquier CipherSuite que no esté soportada por el proveedor IBM Java JSSE FIPS (IBMJSSEFIPS).

Este es el valor predeterminado. En los programas, utilice false.

SÍ
Una conexión SSL debe utilizar una CipherSuite soportada por IBMJSSEFIPS.

En los programas, utilice true.

SSLPEERNAME

Para SSL, un esqueleto de *nombre distinguido* que debe coincidir con el proporcionado por el gestor de colas.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: SSLPEERNAME

Nombre abreviado de la herramienta de administración JMS: SPEER

Acceso programático

Setters/getters

- MQConnectionFactory.setSSLPeerNombre ()

- MQConnectionFactory.getSSLPeerNombre ()

Valores

null
Este es el valor predeterminado. Para obtener más información, consulte Propiedades SSL de objetos JMS.

SSLRESETCOUNT

Para SSL, el número total de bytes enviados y recibidos por una conexión antes de que se renegocie la clave secreta que se utiliza para el cifrado.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: SSLRESETCOUNT

Nombre abreviado de la herramienta de administración JMS: SRC

Acceso programático

Setters/getters

- MQConnectionFactory.setSSLResetRecuento ()
- MQConnectionFactory.getSSLResetRecuento ()

Valores

0
Cero, o cualquier entero positivo menor o igual que 999, 999, 999. Este es el valor predeterminado. Para obtener más información, consulte Propiedades SSL de objetos JMS.

STATREFRESHINT

Intervalo, en milisegundos, entre las actualizaciones de la transacción de larga ejecución que detecta cuando un suscriptor pierde su conexión con el gestor de colas.

Esta propiedad sólo es relevante si SUBSTORE tiene el valor QUEUE.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: STATREFRESHINT

Nombre abreviado de la herramienta de administración JMS: SRI

Acceso programático

Setters/getters

- MQConnectionFactory.setStatusRefreshInterval()
- MQConnectionFactory.getStatusRefreshInterval()

Valores

6000
Cualquier entero positivo puede ser el valor predeterminado. Para obtener más información, consulte Propiedades SSL de objetos JMS.

SUBSTORE

Donde WebSphere MQ classes for JMS almacena datos persistentes relacionados con suscripciones activas.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: SUBSTORE

Nombre abreviado de la herramienta de administración JMS: SS

Acceso programático

Setters/getters

- MQConnectionFactory.setSubscriptionAlmacén ()
- MQConnectionFactory.getSubscriptionStore ()

Valores

BROKER
Utilice el almacén de suscripciones basado en intermediario para mantener los detalles de las suscripciones. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice WMQConstants.WMQ_SUBSTORE_BROKER.

MIGRATE
Transfiera la información de suscripción del almacén de suscripciones basado en cola al almacén de suscripciones basado en intermediario.

Para los programas, utilice WMQConstants.WMQ_SUBSTORE_MIGRATE.

COLA
Utilice el almacén de suscripciones basado en cola para contener los detalles de las suscripciones.

Para los programas, utilice WMQConstants.WMQ_SUBSTORE_QUEUE.

SYNCPOINTALLGETS

Esta propiedad determina si todas las obtenciones se deben realizar bajo punto de sincronismo.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: SYNCPOINTALLGETS

Nombre abreviado de la herramienta de administración JMS: SPAG

Acceso programático

Setters/getters

- MQConnectionFactory.setSyncpointAllGets()
- MQConnectionFactory.getSyncpointAllGets()

Valores

No
Este es el valor predeterminado.

Sí

TARGCLIENT

Esta propiedad determina si el formato WebSphere MQ RFH2 se utiliza para intercambiar información con aplicaciones de destino.

Objetos aplicables

Cola, Tema

Nombre largo de la herramienta de administración JMS: TARGCLIENT

Nombre abreviado de la herramienta de administración JMS: TC

Acceso programático

Setters/getters

- MQDestination.setTargetClient()
- MQDestination.getTargetClient()

Valores

JMS
El destino del mensaje es una aplicación JMS. Este es el valor predeterminado para las herramientas administrativas.

Para programas, utilice WMQConstants.WMQ_CLIENT_JMS_COMPLIANT.

MQ
El destino del mensaje es una aplicación WebSphere MQ no JMS.

Para programas, utilice WMQConstants.WMQ_CLIENT_NONJMS_MQ.

TARGCLIENTMATCHING

Esta propiedad determina si un mensaje de respuesta, enviado a la cola identificada por el campo de cabecera JMSReplyTo de un mensaje de entrada, tiene una cabecera MQRFH2 sólo si el mensaje de entrada tiene una cabecera MQRFH2 .

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nombre largo de la herramienta de administración JMS: TARGCLIENTMATCHING

Nombre abreviado de la herramienta de administración JMS: TCM

Acceso programático

Setters/getters

- MQConnectionFactory.setTargetClientMatching()
- MQConnectionFactory.getTargetClientMatching()

Valores

sí

Si un mensaje de entrada no tiene una cabecera MQRFH2 , la propiedad TARGCLIENT del objeto Queue derivado del campo de cabecera JMSReplyTo del mensaje se envía a MQ. Si el mensaje tiene una cabecera MQRFH2 , la propiedad TARGCLIENT se establece en JMS en su lugar. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice true.

No

La propiedad TARGCLIENT del objeto Queue derivada del campo de cabecera JMSReplyTo de un mensaje de entrada siempre se establece en JMS.

Para los programas, utilice false.

TEMPMODEL

El nombre de la cola modelo a partir de la cual se crean las colas temporales JMS.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nombre largo de la herramienta de administración JMS: TEMPMODEL

Nombre abreviado de la herramienta de administración JMS: TM

Acceso programático

Setters/getters

- MQConnectionFactory.setTemporaryModelo ()
- MQConnectionFactory.getTemporaryModelo ()

Valores

SYSTEM.DEFAULT.MODEL.QUEUE

Cualquier serie puede ser el valor predeterminado.

TEMPQPREFIX

El prefijo que se utiliza para formar el nombre de una cola dinámica de WebSphere MQ .

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Nombre largo de la herramienta de administración JMS: TEMPQPREFIX

Nombre abreviado de la herramienta de administración JMS: TQP

Acceso programático

Setters/getters

- MQConnectionFactory.setTempQPrefix ()
- MQConnectionFactory.getTempQPrefix ()

Valores

'' '' (serie vacía)
El prefijo utilizado es `CSQ.*` en z/OS y `AMQ.*` en todas las demás plataformas. Estos son los valores predeterminados.

Prefijo de cola
El prefijo de cola es cualquier serie que se ajuste a las reglas para formar contenido del campo *DynamicQName* en un descriptor de objeto WebSphere MQ (estructura MQOD), pero el último carácter no en blanco debe ser un asterisco.

TEMPTOPICPREFIX

Al crear temas temporales, JMS genera una serie de tema con el formato '' TEMP/*TEMPTOPICPREFIX/unique_id*'', o si esta propiedad se deja con el valor predeterminado, sólo '' TEMP/*unique_id*''. Especificar un TEMPTOPICPREFIX no vacío permite definir colas de modelo específicas para crear las colas gestionadas para suscriptores a temas temporales creados bajo esta conexión.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: TEMPTOPICPREFIX

Nombre abreviado de la herramienta de administración JMS: TTP

Acceso programático

Setters/getters

- MQConnectionFactory.setTempTopicPrefix()
- MQConnectionFactory.getTempTopicPrefix()

Valores

Cualquier serie no nula que conste sólo de caracteres válidos para una serie de tema de WebSphere MQ . El valor predeterminado es '' '' (serie vacía).

Tema

El nombre del destino de tema JMS, este valor lo utiliza el gestor de colas como serie de tema de una publicación o suscripción.

Objetos aplicables

Tema

Nombre largo de la herramienta de administración JMS: TOPIC

Nombre abreviado de la herramienta de administración JMS: TOP

Valores

Cualquier serie
Una serie que forma una serie de tema IBM WebSphere MQ válida. Cuando utilice IBM WebSphere MQ como proveedor de mensajería con WebSphere Application Server, especifique un valor que coincida con el nombre por el que se conoce el tema para fines administrativos en WebSphere Application Server.

Conceptos relacionados

Series de tema

TRANSPORT

La naturaleza de una conexión con un gestor de colas o intermediario.

Objetos aplicables

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: TRANSPORT

Nombre abreviado de la herramienta de administración JMS: TRAN

Acceso programático

Setters/getters

- MQConnectionFactory.setTransportTipo ()
- MQConnectionFactory.getTransportTipo ()

Valores

BIND
Para una conexión con un gestor de colas en modalidad de enlaces. Este es el valor predeterminado para las herramientas administrativas.

Para programas, utilice WMQConstants.WMQ_CM_BINDINGS.

CLIENT
Para una conexión con un gestor de colas en modalidad de cliente.

Para programas, utilice WMQConstants.WMQ_CM_CLIENT.

DIRECT
Para una conexión en tiempo real con un intermediario que no utiliza el túnel HTTP.

Para los programas, utilice WMQConstants.WMQ_CM_DIRECT_TCPIP.

DIRECTHTTP
Para una conexión en tiempo real a un intermediario utilizando el túnel HTTP. Solo se da soporte a HTTP 1.0 .

Para los programas, utilice WMQConstants.WMQ_CM_DIRECT_HTTP.

WILDCARDFORMAT

Esta propiedad determina qué versión de sintaxis de comodín se va a utilizar.

Objetos aplicables

ConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XATopicConnectionFactory

Nombre largo de la herramienta de administración JMS: WILDCARDFORMAT

Nombre abreviado de la herramienta de administración JMS: WCFMT

Acceso programático

Setters/getters

- MQConnectionFactory.setWildCardFormat()
- MQConnectionFactory.getWildCardFormat()

Valores

TOPIC_ONLY
Sólo reconoce comodines de nivel de tema, tal como se utiliza en la versión 2 del intermediario. Este es el valor predeterminado para las herramientas administrativas.

Para los programas, utilice WMQConstants.WMQ_WILDCARD_TOPIC_ONLY.

CAR_SOLO
Sólo reconoce caracteres comodín, tal como se utiliza en la versión 1 del intermediario.

Para los programas, utilice WMQConstants.WMQ_WILDCARD_CHAR_ONLY.

Dependencias entre propiedades de WebSphere MQ clases para objetos JMS

La validez de algunas propiedades depende de los valores particulares de otras propiedades.

Esta dependencia puede producirse en los siguientes grupos de propiedades:

- Propiedades del cliente
- Propiedades para una conexión en tiempo real con un intermediario
- Series de inicialización de salida

Propiedades del cliente
Para una conexión con un gestor de colas, las propiedades siguientes sólo son relevantes si TRANSPORT tiene el valor CLIENT:

- HOSTNAME
- PORT
- CHANNEL
- LOCALADDRESS
- CCDTURL
- CCSID
- COMPHDR
- COMPMSG
- RECEXIT
- RECEXITINIT
- SECEXIT
- SECEXITINIT
- SENDEXIT
- SENDEXITINIT
- SHARECONVALLOWED
- SSLCIPHERSUITE
- SSLCRL
- SSLFIPSREQUIRED
- SSLPEERNAME
- SSLRESETCOUNT
- APPLICATIONNAME

No puede establecer valores para estas propiedades utilizando la herramienta de administración si TRANSPORT tiene el valor BIND.

Si TRANSPORT tiene el valor CLIENT, el valor predeterminado de la propiedad BROKERVER es V1 y el valor predeterminado de la propiedad PORT es 1414. Si establece el valor de BROKERVER o

PORT explícitamente, un cambio posterior en el valor de TRANSPORT no altera temporalmente las opciones.

Propiedades para una conexión en tiempo real con un intermediario
Solo son relevantes las propiedades siguientes si TRANSPORT tiene el valor DIRECT o DIRECTHTTP:

- BROKERVER
- CLIENTID
- description
- DIRECTAUTH
- HOSTNAME
- LOCALADDRESS
- MAXBUFFSIZE
- MULTICAST (soportado solo para DIRECT)
- PORT
- PROXYHOSTNAME (soportado sólo para DIRECT)
- PROXYPORT (soportado sólo para DIRECT)

Si TRANSPORT tiene el valor DIRECT o DIRECTHTTP, el valor predeterminado de la propiedad BROKERVER es V2y el valor predeterminado de la propiedad PORT es 1506. Si establece el valor de BROKERVER o PORT explícitamente, un cambio posterior en el valor de TRANSPORT no altera temporalmente las opciones.

Series de inicialización de salida
No establezca ninguna de las series de inicialización de salida sin proporcionar el nombre de salida correspondiente. Las propiedades de inicialización de salida son:

- RECEXITINIT
- SECEXITINIT
- SENDEXITINIT

Por ejemplo, si especifica `RECEXITINIT(myString)` sin especificar `RECEXIT(some.exit.classname)` , se produce un error.

La propiedad ENCODING

La propiedad ENCODING comprende tres subpropiedades, en doce combinaciones posibles.

Los valores válidos que puede tomar la propiedad `ENCODING` se construyen a partir de las tres subpropiedades:

Codificación de enteros
Normal o invertido

Codificación decimal
Normal o invertido

codificación de coma flotante
IEEE normal, IEEE invertido o z/OS

La propiedad `ENCODING` se expresa como una serie de tres caracteres con la sintaxis siguiente:

```
{N|R}{N|R}{N|R|3}
```

En esta serie:

- N indica normal
- R indica invertido
- 3 denota z/OS

- El primer carácter representa *codificación de enteros*
- El segundo carácter representa la *codificación decimal*
- El tercer carácter representa la *codificación de coma flotante*

Esto proporciona un conjunto de doce valores posibles para la propiedad ENCODING .

Hay un valor adicional, la serie NATIVE, que establece los valores de codificación adecuados para la plataforma Java.

Los ejemplos siguientes muestran combinaciones válidas para ENCODING:

```
ENCODING(NNR)
ENCODING(NATIVE)
ENCODING(RR3)
```

Propiedades SSL de los objetos JMS

Habilite el cifrado SSL (Secure Sockets Layer) utilizando la propiedad SSLCIPHERSUITE. A continuación, puede cambiar las características del cifrado SSL utilizando otras propiedades.

Cuando especifique TRANSPORT (CLIENT), puede habilitar la comunicación cifrada SSL (Secure Sockets Layer) utilizando la propiedad SSLCIPHERSUITE. Establezca esta propiedad en una CipherSuite válida proporcionada por el proveedor JSSE; debe coincidir con la CipherSpec especificada en el canal SVRCONN especificado por la propiedad CHANNEL.

Sin embargo, las CipherSpecs (tal como se especifica en el canal SVRCONN) y las CipherSuites (tal como se especifica en los objetos ConnectionFactory) utilizan distintos esquemas de denominación para representar los mismos algoritmos de cifrado SSL. Si se especifica un nombre CipherSpec reconocido en la propiedad SSLCIPHERSUITE, JMSAdmin emite un aviso y correlaciona la CipherSpec con su CipherSuiteequivalente. Consulte SSL CipherSpecs y CipherSuites en JMS para obtener una lista de CipherSpecs reconocidas por WebSphere MQ y JMSAdmin.

Si necesita una conexión para utilizar una CipherSuite soportada por el proveedor IBM Java JSSE FIPS (IBMJSSEFIPS), establezca la propiedad SSLFIPSREQUIRED de la fábrica de conexiones en YES. El valor predeterminado de esta propiedad es NO, lo que significa que una conexión puede utilizar cualquier CipherSuitesoportada. La propiedad se ignora si no se ha establecido SSLCIPHERSUITE.

El SSLPEERNAME coincide con el formato del parámetro SSLPEER, que se puede establecer en definiciones de canal. Es una lista de pares de nombre y valor de atributo separados por comas o signos de punto y coma. Por ejemplo:

```
SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

El conjunto de nombres y valores forma un *nombre distinguido*. Para obtener más detalles sobre los nombres distinguidos y su uso con WebSphere MQ, consulte Seguridad.

El ejemplo proporcionado comprueba el certificado de identificación presentado por el servidor durante la conexión. Para que la conexión sea satisfactoria, el certificado debe tener un nombre común que empiece por QMGR., y debe tener al menos dos nombres de unidad organizativa, el primero de los cuales es IBM y el segundo WEBSPHERE. La comprobación no distingue entre mayúsculas y minúsculas.

Si SSLPEERNAME no está establecido, no se realiza ninguna comprobación de este tipo. SSLPEERNAME se ignora si no se ha establecido SSLCIPHERSUITE.

La propiedad SSLCRL especifica cero o más servidores CRL (Lista de revocación de certificados). El uso de esta propiedad requiere una JVM en Java 2 v1.4. Esta es una lista delimitada por espacios de entradas con el formato:

```
ldap://hostname:[port]
```

seguido opcionalmente por un único/. Si se omite *port* , se presupone el puerto LDAP predeterminado de 389. En el momento de la conexión, el certificado SSL presentado por el servidor se compara con los servidores de CRL especificados. Consulte Seguridad para obtener más información sobre la seguridad de CRL.

Si SSLCRL no está establecido, no se realiza ninguna comprobación de este tipo. SSLCRL se ignora si no se ha establecido SSLCIPHERSUITE.

La propiedad SSLRESETCOUNT representa el número total de bytes enviados y recibidos por una conexión antes de que se renegocie la clave secreta que se utiliza para el cifrado. El número de bytes enviados es el número antes del cifrado y el número de bytes recibidos es el número después del cifrado. El número de bytes también incluye la información de control enviada y recibida por las clases de WebSphere MQ para JMS.

Por ejemplo, para configurar un objeto ConnectionFactory que se puede utilizar para crear una conexión a través de un canal MQI habilitado para SSL con una clave secreta que se renegocia después de que hayan fluido 4 MB de datos, emita el mandato siguiente a JMSAdmin:

```
ALTER CF(my.cf) SSLRESETCOUNT(4194304)
```

Si el valor de SSLRESETCOUNT es cero, que es el valor predeterminado, la clave secreta nunca se renegocia. La propiedad SSLRESETCOUNT se ignora si SSLCIPHERSUITE no está establecido.

Avisos

Esta información se ha desarrollado para productos y servicios ofrecidos en los Estados Unidos.

Es posible que IBM no ofrezca los productos, servicios o las características que se tratan en este documento en otros países. Consulte al representante local de IBM para obtener información sobre los productos y servicios disponibles actualmente en su zona. Las referencias a programas, productos o servicios de IBM no pretenden indicar ni implicar que sólo puedan utilizarse los productos, programas o servicios de IBM. En su lugar podrá utilizarse cualquier producto, programa o servicio equivalente que no infrinja ninguno de los derechos de propiedad intelectual de IBM. No obstante, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio no IBM.

IBM puede tener patentes o solicitudes de patentes pendientes que cubran el tema principal descrito en este documento. El suministro de este documento no le otorga ninguna licencia sobre estas patentes. Puede enviar consultas sobre licencias, por escrito, a:

IBM Director
of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Para consultas sobre licencias relacionadas con información de doble byte (DBCS), póngase en contacto con el Departamento de propiedad intelectual de IBM de su país o envíe las consultas por escrito a:

Licencias de Propiedad Intelectual
Ley de Propiedad intelectual y legal
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokio 103-8510, Japón

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país donde estas disposiciones contradigan la legislación vigente: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN NINGÚN TIPO DE GARANTÍA, YA SEA EXPLÍCITA O IMPLÍCITA, INCLUYENDO, PERO SIN LIMITARSE A, LAS GARANTÍAS IMPLÍCITAS DE NO INCUMPLIMIENTO, COMERCIABILIDAD O IDONEIDAD PARA UNA FINALIDAD DETERMINADA. Algunas legislaciones no contemplan la exclusión de garantías, ni implícitas ni explícitas, en determinadas transacciones, por lo que puede haber usuarios a los que no les afecte dicha norma.

Esta información puede contener imprecisiones técnicas o errores tipográficos. La información aquí contenida está sometida a cambios periódicos; tales cambios se irán incorporando en nuevas ediciones de la publicación. IBM puede efectuar mejoras y/o cambios en los productos y/o programas descritos en esta publicación en cualquier momento y sin previo aviso.

Cualquier referencia en esta información a sitios web que no son de IBM se realiza por razones prácticas y de ninguna manera sirve como un respaldo de dichos sitios web. Los materiales de dichos sitios web no forman parte de este producto de IBM y la utilización de los mismos será por cuenta y riesgo del usuario.

IBM puede utilizar o distribuir cualquier información que el usuario le proporcione del modo que considere apropiado sin incurrir por ello en ninguna obligación con respeto al usuario.

Los titulares de licencias de este programa que deseen información del mismo con el fin de permitir: (i) el intercambio de información entre los programas creados de forma independiente y otros programas (incluido este) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N

Rochester, MN 55901
U.S.A.

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluyendo, en algunos casos, el pago de una cantidad.

El programa bajo licencia que se describe en esta información y todo el material bajo licencia disponible para el mismo lo proporciona IBM bajo los términos del Acuerdo de cliente de IBM, el Acuerdo de licencia de programas internacional de IBM o cualquier acuerdo equivalente entre las partes.

Los datos de rendimiento incluidos en este documento se han obtenido en un entorno controlado. Por consiguiente, los resultados obtenidos en otros entornos operativos pueden variar de manera significativa. Es posible que algunas mediciones se hayan realizado en sistemas en nivel de desarrollo y no existe ninguna garantía de que estas mediciones serán las mismas en sistemas disponibles generalmente. Además, algunas mediciones pueden haberse estimado por extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables a su entorno específico.

La información relativa a productos que no son de IBM se obtuvo de los proveedores de esos productos, sus anuncios publicados u otras fuentes de disponibilidad pública. IBM no ha comprobado estos productos y no puede confirmar la precisión de su rendimiento, compatibilidad o alguna reclamación relacionada con productos que no sean de IBM. Las preguntas relacionadas con las posibilidades de los productos que no sean de IBM deben dirigirse a los proveedores de dichos productos.

Todas las declaraciones relacionadas con una futura intención o tendencia de IBM están sujetas a cambios o se pueden retirar sin previo aviso y sólo representan metas y objetivos.

Este documento contiene ejemplos de datos e informes que se utilizan diariamente en la actividad de la empresa. Para ilustrar los ejemplos de la forma más completa posible, éstos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con los nombres y direcciones utilizados por una empresa real es puramente casual.

LICENCIA DE COPYRIGHT:

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente que ilustran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de cualquier forma sin pagar ninguna cuota a IBM para fines de desarrollo, uso, marketing o distribución de programas de aplicación que se ajusten a la interfaz de programación de aplicaciones para la plataforma operativa para la que se han escrito los programas de ejemplo. Los ejemplos no se han probado minuciosamente bajo todas las condiciones. IBM, por tanto, no puede garantizar la fiabilidad, servicio o funciones de estos programas.

Puede que si visualiza esta información en copia software, las fotografías e ilustraciones a color no aparezcan.

Información acerca de las interfaces de programación

La información de interfaz de programación, si se proporciona, está pensada para ayudarle a crear software de aplicación para su uso con este programa.

Este manual contiene información sobre las interfaces de programación previstas que permiten al cliente escribir programas para obtener los servicios de IBM WebSphere MQ.

Sin embargo, esta información puede contener también información de diagnóstico, modificación y ajustes. La información de diagnóstico, modificación y ajustes se proporciona para ayudarle a depurar el software de aplicación.

Importante: No utilice esta información de diagnóstico, modificación y ajuste como interfaz de programación porque está sujeta a cambios.

Marcas registradas

IBM, el logotipo de IBM , ibm.com, son marcas registradas de IBM Corporation, registradas en muchas jurisdicciones de todo el mundo. Hay disponible una lista actual de marcas registradas de IBM en la web en "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas.

Microsoft y Windows son marcas registradas de Microsoft Corporation en EE.UU. y/o en otros países.

UNIX es una marca registrada de Open Group en Estados Unidos y en otros países.

Linux es una marca registrada de Linus Torvalds en Estados Unidos y en otros países.

Este producto incluye software desarrollado por Eclipse Project (http://www.eclipse.org/).

Java y todas las marcas registradas y logotipos son marcas registradas de Oracle o sus afiliados.

Número Pieza:

(1P) P/N: